[과목2]

안드로이드
(Android)

차 례

제14장 서비스 및 방송 수신자

14.1 노티피케이션
14.2 방송 수신자
14.3 알람
14.4 서비스

안드로이드 예상문제

제1장 안드로이드 둘러보기

1.1 안드로이드 둘러보기

1.1.1 모바일 단말기와 무선 인터넷

-스마트폰과 태블릿 PC

■ 스마트폰의 의미

- 손 안의 PC 혹은 스마트폰은 모바일 인터넷 단말기
- 과거 일부 비즈니스 계층과 얼리 어댑터의 전유물에서 최근 일반 대중의 생활로 빠르게 확산
- 대표적인 스마트폰
  -세계 최초 1992년 IBM의 Simon
  -2004년도 RIM이 출시한 블랙베리
  -2007년도 애플의 아이폰
- 국내에서는 WIPI의 휴대폰 의무 탑재 폐지 이후 애플 아이폰, 모토롤라의 모토로이, 삼성전자의 갤럭시 등 출시
- 초기와는 달리 네트워크, 단말기, 컨텐츠 측면에서 스마트폰을 위한 환경이 조성 → 손쉽게인터넷 사용
- 특히 사용자의 기호를 반영한 양질의 컨텐츠가 다수 개발

-세계 스마트폰 시장 전망

표 1.1 세계 스마트폰 시장 전망

구분		'07년	'08년	'09년	'10년	'11년	'12년	'13년
휴대폰 판매수		1,151	1,209	1,114	1,202	1,306	1,432	1,568
스마트폰	판매수	121	143	178	254	351	469	604
	성장률	49	18	24	43	38	34	29
	비중	10.5	11.8	15.9	21.1	26.9	32.8	38.5

출처: 삼성경제연구소 CEO Information 741호 (단위: 백만대, %)

- 세계 태블릿 PC 전망

표 1.2 세계 태블릿 PC 전망

기관	2010년	2011년	2012년	2013년
Gartner(2010.10)	19	55	103	154
JP Morgan(2010.10)	15	36	66	-
BNP Paribas(2010.9)	15	45	-	-
Morgan Stanley(2010.9)	15	50	72	85

1.1.2 무선 인터넷

-무선 인터넷의 의의

■유선 케이블 대신에 전파를 사용하며 이동 중에 무선 통신을 기반으로 수행하는 인터넷을 통칭

■전파는 물리적 특성상 지역적·국가적 경계선이 없고 제한된 자원

■전파는 유선 케이블 없이 통신을 가능하게 하는 편의성을 제공하기 때문에 정보통신 분야에서 주목 대상

-스마트폰 대중화의 핵심 모바일 인프라

■전파를 이용한 무선통신과 정보기술의 발전

■특히 3G 이동통신과 와이파이 → 모바일 빅뱅

-와이파이와 3G 이동통신

그림 1.1 이동통신

1.2 모바일 플랫폼

1.2.1 개요

-모바일 플랫폼이란?

■무선 인터넷 서비스를 제공하기 위하여 PC의 운영체제와 담당하는 미들웨어에 해당하는 기본 소프트웨어

■모바일 플랫폼을 도입함으로써 텍스트 위주의 서비스에서 멀티미디어 서비스 제공

-모바일 플랫폼 방식

표 1.3 모바일 플랫폼 방식

종류	장단점	예
인터프리터 방식 (VM 방식)	- 보안성이 좋음 - 많은 자원을 필요 - 성능 저하 - 하드웨어 독립적	- Sun의 J2ME - 신지소프트의 GVM, SK_VM
바이너리 방식 (Native 방식)	- 우수한 성능 - 보안에 취약 - 안정성이 떨어짐 - 하드웨어 의존적	- Qualcomm의 BREW - 모빌탑의 MAP

1.2.2 초기 모바일 플랫폼

-기능

■단문 메시지와 텍스트 기반의 컨텐츠, 저속 및 고비용, 불안전한 구동 기술

-종류 및 특징

표 1.4 초기 모바일 플랫폼

종류	특징
WAP	WAP 포럼을 통해 표준화 WAP 및 WML에 대한 이해 필요 별도의 컨텐츠 제작이 필요
ME	마이크로소프트사가 개발 게이트웨이가 필요 없기 때문에 투자비 저렴 HTML 컨텐츠(m-HTML) 사용 가능
i-Mode	일본의 NTT DoCoMo사가 시장 선점을 위하여 개발 HTML/HTTP 방식을 채용 컨텐츠 제작 비용 저렴

■WAP: Wireless Application Protocol

■ME: Mobile Explorer

1.2.3 독점 모바일 플랫폼

-춘추 전국 시대

표 1.5 독점 모바일 플랫폼

회사	플랫폼	개발 언어	개발 업체	수행 방식
LGT	KVM	Java	SUN	VM
	키티호크	Java	아로마소프트	VM
SKT	SK-VM	Java	XCE	VM
	GVM	C/C++	신지소프트	VM
	WITOP	Java, C/C++	SKT	VM
KTF	MAP	C/C++	Mobiletop	Native
	BREW	C/C++	Qualcomm	Native

-독점 플랫폼으로 인한 문제점

■CP의 컨텐츠 개발의 어려움

■망 공급자와의 관계 등이 너무 복잡

■유선 인터넷 사용자와 같은 다양성 충족이 어려움

1.2.4 WIPI

-개요

■Wireless Internet Platform for Interoperability

■정통부에서 Java와 C/C++ 언어를 모두 포함하는 한국형 무선 인터넷 표준 플랫폼

■비동기 IMT-2000으로 국제 표준으로 상정

■2005년 4월 1일부터 국내 휴대폰에 탑재 의무화

■2009년 4월 1일부터 국내 휴대폰에 탑재 자율화 → 모바일 컨텐츠 산업의 활성화 필요

-강점

-■이통사와 독립적인 플랫폼 사용으로 컨텐츠의 중복 개발 방지

■ 플랫폼의 국산화 → 해외 로열티 유출 감소
■ 모바일 산업 보호. 외국산 단말기 진입 제한으로 제조사들의 내수시장 지배
-문제점
■ 글로벌 추세에 대한 적절한 대응 실패 → 모바일 컨텐츠 산업의 후진
■ 쉽지 않은 WIPI의 세계화 → 국내 단말기 제조사만의 독점
■ 참여사들의 역할 누수 → 실패
  • 이통사는 이윤만 추구
  • 제조사는 WIPI 보급에 등한
  • 표준화단체는 WIPI의 개선에 늑장 대응

1.3 모바일 시장은 전쟁터

1.3.1 모바일 생태계

-모바일 생태계의 변화

그림 1.2 모바일 생태계 변화

-전통적 가치사슬
-구성원
■ 컨텐츠, 플랫폼, 네트워크, 단말기 관련 회사
■ 이통사들이 네트워크와 플랫폼을 장악
-컨텐츠 유통 경로
■ CP의 컨텐츠 개발 → 이통사의 검수 → CP가 이통사 서버에 컨텐츠 업로드 → 소비자에게 판매
■ CP가 개발한 컨텐츠가 이통사의 검수과정을 통과하지 못하면?
-네트워크와 주파수를 보유한 사업자가 가치사슬의 주도권 확보
■ 네트워크는 막대한 초기 투자 비용 요구, 그러나 가입자당 한계비용은 거의 없음
■ 주파수는 제한된 자원으로 대부분 정부 차원에서 관리
-이통사가 통신 서비스뿐만 아니라 단말기 제조와 유통, 단말기 OS와 미들웨어, 컨텐츠와 애플리케이션에 막대한 영향

-생태계 변화 배경
■ 다양한 멀티미디어 서비스 채택 강화
■ 멀티미디어 컨텐츠 사용 증가
■ 단말기 개발 기간 단축 및 고성능화
■ 다양한 애플리케이션 및 솔루션에 대한 사용 요구 증가
■ 데이터 처리 속도 및 메모리 용량의 급격한 증가
■ 무선 네트워크(WiFi 포함) 가용성 ↑
■ 관련 소프트웨어 재사용 용이
■ 휴대폰과 모바일 서비스 연계 비즈니스 모델 등장

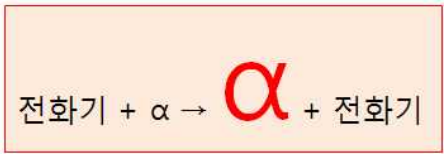


그림 1.3

-애플 아이폰 폭풍
■ 기존의 폐쇄적인 유통 구조 → 서비스 플랫폼 중심으로 개방적 유통 구조
■ 모바일 시장 구조 변화

그림 1.4

1.3.2 모바일 업체의 전략

-구글의 전략
■ 궁극적으로 모바일 인터넷을 클라우드 환경에서 구현
■ 무상일뿐 아니라 매력적인 서비스/친화성/오픈소스 방식의 개발
■ 무상의 모바일 플랫폼 제공 → 스마트폰 저가 → 인터넷 연결 가능한 단말기 증가 → 구글이 제공하는 서비스의 잠재적 고객 증가 → 무상 서비스를 통하여 사용자 확보 → 광고를 통한 수익 증대
■ BRICs 등의 신흥국은 PC보다는 휴대폰이 인터넷 접속 수단

■서비스를 위한 네트워크 접속 비용 최소화 유도
■오픈 소스를 통한 전세계 소프트웨어 개발자 유치 효과

구글이 보유한 Web 2.0 서비스를 유통시키기 위한 모바일 채널 확보

-애플
■컨텐츠, 플랫폼, 단말기로 이어지는 수직 결합된 비즈니스 모형
■서비스 플랫폼을 중심으로 개방적 생태계 구현

그림 1.5

-삼성
■010년 바다 플랫폼을 탑재한 리눅스 스마트폰 '웨이브'
■바다 플랫폼은 SNS, LBS, 커머스 등과 같은 융합 서비스 개발이 가능한 특징
■현재 주로 저가 스마트폰에 바다 플랫폼 활용
-노키아
■유무선 통합 엔터테인먼트 포털 게이트웨이인 OVI
■2010년 초 심비안(Symbian) 소스 코드 공개
■펄프 회사 → 전선회사 → 세계 1위 휴대폰 단말기 회사 → 서비스 사업자?

-과제 혹은 문제점
■구글
  • 이통사의 협조가 필요
  • 그러나 이통사는 구글의 잠재적 경쟁자
■애플
  • 빈약한 단말기 제품 라인
  • 혁신적 UI에 대한 고객 충성도의 지속성?
  • 제한된 이통사를 통한 서비스 확장성?

■ 노키아

- 인수합병 및 전략적 제휴 노력 → 그러나 미흡한 컨텐츠 수준
- 공개된 심비안에 대한 경쟁사들의 채택이 미지수
- 심비안의 인터페이스에 대한 불만족
- 이통사들의 참여 여부

플랫폼 혹은 소스의 개방화 → 양날의 검: 신규 업체에 주도권 빼기기 쉬움

1.3.3 컨텐츠 장터

-업체별 스마트폰 플랫폼과 컨텐츠 장터

표 1.6

업체	플랫폼	개방/폐쇄	컨텐츠 장터
애플	아이폰 OS	폐쇄형	애플 앱스토어
구글	안드로이드	개방형	안드로이드마켓
RIM	블랙베리	폐쇄형	앱월드
노키아	심비안	개방형	오비 스토어
MS	윈도모바일	개방형	윈도 마켓 플레이스
삼성	바다	개방형	삼성 애플리케이션 스토어

• AppStore와 Android Market

표 1.7

	AppStore	Android Market
서비스 시기	2008. 7. 11	2008. 8.28
컨텐츠 등록	Apple의 통제	자율
개발도구 사용	등록자만 가능	누구나 가능
수익 분배	CP와 Apple이 7:3	CP와 이통사(혹은 솔루션업체)가 7:3
VoIP	이통사와의 관계로 불허	허용
단말기 출시	2007. 6. 29	2008. 10. 22
단말기 제조	Apple	모든 단말기 제조 업체

-WAC
■ Wholesale App. Community
■ 2011년 초 구축 예정
■ 세계 24개 통신 및 단말 제조 업체가 공동 참여하는 개방형 컨텐츠 거래 장터, 즉 수퍼 앱스토어
■ 참여사들의 운영 플랫폼이 다른 상황에서 표준화 유도의 어려움
■ 참여사들의 기여도 분담?
■ 1년이란 시간적 촉박성
■ 제2의 WIPI 가능성
■ 성공 여부는 무엇보다도 참여사들의 의지와 실행력

1.3.4 PC와 스마트폰 시장

-PC 시장
■ 80년대 초 PC 시장의 최강자는 애플
■ 애플은 개인용 컴퓨터 시대를 만든 주역이자 가능성을 확인
■ 애플은 폐쇄형 시스템
  • 매킨토시는 하드웨어, 운영체제, 드라이버 모두 애플이 개발
  • 애플리케이션과 주변기기 영역만 외부에 개방

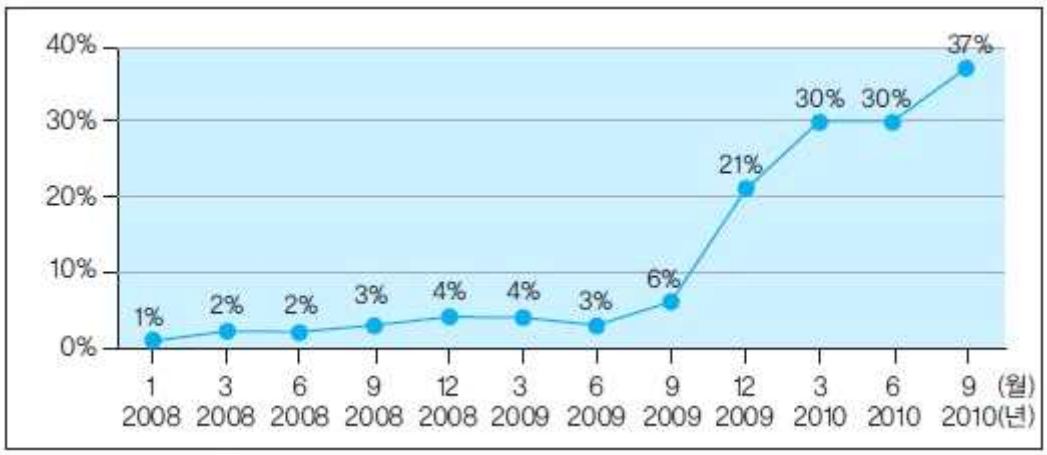
I-BM은 오픈 아키텍쳐
  • 다양한 운영체제와 하드웨어를 활용할 수 있는 개방형 구조
  • 세계의 다양한 업체(컴팩, HP 등)들이 IBM 호환 시장에 참여
  • MS도 소프트웨어 제국 건설 발판 마련

-애플의 몰락
  • 혁신적인 UI 운영체제 및 우수한 디자인
  • 고가 및 많은 적군
  • 외부 업체의 참여와 상생에 소홀

-스마트폰 시장
■ 애플이 스마트폰의 선두주자는 아님
■ 애플
  • 폐쇄적 모형: 아이폰 하드웨어와 운영체제도 애플이 주도
  • 앱스토어도 아이폰만을 위한 서비스
  • 이통사도 애플이 OK 해야 아이폰을 공급 → 다수의 잠재적 적군 (MWC 2010의 WAC이 증거)
  • 외부 SW업체 및 개발자들은 아이폰에 열광
■ 구글이나 MS
  • 다양한 단말기 제조업체, 이통사가 자사 플랫폼을 탑재한 스마트폰 제조
  • 컨텐츠 거래 장터도 애플보다는 유연하며 이통사와의 공존을 강조
■ 역사의 반복? 그러면 애플은 왜?
  • 풍부한 컨텐츠
  • IPod와 iTunes 성공 경험

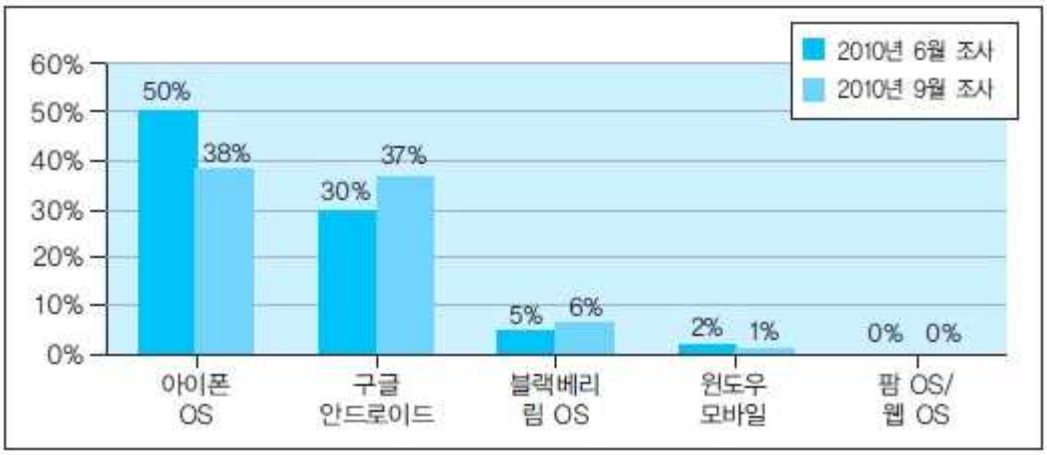
-안드로이드 플랫폼에 대한 선호도 추이

그래프 1.1



-스마트폰을 위한 모바일 플랫폼 선호도

그래프 1.2



1.4 안드로이드 개요

1.4.1 개요

-안드로이드 의미

- 인간의 모습을 한 로봇
- 안드로이드는 검색 광고 업계의 거인 구글이 모바일 시장에 진출하기 위하여 인수한 조그마한 신생 업체의 이름에 불과
- 그러나 이제 안드로이드가 아이폰과 함께 현재 스마트폰을 이끌어나갈 주요 단어

-Google?

1.4.2 안드로이드 플랫폼

-개요

■ Open, Complete. Free한 플랫폼
■ 운영체제, 미들웨어, 자바로 개발된 핵심 애플리케이션(key application)을 포함하는 모바일 기기의 소프트웨어 집합체
■ 단말기에서 하드웨어를 제외한 나머지 모든 소프트웨어 계층
■ 개방형 플랫폼으로 소스 코드를 완전 개방 → 제한 없이 안드로이드 기반의 모바일 기기 제작 가능
■ 2007년 11월 구글과 OHA(Open Handset Alliance)라는 모임에 의해 발표한 모바일 플랫폼
■ 아파치 2.0 라이센스를 가지며 소스 코드로 모든 사람이 빌드 가능
■ 기본적으로 JAVA 프로그래밍 언어를 사용하며 어플리케이션의 빌드, 컴파일, Test 및 디버그를 할 수 있는 SDK를 제공

-특징

■ 애플리케이션 프레임워크: 컴포넌트의 재사용과 대체를 가능케 함
■ Dalvic 가상머신: 모바일 디바이스에 최적화
■ 통합(Integrated) 브라우저: 오픈 소스 Webkit 엔진 기반
■ 최적화된 그래픽: 구글이 만든 2D 그래픽 라이브러리 사용
■ OpenGL ES 1.0 스펙에 기반한 3D 그래픽 사용(하드웨어 가속은 선택 사항)
■ SQLite: 정형화된 데이터 저장공간을 위함
■ 미디어 지원: 일반적 포맷(MPEG4, H.264, MP3, AAC, AMR, JPG, PNG, GIF)을 지원
■ GSM Telephony, Bluetooth, EDGE, 3G, WiFi, 카메라, GPS, 나침반, 가속도계(하드웨어 의존적)
■ 풍부한 개발 환경: 디바이스 에뮬레이터, 디버깅 도구, 메모리 및 성능 프로파일링, Eclipse IDE를 위한 플러그인

1.4.3 안드로이드 아키텍처

-구조

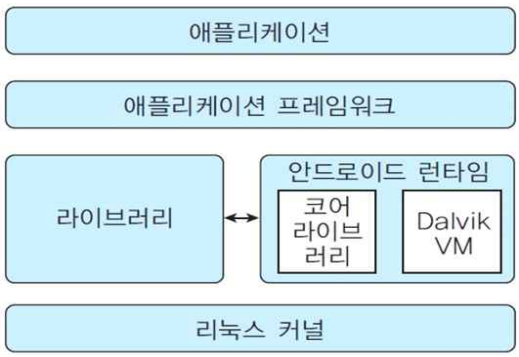


그림 1.6

-애플리케이션

■ Gmail과 연동되는 이메일 클라이언트
■ SMS 문자 메시지 프로그램
■ 캘린더
■ 구글 맵 애플리케이션
■ Webkit 기반 웹 브라우저
■ 컨택트(contact)

■ 네이티브 애플리케이션과 서드파티 애플리케이션이 하드웨어 접근 혹은 실행 권한 등 모든 면에서 동등
  • 동일한 API 사용
  • 동일한 런타임에서 실행

-애플리케이션 프레임워크
■ 자바 기반의 애플리케이션 프레임워크. 대부분 JNI(Java Native Interface)를 통해 native C/C++ 코드로 작성
-라이브러리
■ Binoic Libc는 임베디드 디바이스를 위한 최적화된 libc
  • 직접 구현한 이유: 사용자 영역에서 GPL에 적용되지 않기 위하여, 개별 프로세스마다 포함되는 영역이기 때문에 크기를 최소화하기 위하여, 제한된 CPU의 최적 성능 효과를 위하여
■ Function Library
  • 웹 브라우저를 위한 Webkit
  • PacketVideo의 OpenCORE 플랫폼 기반의 미디어 프레임워크
  • 가벼운 데이터 베이스인 SQLite

-안드로이드 런타임
■ 여러개의 VM을 동시에 실행시키도록 디자인되어 있으며 프로세스 독립과 메모리 관리 쓰-레드를 OS에 의존
■ 모든 안드로이드 애플리케이션은 Dalvic 가상머신 내에서 자신의 인스턴스를 가지고 자신의 프로세스내에서 동작
■ 달빅 VM
  • 런타임 환경을 통해 더 확실하게 HW와 SW를 계층화
  • 임베디드 단말에 최적화된 VM으로 메모리 보호(Memory Protection), 가비지 컬렉션(Garbage-collection), 라이프 사이클 관리 등의 특징
  • Sun의 JVM보다 작은 메모리 환경에서도 잘 실행할 수 있도록 성능 개선의 효과도 얻으면서 자바를 탑재할 때의 라이선스 비용의 문제도 해결

1.5 안드로이드 애플리케이션

1.5.1 개요
-안드로이드 애플리케이션은 일반적으로 자바 프로그래밍 언어로 작성
-소스 코드는 컴파일된 후 클래스 파일을 거쳐 애플리케이션에 필요한 리소스 등과 함께 AAPT라는 안드로이드 도구에 의하여 하나의 패키지로 통합
-아카이브 파일로 압축된 안드로이드 패키지는 .apk라는 확장자를 가지며 하나의 안드로이드 애플리케이션으로 간주
-또한 안드로이드 패키지는 모바일 디바이스에 배포 및 설치하기 위한 수단

1.5.2 애플리케이션 컴포넌트
-개요
■ 안드로이드 애플리케이션은 자바 언어로 작성
■ 컴파일 후 압축된 아카이브 파일(.apk 파일)로 모바일 디바이스에 배포
■ apk 파일 내부에 있는 모든 코드는 하나의 애플리케이션으로 간주

■ 애플리케이션은 4가지 컴포넌트의 조합으로 구성
  • Activity
  • Service
  • Broadcast Receiver
  • Content Provider
■ 애플리케이션의 컴포넌트를 AndroidManifest.xml에 포함
■ Intent는 어떤 작업에 대한 요청, 즉 Activity, Service, Broadcast Receiver 컴포넌트를 활성화하는데 필요한 명령과 데이터를 담은 클래스

-Activity
■ 뷰와 이벤트 응답으로 이루어진 인터페이스
  • 애플리케이션에 하나 이상의 Activity 포함 가능
■ Service
  • 비쥬얼한 사용자 인터페이스 없이 정해지지 않은 시간 동안 내부에서 실행되는 코드
  • 예: 백그라운드 음악 재생, 네트워크 상에서 데이터 전송 등
■ Broadcast Receiver
  • 외부 이벤트를 처리하는데 사용되며 사용자에게 발생한 이벤트의 종류를 알려줌
  • 예: 시간대 변경, 배터리 부족, 사진 촬영, 사용자 언어 설정 변경 등
■ Content Provider
  • 다른 애플리케이션에게 유용한 애플리케이션 데이터의 특정 집합을 생성
  • 애플리케이션 사이에 데이터를 공유할 때 유용

1.5.3 애플리케이션 종류

-Managed Application
■ 가장 기본적이고 주요한 안드로이드 애플리케이션 개발 방식
■ 달빅 VM에서 동작하는 순수 자바로만 개발한 프로그램
■ 안드로이드는 달빅 위에 여러 프레임워크 API를 제공하고 있고, OpenGL이나 Multimedia 등의 라이브러리에서 지원하는 기능 또한 API 형태로 달빅 위에서 이용할 수 있도록 제공
■ 개발자는 SDK에서 제공하는 이클립스 플러그인을 통해서 쉽게 UI를 구성하면서 개발 가능
■ 자바라는 언어의 특성상, 약간의 속도 저하와 예측 불가능한 Garbage collection의 수행이라는 약점
■ 하드웨어나 라이브러리와 밀접하게 결합이 되어야 하는 경우는 기존의 프레임워크 API로는 불충분

-Native Application
■ 달빅 내에서 C/C++으로 구성된 동적 라이브러리를 적재하여 JNI(Java Native Interface) 형태로 함수를 호출하는 방식
■ 구글이 최근 NDK(Native Development Kit)를 따로 배포하여 C/C++의 코드를 손쉽게 .so 형태의 라이브러리로 교차컴파일
■ CPU의 의존도가 높은 프로그램 또한 하드웨어를 직접 다루거나 기존 달빅 API로 구현하기 어려운 경우 효율적
■ 메모리 제어 등 프로그램 개발에 보다 세심한 관리가 필요

-AJAX Application
■ JavaScript와 Ajax 등으로 웹페이지를 작성하고 안드로이드의 브라우저를 통해서 수행하는 방식

■ 웹킷(Webkit) 코어와 SquirrelFish 자바스크립트 엔진을 사용하고 있으므로 PC 기반의 동작과 유사한 기능 제공
■ 구글의 대표 웹 서비스(Gmail, Google Docs 등)가 Ajax로만 구현
■ 항상 브라우저를 통해서 수행이 되어야 하므로 백그라운드 서비스 등이 불가능하고, 시스템이나 프레임워크 내의 접근이 불가능

1.6 안드로이드의 미래

1.6.1 개요

■ 안드로이드는 모바일 컴퓨팅의 미래를 만들어갈 강한 잠재력이 있는 모바일 플랫폼
■ 안드로이드는 오픈 소스 환경이며 여러 모바일 업체와 개발자들이 참여하고 있으며 발전 속도가 매우 빠르다.
■ 안드로이드는 구글 및 OHA의 주도로 이루어지지만, 플랫폼에 포함되어 있는 수많은 라이브러리는 계속 향상된 버전을 적용
  • 예를 들어 리눅스 커널은 버전 2.6.18부터 시작했지만 안드로이드 1.6은 리눅스 커널 2.6.29을 사용하며, 12월에 발표한 안드로이드 2.3는 최신 리눅스 커널인 2.6.35 기반
  • 웹킷의 경우도 오픈 소스의 변경 사항에 따라 끊임없이 버전업

-안드로이드 플랫폼의 버전별 변화

표 1.8 안드로이드 플랫폼의 버전별 변화

버전	코드 이름	발표	API Level	특징 및 기능
1.0		2008. 9.	1	개발툴인 SDK만 발표
1.1		2009. 2.	2	API Level 도입, 마키 지원, 뷰 패딩 개선
1.5	Cupcake	2009. 4.	3	AVD 도입, 비디오 레코딩, 블루투스 지원, 풀스크린 위젯, 음성 인식 지원, 홈스크린, 다국어 지원, 자동 완성 입력기
1.6	Donut	2009. 9.	4	다양한 해상도 지원, 통합 검색, TTS 지원, 카메라와 동영상 UI 갱신, 제스처
2.0	Eclair	2009. 10.	5	익스체인지 지원, 카메라 내장 플래시 지원, 디지털줌, 개선된 가상 키보드, 블루투스 2.1
2.1	Eclair	2010. 1.	7	라이브 배경화면, 버그 수정
2.2	Froyo	2010. 5.	8	성능 개선, 부드러운 UX 제공, 플래시 지원, 애플리케이션 자동 업데이트 기능, 확장 메모리에서 애플리케이션 설치
2.3	Gingerbread	2010. 12.	9	빠른 텍스트 입력과 간편해진 선택 및 복사/붙여쓰기 기능 지원, 화면 해상도 개선 및 3D 영상 지원, NFC(Near-Field Communication) 칩을 이용한 모바일 결재 실현

-코드명 규칙
■ 안드로이드 코드명은 음식 이름을 사용
■ 알파벳 순서대로 명명
-코드명과 버전

그림 1.7 안드로이드 코드명은 음식 이름

1.6.2 안드로이드 전망

-안드로이드 플랫폼의 개방과 무료
■ 성능이 우수하면서도 폐쇄적이면 실패
  • 운영체제의 경우 IBM OS2
  • 컴퓨터의 경우 애플 매킨토시
  • 비디오 레코더의 경우 소니의 베타 방식
■ 안드로이드는 모든 소스를 오픈하며 어떤 모바일 단말기 제조사도 안드로이드 SDK를 사용하고 수정해도 라이선스에 문제가 없다.
■ 안드로이드는 모든 모바일 단말기 뿐만 아니라 임베디드 단말기를 위한 플랫폼으로 널리 사용 가능

-안드로이드 SDK의 편의성과 강력함
■ 아이폰 개발 도구
  • 사용하기 어렵고 주관적일 수 있는 애플리케이션 등록 과정
  • 애플 Object-C 언어는 아이폰 애플리케이션 이외에는 거의 미사용
■ 안드로이드 개발 도구
  • C/C++ 언어로 개발하는 경우도 있지만 자바 프로그래밍 언어를 주로 사용.
  • 더구나 개발 환경도 이클립스라는 좋은 도구의 도움 가능
■ 안드로이드에서는 네이티브 애플리케이션과 서드파티 애플리케이션이 평등 → 미리 설치된 애플리케이션을 확장하거나 아예 대체 가능
■ 모바일 컴퓨팅의 미래가 사용자가 원하는 컨텐츠에 달려 있다면 안드로이드가 모바일 시장에서 승리할 가능성이 농후

-구글의 행보
■ 구글은 안드로이드를 성공적인 범용 임베디드 플랫폼으로 만들려면 좋은 개발자 유치와 단말 제조사들의 지원을 확보
■ 안드로이드 경진 대회와 같은 행사를 통해서 개발자들의 관심을 유도
■ HTC 뿐만 아니라 다른 여러 모바일 단말기 제조사들로 하여금 앞 다투어 안드로이드 단말기를 출시 유도

■ 내비게이션 단말기와 IPTV와 같은 임베디드 환경에도 안드로이드 플랫폼을 적절하고 신속하게 적용

제2장 개발 환경 구축

2.1 교차개발환경

2.1.1 개발환경

-호스트 시스템과 타깃 시스템

그림 2.1

2.1.2 툴체인

-필요한 소프트웨어

■ 자바개발도구 JDK 5.0 이상
■ 안드로이드 SDK
■ 이클립스
■ ADT 플러그인
■ 자바 네이티브 인터페이스

-다운로드할 폴더 생성

■ D:₩android 폴더 생성
■ D:₩android₩software 폴더 생성

-설치과정

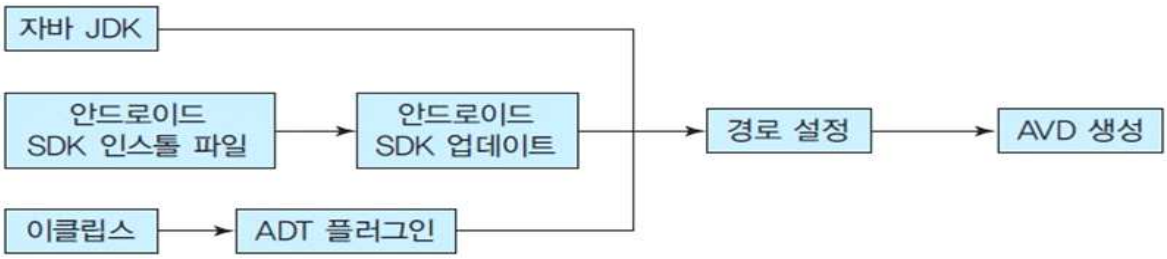


그림 2.2

2.2 JDK와 안드로이드 SDK 설치

2.2.1 JDK 설치

-D:₩android₩software₩jdk-6u22-windows-i586.exe 파일을 2클릭하여 설치

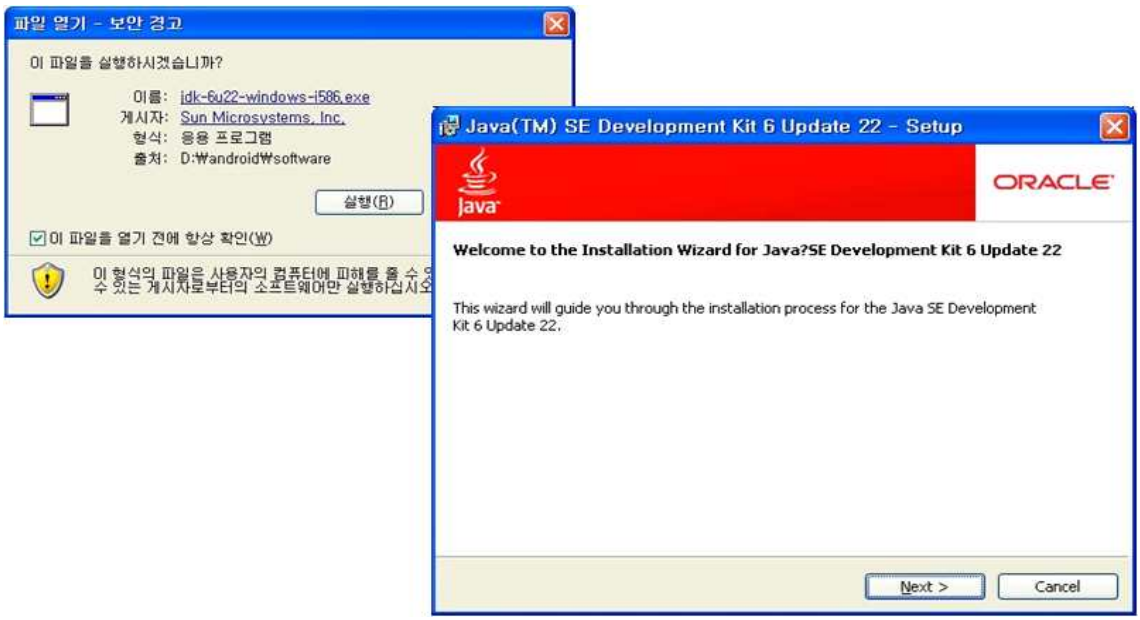


그림 2.3 설치

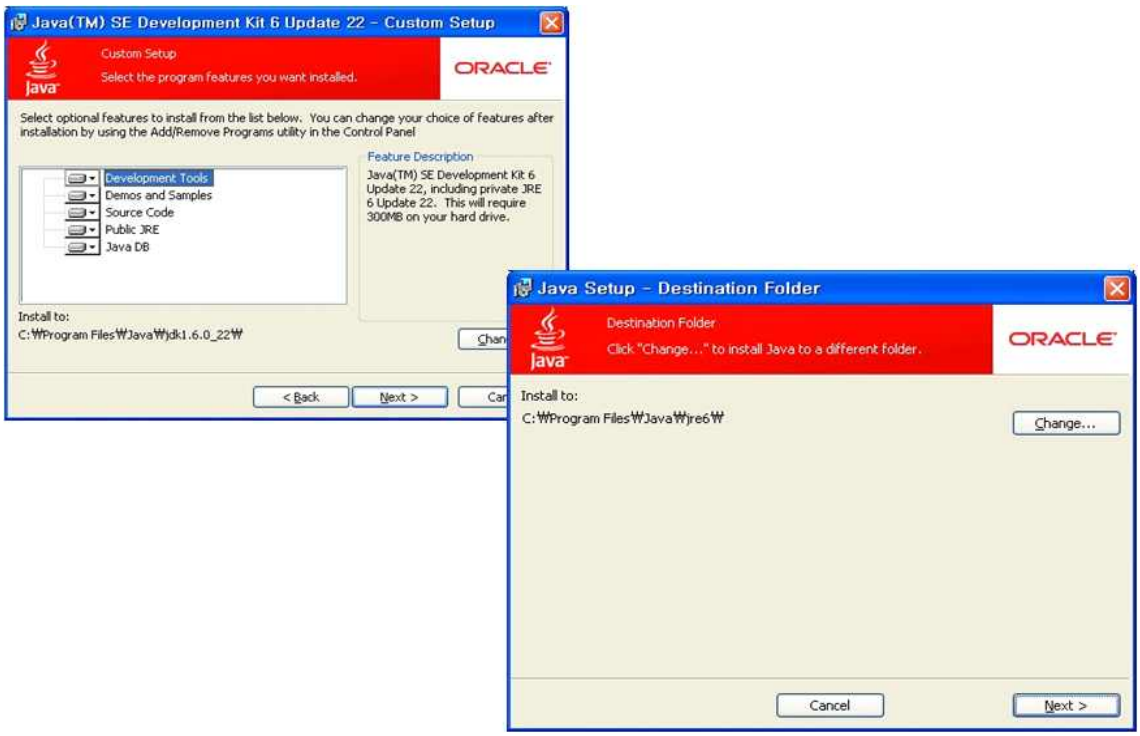


그림 2.4 설치

-등록 과정이 나타나지만 무시해도 좋다.

그림 2.5 설치

-자바 SDK의 각종 Tool에 대한 경로 설정

■ 실행창에서 좀더 쉽게 Tool을 이용하기 위한 목적

■ 바탕화면의 내 컴퓨터를 우클릭 - 속성 - 고급 탭의 환경 변수 버튼을 클릭

■ 만약 하단 시스템 변수 부분에 Path 혹은 PATH 라는 변수가 있다면 변수를 선택한 후 편집 버튼을 클릭

■ 환경 변수 창 확인 버튼 클릭

그림 2.6 설치

2.3 안드로이드 SDK 인스톨 파일 설치

-D:₩android₩software₩installer_r08-windows.exe 파일을 2 클릭

그림 2.7 설치

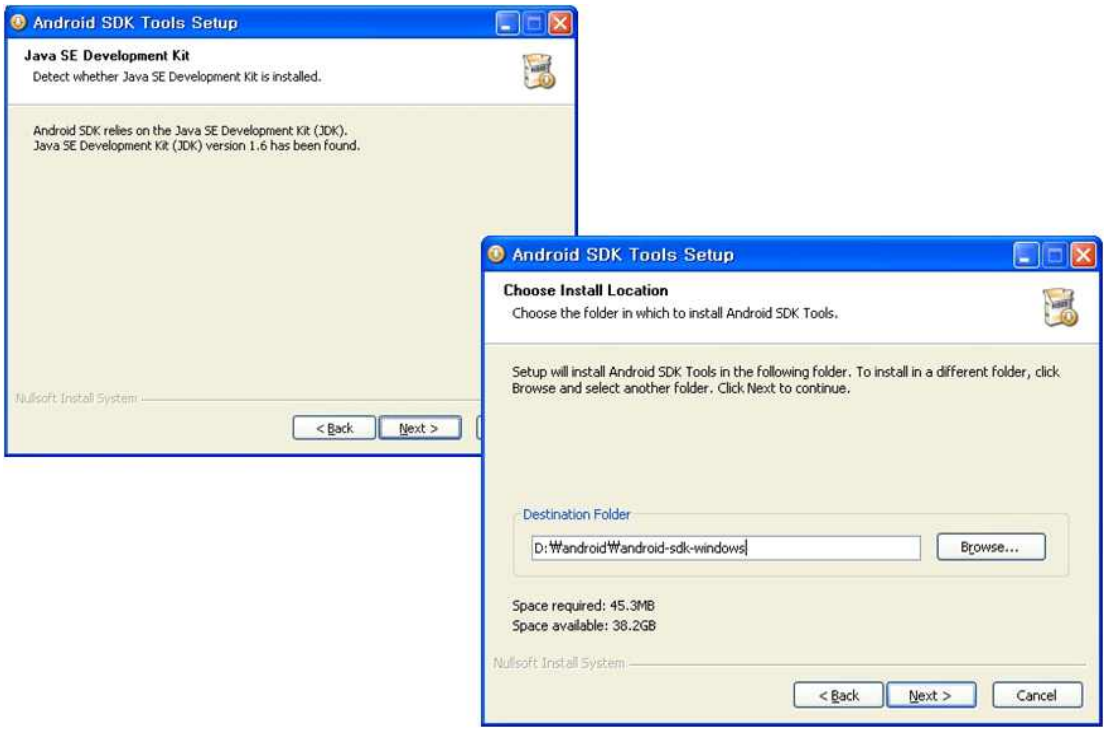


그림 2.8 설치

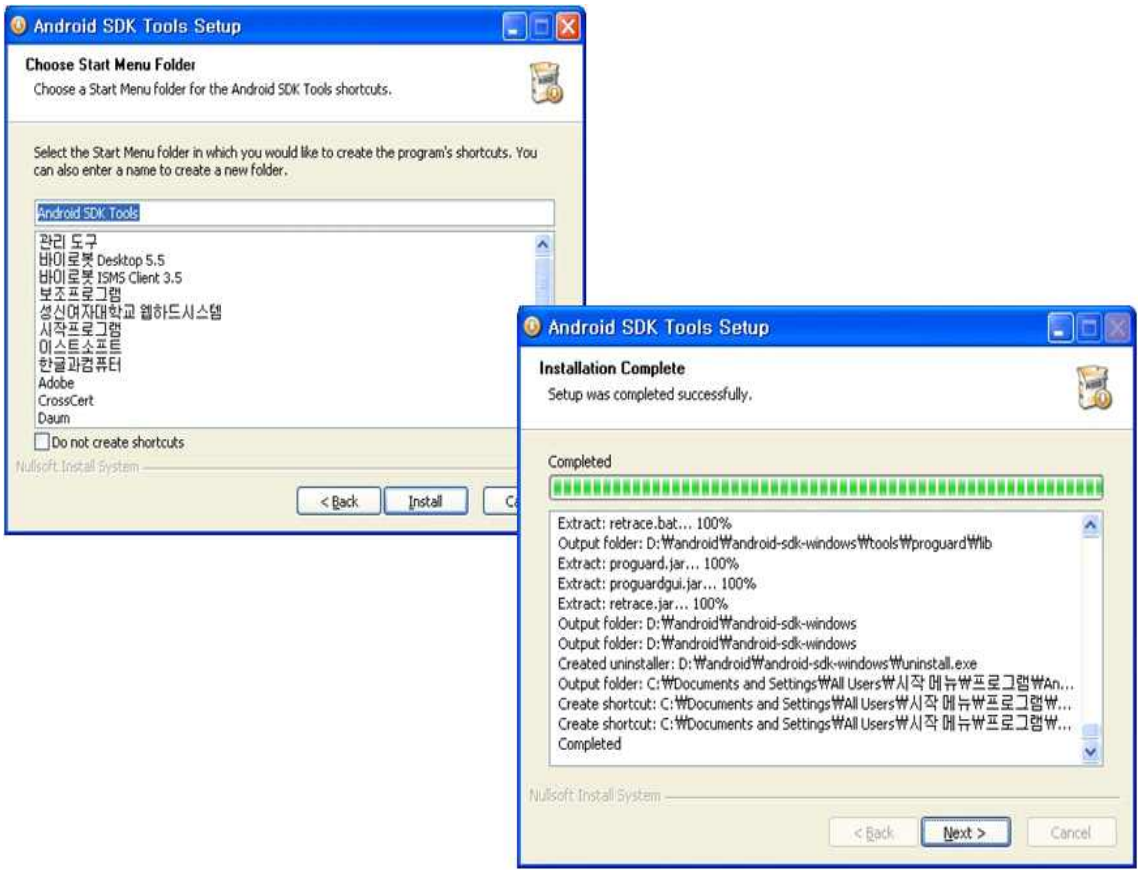


그림 2.9 설치

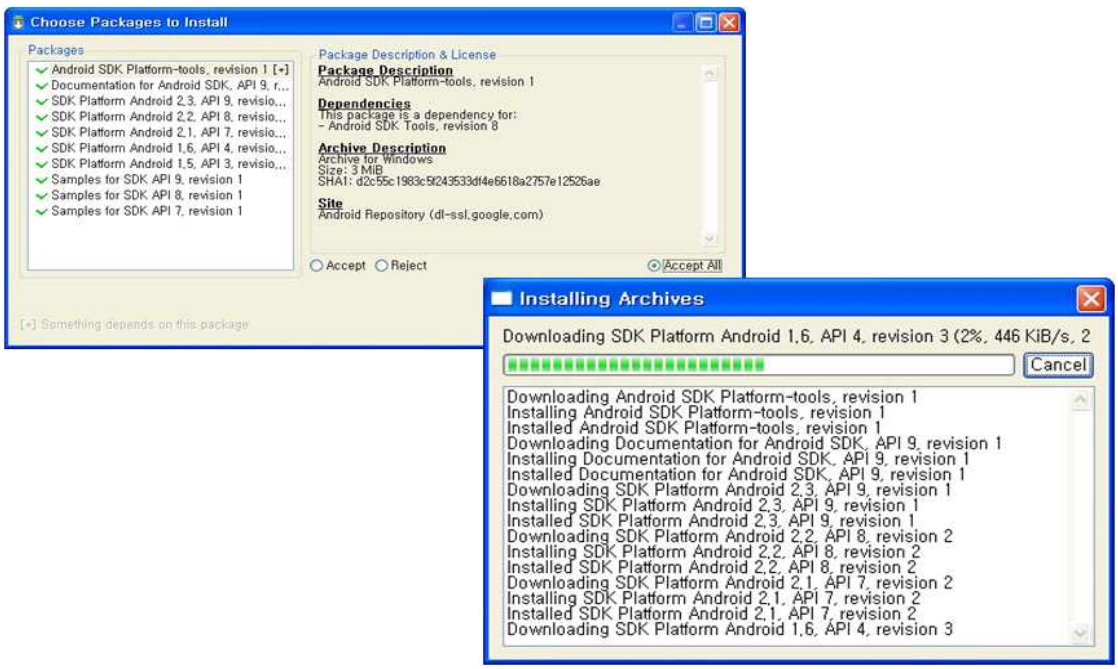


그림 2.10 설치

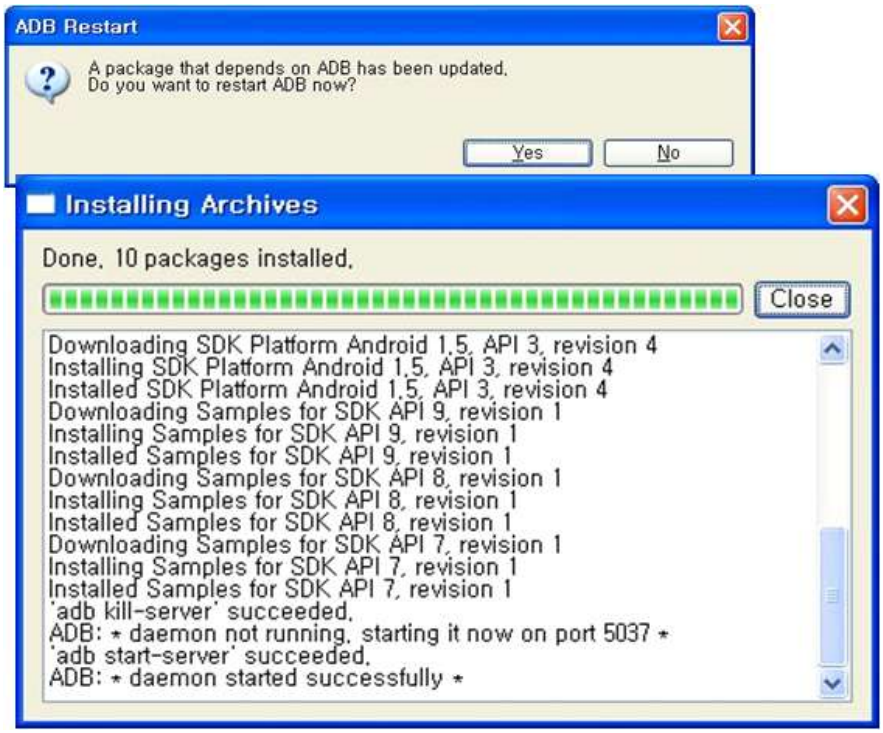


그림 2.11 설치

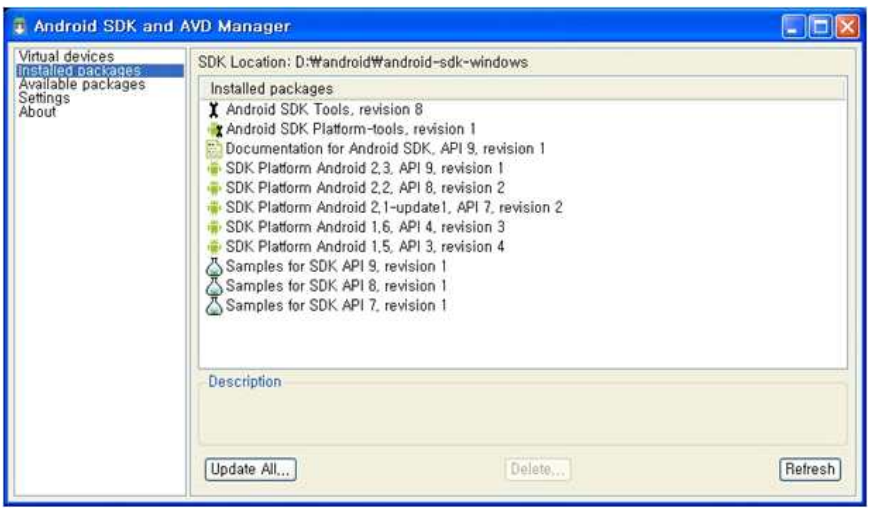


그림 2.12 설치

-경로 설정

■환경 변수 창 확인 버튼 클릭

그림 2.13 설치

2.4 이클립스 설치 및 설정

2.4.1 이클립스 설치

-설치

■ D:₩android₩software₩eclipse-java-galileo-SR1-win32.zip 압축 파일을 D:₩android에 압축 풀기

■ D:₩android₩eclipse에 있는 eclipse 실행 파일을 우클릭하여 바로 가기 만들기 선택

■ 새로 생성된 eclipse 바로 가기 아이콘을 바탕화면으로 이동

■ 바탕화면의 eclipse 아이콘을 2클릭하여 실행

-작업 공간 설정

■ Workspace Launcher의 Workspace를 D:₩android₩workspace로 변경하고 기본값으로 체크하면 Eclipse IDE 환영 창이 나타남 → 환영 창을 닫음

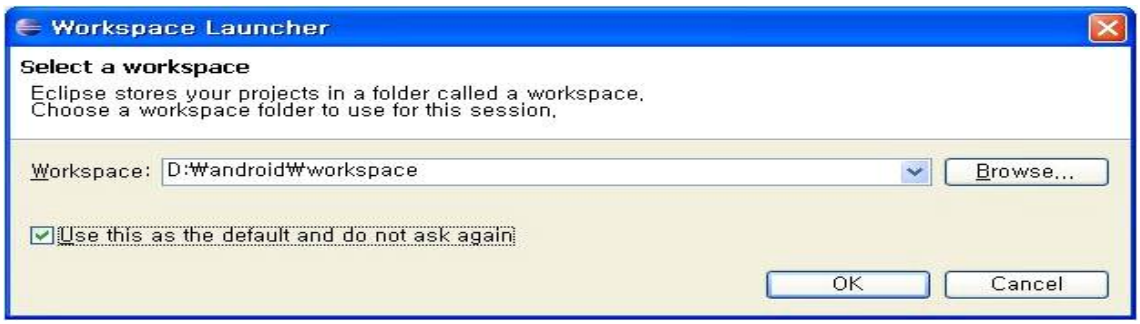


그림 2.14 설치

-설치된 이클립스 초기 화면

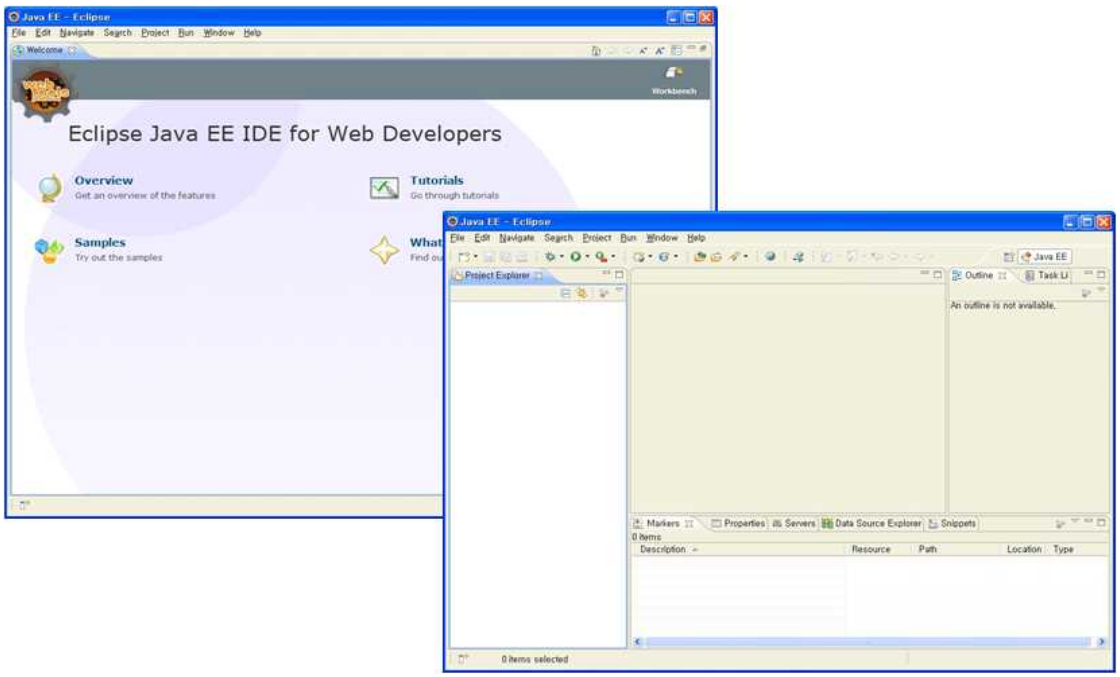


그림 2.15 설치

2.4.2 ADT 플러그인 설치

-이클립스에서 Help → Install New Software 선택

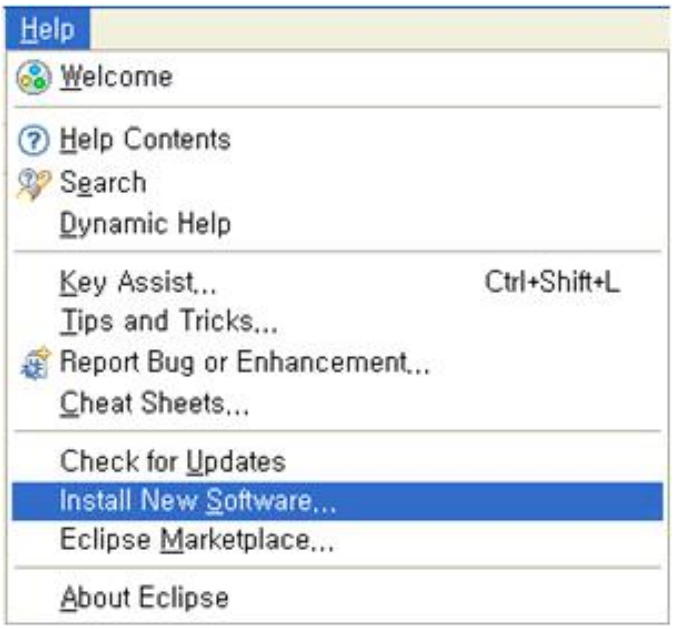


그림 2.16

-Install 창에서 Add … 탭을 선택

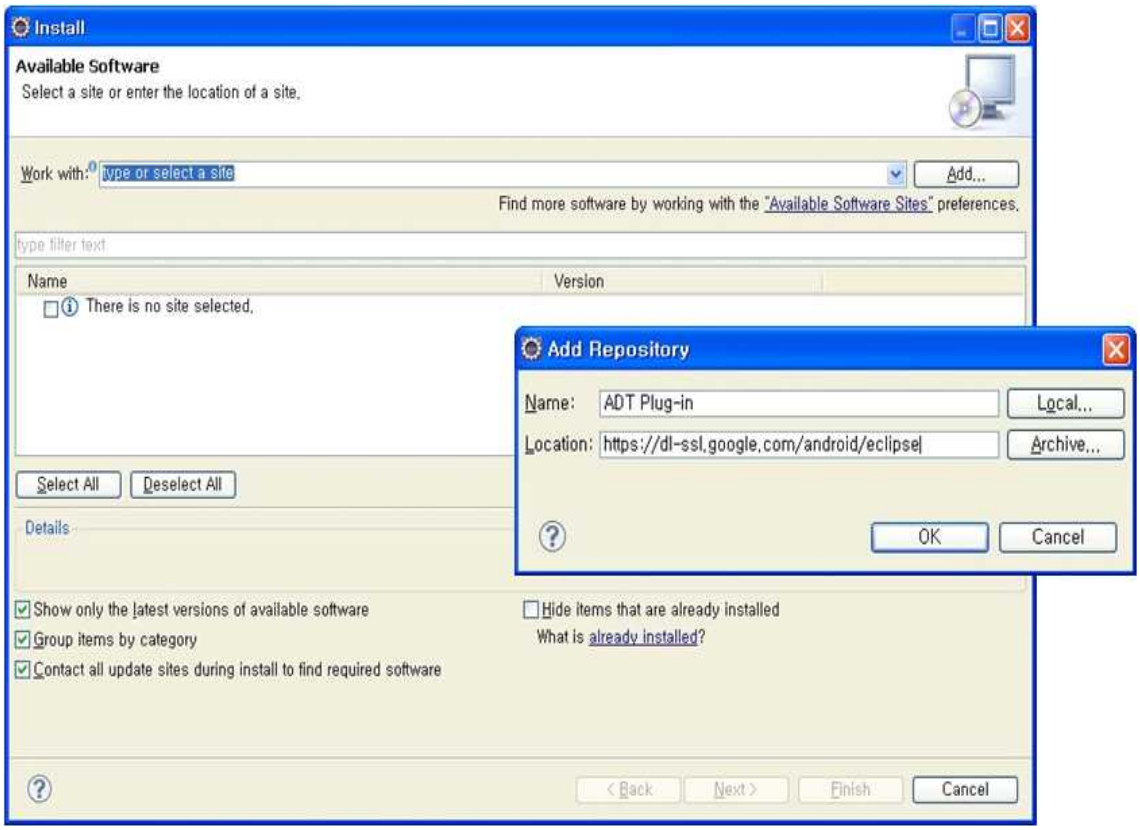


그림 2.17

-Install 창의 중앙에 Developer Tools를 선택하면 [Next] 버튼이 활성화 → [Next] 버튼을 클릭

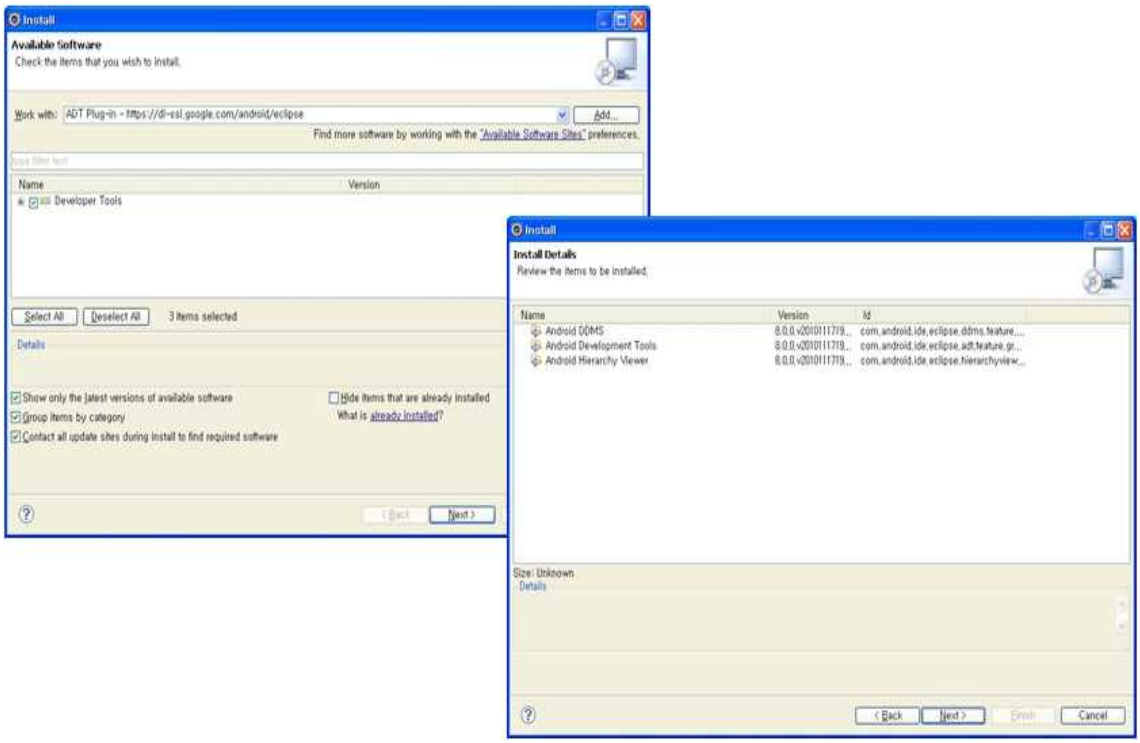


그림 2.18

-Install 창의 Review Licenses에 대하여 라이선스 동의 항목을 체크하고 [Finish] 버튼 클릭

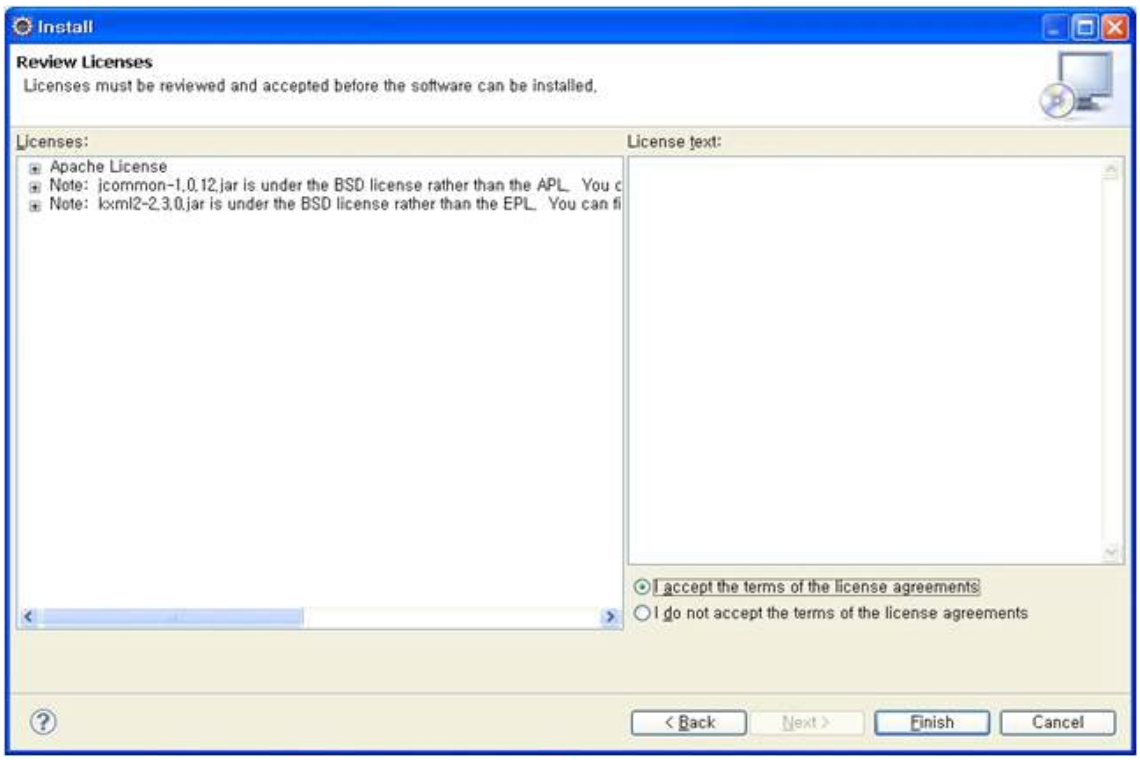


그림 2.19

-Install 창에서 각종 플러그인이 오랜 시간 동안 설치되며 과정을 보여줌. 설치 도중 Security Warning 창이 나타나면 [OK] 버튼 클릭

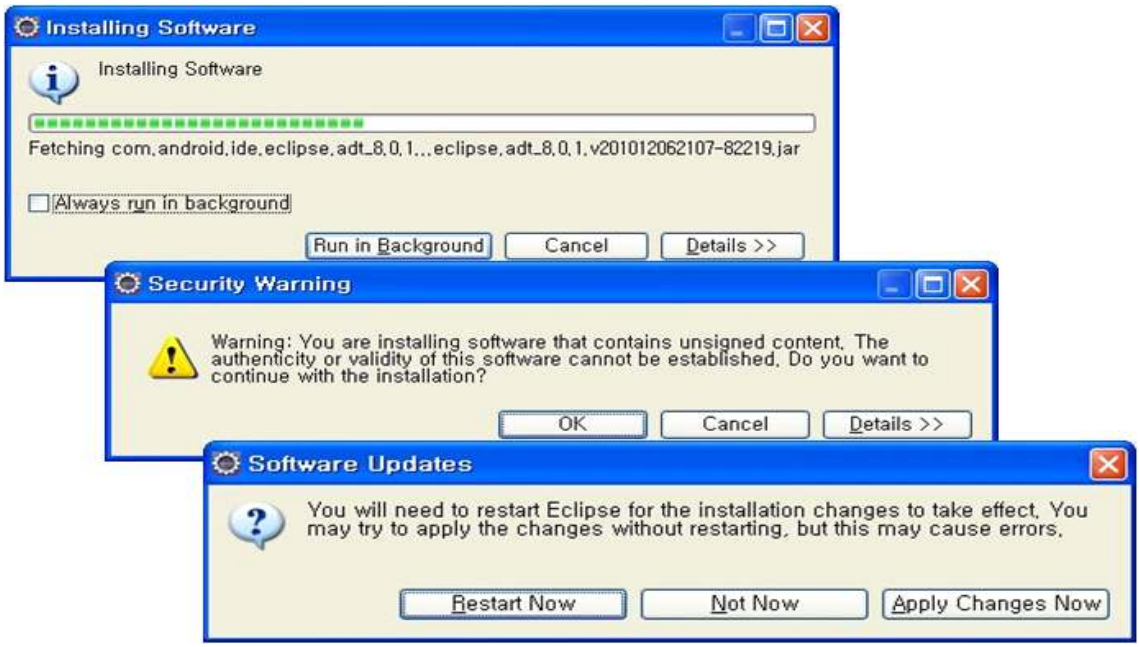


그림 2.20

-ADT가 설치된 이클립스 - 자바 퍼스펙티브

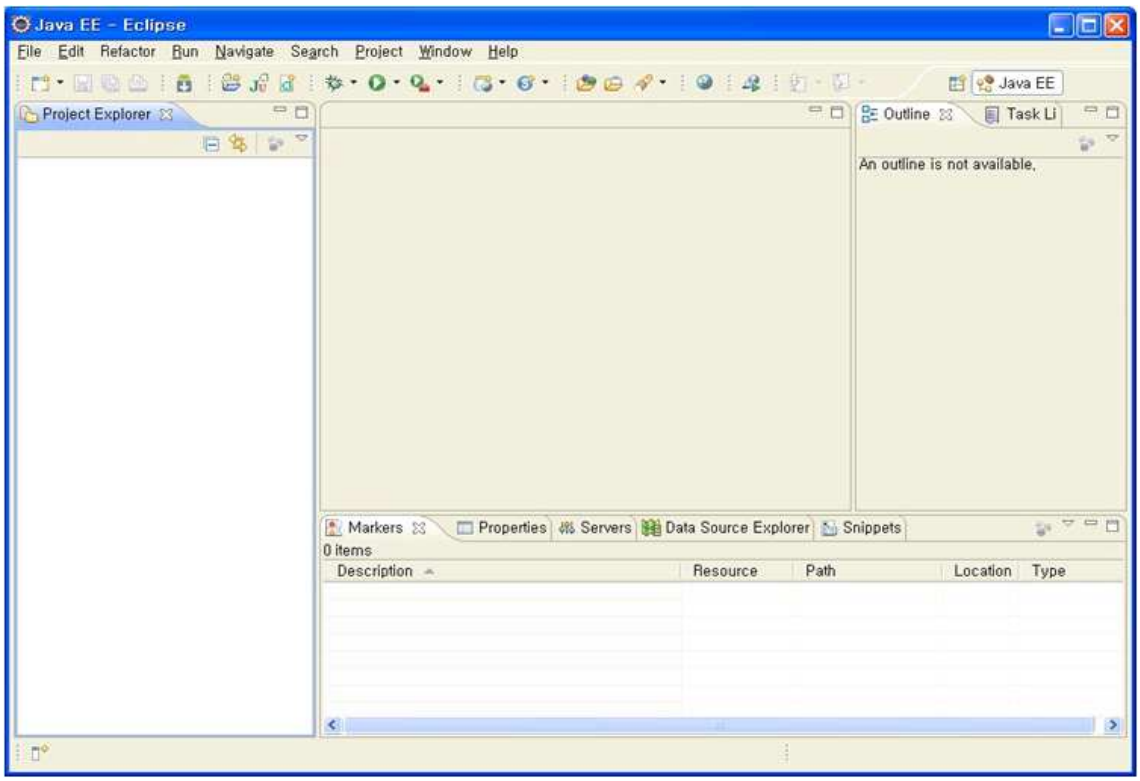


그림 2.21

2.4.3 안드로이드 SDK 경로 설정

-이클립스가 다시 시작되면 Windows → Preference 메뉴를 선택

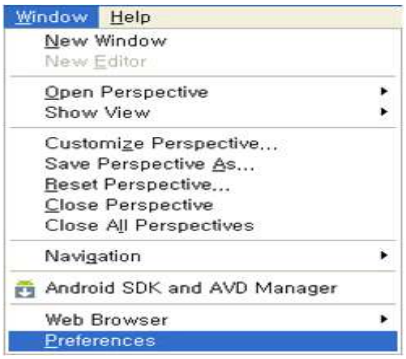


그림 2.22

-Preference 창이 나타나면 Android를 선택 → Preference 창의 우측 [Browse …] 버튼을 클릭

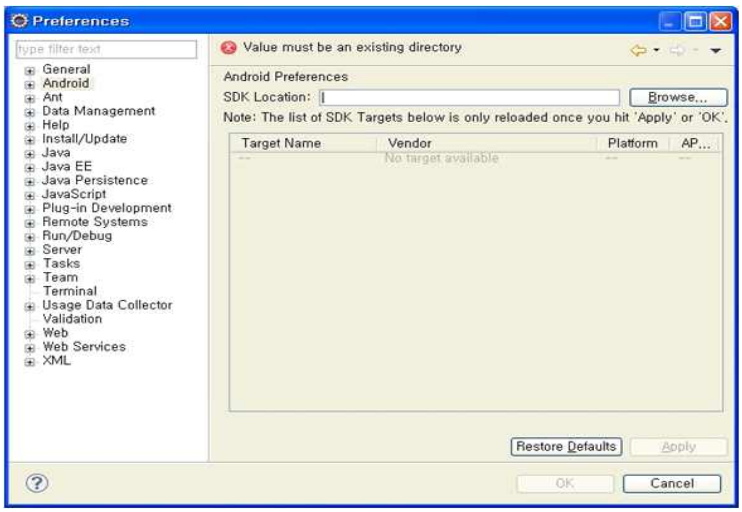


그림 2.23

-폴더 찾아보기 창이 나타나면 D:₩android₩android-sdk -windows 폴더를 선택하고 [확인]

그림 2.24

- Preference 창의 하단에 있는 [Apply] 버튼과 [OK] 버튼을 차례대로 클릭

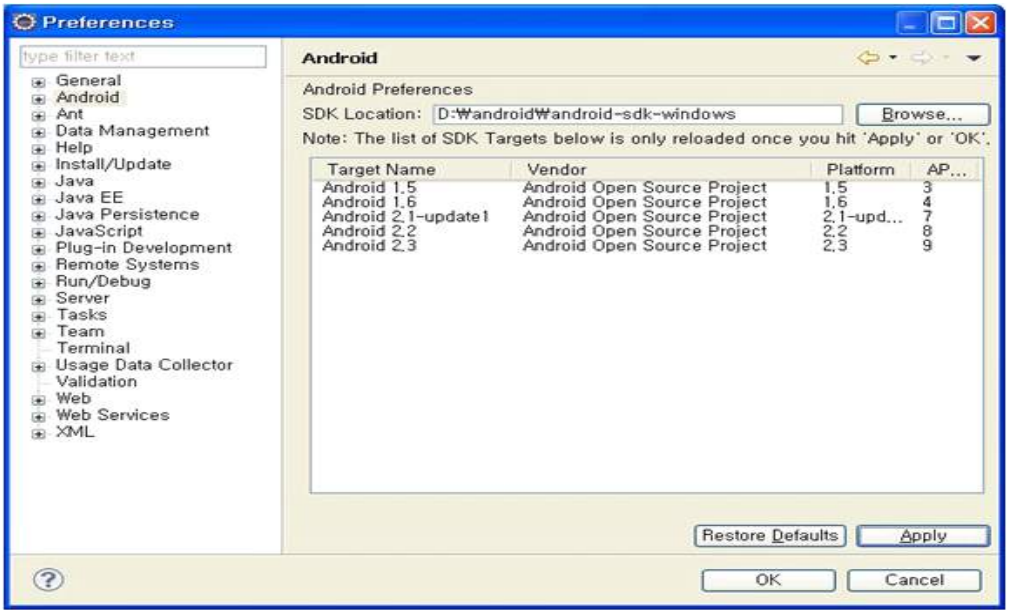


그림 2.25

2.5 가상 단말기 AVD

2.5.1 AVD 생성

-AVD란

- Android Virtual Devices
- SDK 1.5부터 에뮬레이터를 실행하기 위하여 최소 1개의 AVD 필요
- AVD로 인하여 여러 버전의 안드로이드 디바이스를 위한 애플리케이션을 각각의 버전과 SDK Add-On에 맞게 테스트 가능
- 예를 들어 카메라가 있는 경우, 쿼티 자판이 있는 경우, 1.1 SDK 탑재한 단말, 1.5 SDK를 탑재한 디바이스 등 여러 가지 구성을 가지고 있는 가상의 디바이스를 지원 가능하게 함
- 각 AVD마다 하나의 안드로이드 에뮬레이터를 구동할 수 있음

-이클립스 툴바에서 [Android AVD and SDK Manager] 버튼을 클릭 혹은 …

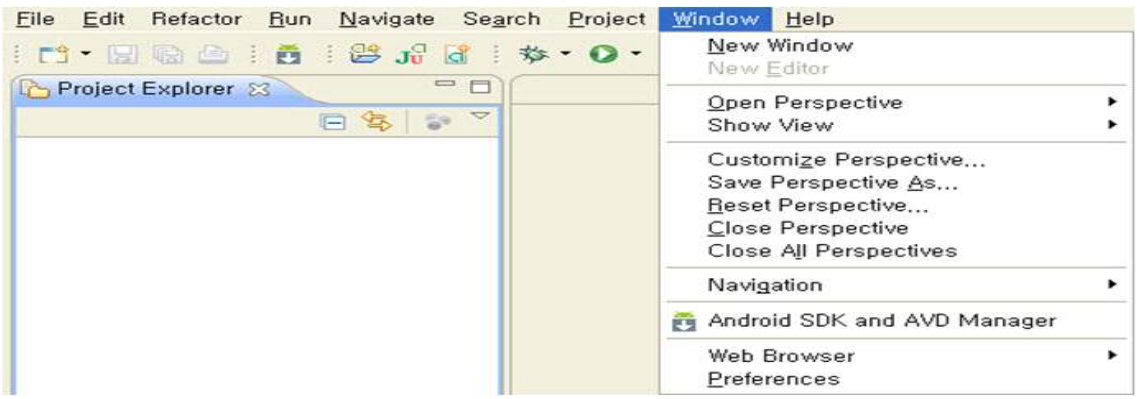


그림 2.26

-Android AVD and SDK Manager 창의 우측에 있는 [New…] 버튼을 클릭

그림 2.27

-Create new AVD 창의 Name 항목에 적절한 디바이스 이름 선택

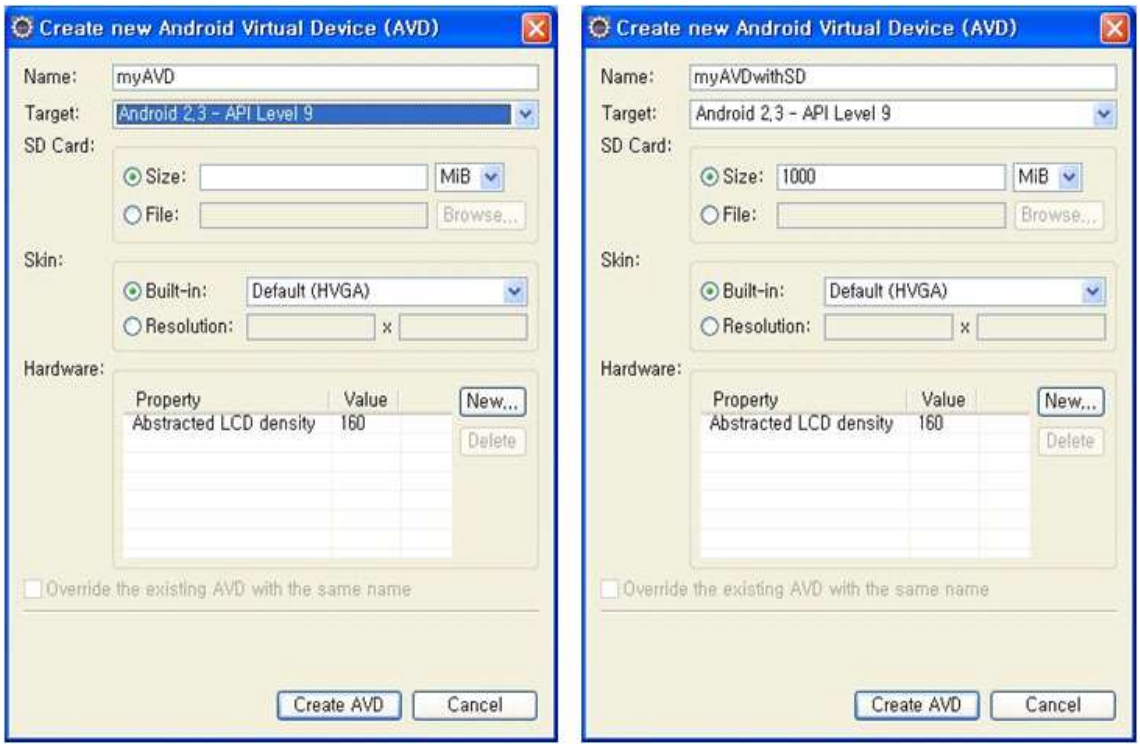


그림 2.28

2.5.2 가상 SD 카드 생성

-SD 카드의 생성

■ SD 카드의 크기를 입력

■ 혹은 기존에 생성한 SD 카드를 사용하려면 File을 선택한 후 [Browse…] 버튼을 눌러 가상 SD 카드 파일을 선택

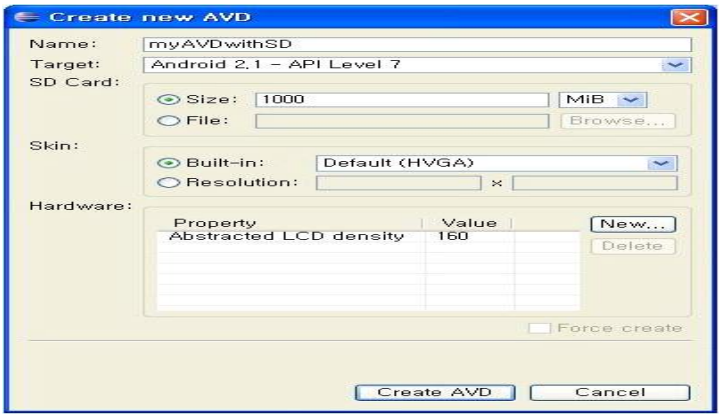


그림 2.29

2.5.3 AVD와 가상 SD 카드 확인

-이클립스 툴바에서 [Android AVD and SDK Manager] 버튼을 클릭 → 왼쪽의 Virtual Devices 선택

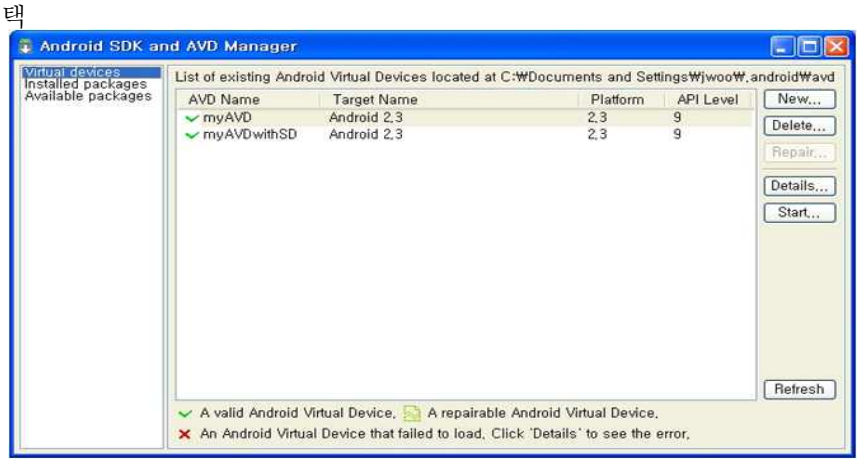


그림 2.30

2.6 안드로이드 SDK 둘러보기

2.6.1 안드로이드 SDK란?

-안드로이드 애플리케이션의 개발, 테스트, 디버그 하는데 필요한 모든 API들과 도구를 포함
-안드로이드 SDK 플러그인을 사용하여 Eclipse IDE에 적용 가능
-안드로이드 SDK 폴더

그림 2.31

-주요 구성 요소

■ 안드로이드 API
- 구글이 네이티브 안드로이드 애플리케이션을 개발하기 위하여 사용한 것과 동일한 라이브러리

■ 개발도구
- 안드로이드 응용 프로그램 개발, 즉 애플리케이션의 컴파일, 디버그, 테스트하는데 필요한 각종 도구
- 이클립스상에서 개발하는 경우 이 도구들을 직접 다루지는 않음

■ 풍부한 문서
- 안드로이드 SDK의 각종 패키지, 클래스에 대한 설명
- 안드로이드 개발을 시작하는 방법과 원리를 설명
- Java의 경우 Java SDK Documentation과 유사

■ 샘플 코드
- 안드로이드 API 기능의 사용법을 나타낸 예제 프로그램 코드
- 이클립스 프로젝트에 추가하여 실행 가능

2.6.2 주요 안드로이드 도구

-emulator.exe

■ Dalvic 가상머신의 구현으로 하드웨어 중립적

■ 안드로이드용 응용 프로그램이 실제 휴대단말에서 동작하는 모습을 PC로 확인

■ 옵션을 포함하여 실행 가능하지만 대부분 이클립스에서 구동

■ 구동하는 것은 리눅스 시스템을 부팅하는 것이기 때문에 장시간 요구. 에뮬레이터를 구동한 후 계속 작업 가능

그림 2.32

-adb.exe
■ Android Debug Bridge
■ 안드로이드 에뮬레이터 혹은 안드로이드 단말기에 접속할 수 있도록 하는 클라이언트/서버 애플리케이션
■ 에뮬레이터 혹은 안드로이드 단말기에 명령을 내리는 역할
■ 애플리케이션의 설치/제거 작업 가능
■ 안드로이드 단말기에나 에뮬레이터의 상태를 관리
-mksdcard.exe
■ 하드디스크의 일부분을 안드로이드 에뮬레이터에서 가상의 SD 카드로 생성
-dx.bat
■ Dalvic VM에 구동할 수 있는 응용 프로그램으로 만들어주는 컴파일러
■ 컴파일이 완료되면 *.dex의 확장자를 가진 파일을 생성

-aapt.exe
■ Android Asset Packaging Tool
■ 배포 가능한 안드로이드 패키지 파일(*.apk)을 생성
-aidl.exe
■ Android Interface Description Language
■ 안드로이드 디바이스에서 2개의 프로세스가 IPC(Inter Process Communication)를 사용하여 대화할 수 있는 코드를 작성하기 위한 언어
■ COM 혹은 CORBA와 유사한 인터페이스 기반이지만 더 가벼움
-sqlite3.exe
■ SQLite 데이터베이스 파일을 제어하는 도구

2.7 에뮬레이터 조작하기

2.7.1 AVD의 실행
-Android AVD and SDK Manager 창의 [Start] 버튼을 클릭

그림 2.33

-혹은 DOS 창에서
emulator -avd <avd_name>

2.7.2 에뮬레이터 키와 호스트 키

표 2.1

에뮬레이터 키	호스트 키
Home	Home
Back	Esc
통화 버튼	F3
통화 종료 버튼	F4
탐색	F5
전원 버튼	F7
소리 증가 버튼	KEYPAD_PLUS, [Ctrl] + [5]
소리 감소 버튼	KEYPAD_MINUS, [Ctrl] + [F6]
카메라 버튼	Ctrl-KEYPAD_5, [Ctrl] + [F3]
레이아웃을 이전 방향 전환	KEYPAD_7, [Ctrl] + [F11]
레이아웃을 다음 방향 전환	KEYPAD_9, [Ctrl] + [F12]
D-패드 좌/우/상/하	KEYPAD_4/6/8/2
D-패드 중앙	KEYPAD_5

제3장 안드로이드 프로그램의첫걸음

3.1 프로젝트 생성과에뮬레이터 구동

3.1.1 프로젝트 생성

-단계 1

그림 3.1

혹은

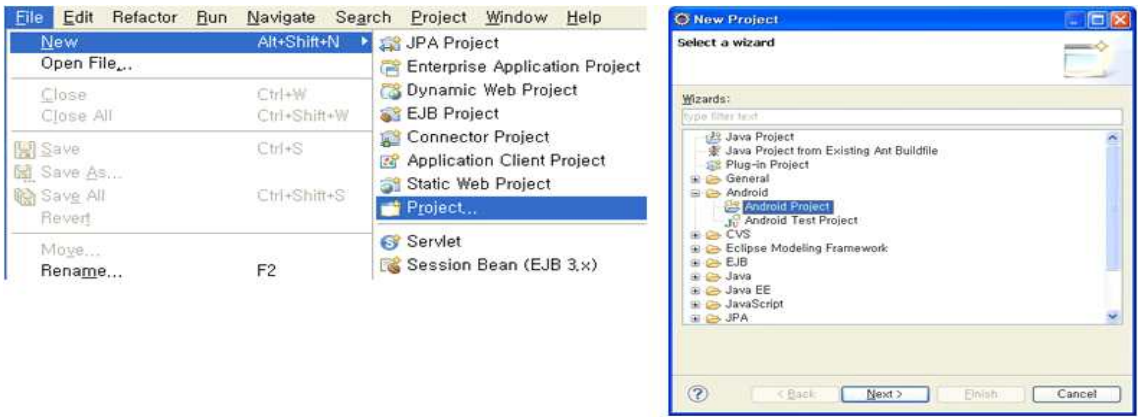


그림 3.2

-단계2

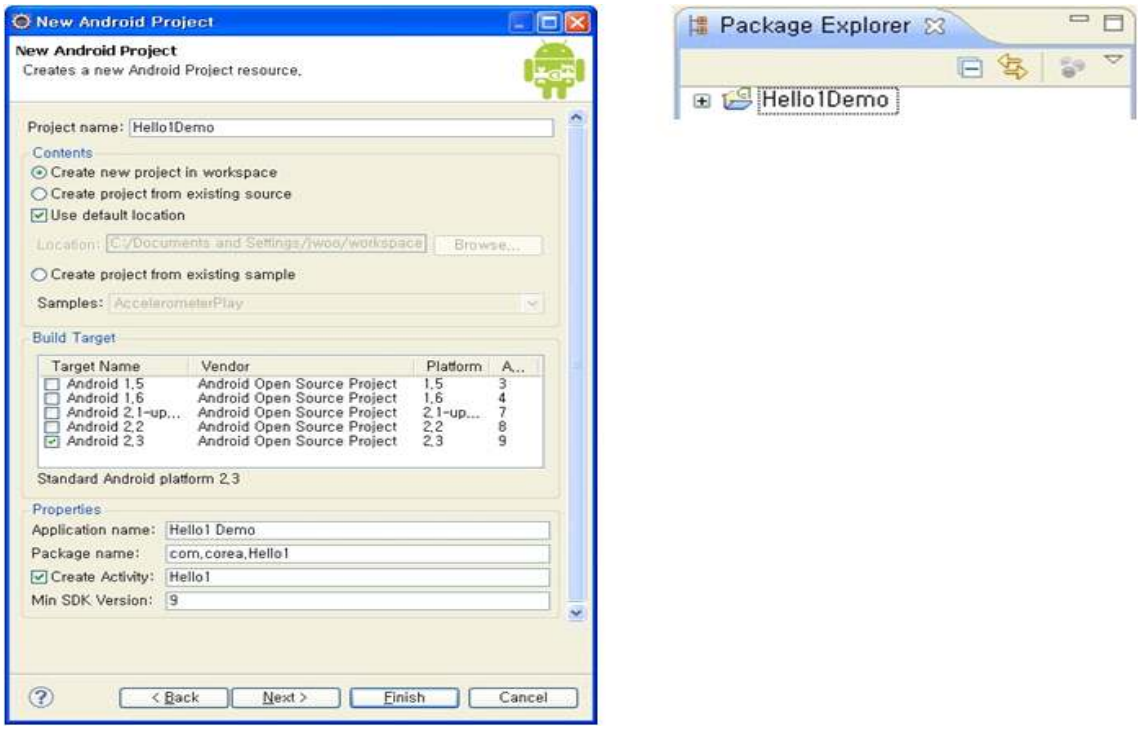


그림 3.3

3.1.2 에뮬레이터 구동

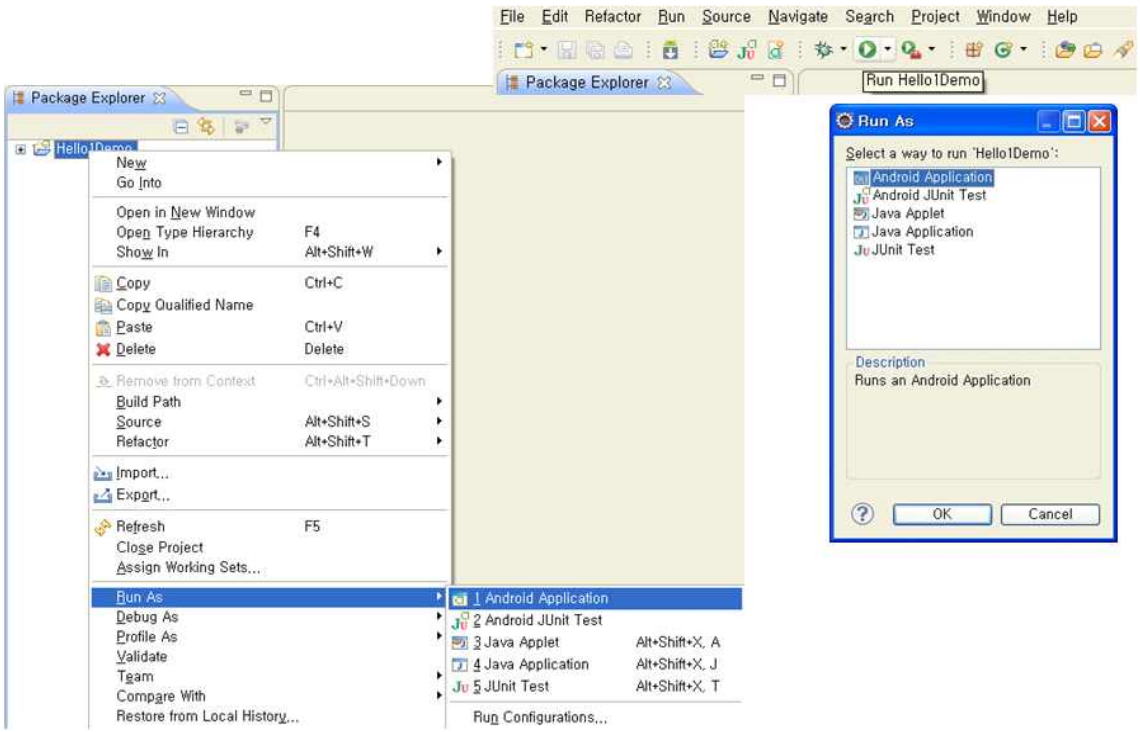


그림 3.4

-부팅 과정

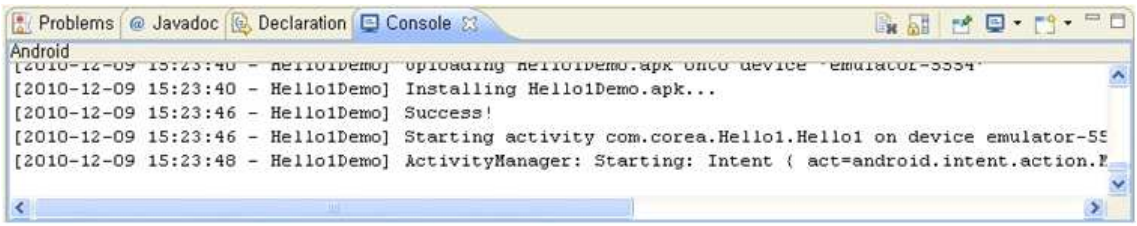


그림 3.5

-결과

그림 3.6

3.1.3 프로젝트 이름, 패키지 이름 …

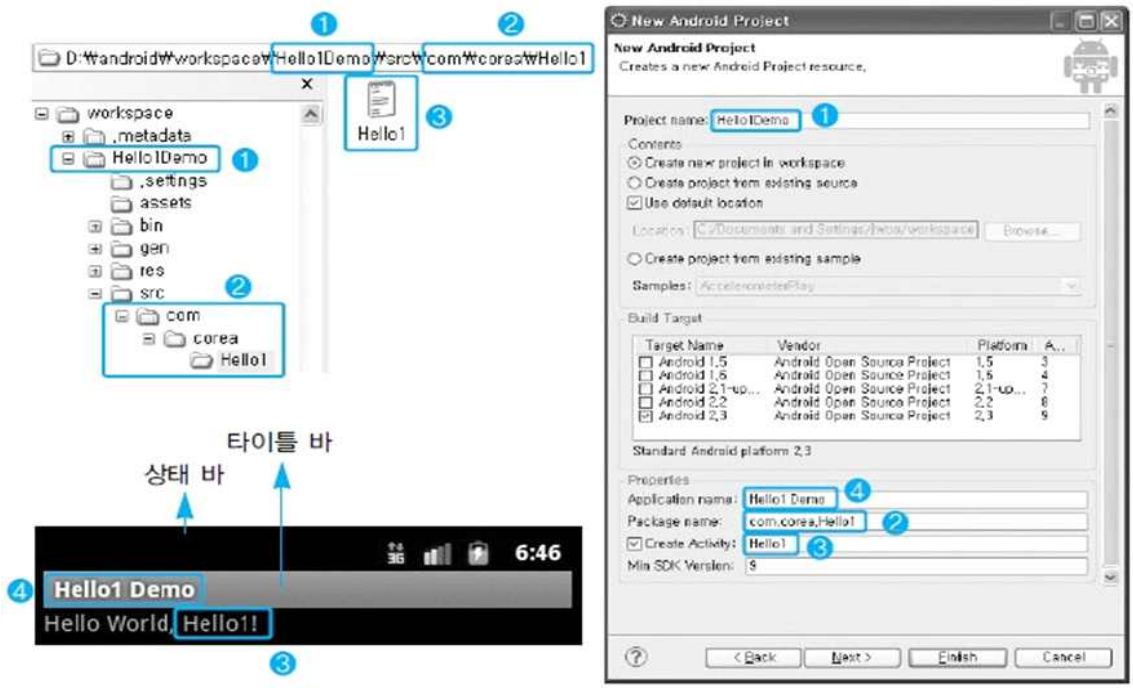


그림 3.7

3.2 프로젝트 파일과 소스 코드 이해

3.2.1 프로젝트 구성 파일

-안드로이드 애플리케이션의 구성

■ 새로운 프로젝트를 생성 다음과 같은 폴더와 파일이 나타남

- 소스 코드를 위한 /src와 /gen
- 리소스 관리를 위한 /drawable, /layout, /values, /assets
  -XML, PNG, JPEG 등의 다양한 종류의 리소스 파일을 지원
- 안드로이드 라이브러리
- Androidmanifest.xml

그림 3.8

-src/ 폴더와 Hell1.java 소스 코드

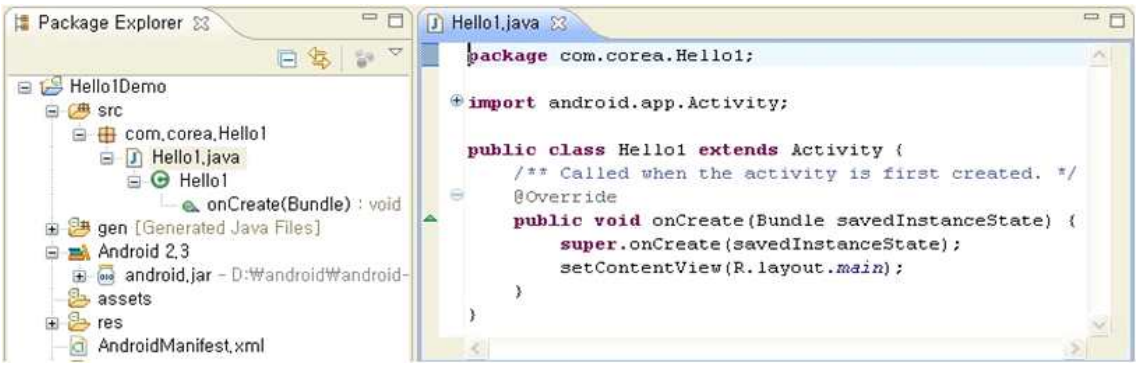


그림 3.9

-gen/ 폴더와 R.java 소스 코드

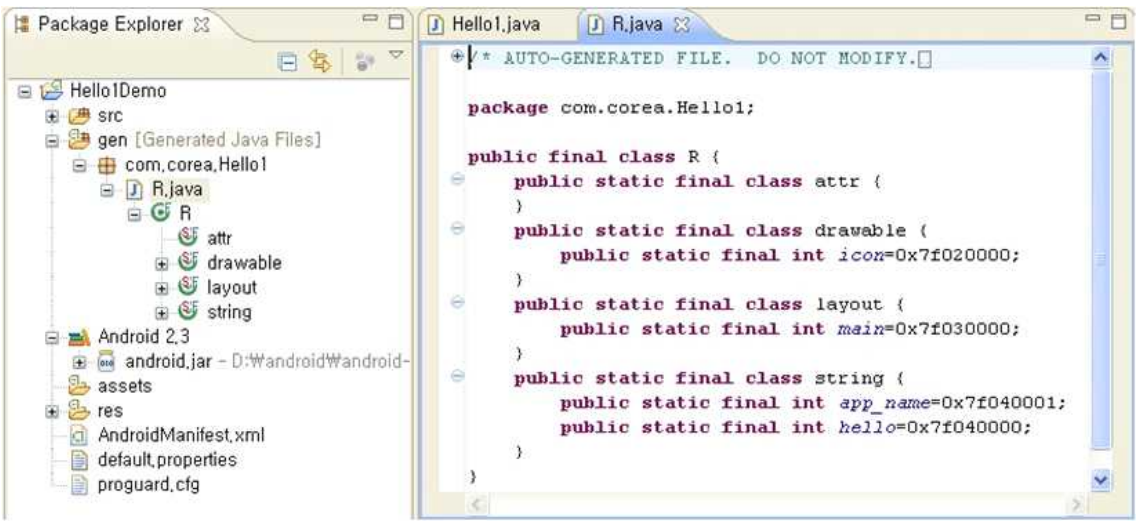


그림 3.10

-리소스

■ res/ 폴더

그림 3.11

-이미지 파일(/drawable)

■ jpg, png 등의 파일 형식을 지원

■ 모든 프로젝트에는 어플리케이션의 아이콘인 icon.png를 포함

■ 리소스를 추가시키게 되면 해당 파일의 이름을 리소스 명으로 등록

■ 파일 이름은 모두 소문자 및 일부 한정된 특수문자([a-z0-9_.])로 구성

■ 해상도(고해상도, 중해상도, 저해상도)에 따라 다른 리소스를 사용

-리소스

■ 레이아웃(/layout)

- 어플리케이션의 화면을 구성하는 레이아웃 파일들을 포함
- 액티비티의 레이아웃인 main.xml 파일을 포함

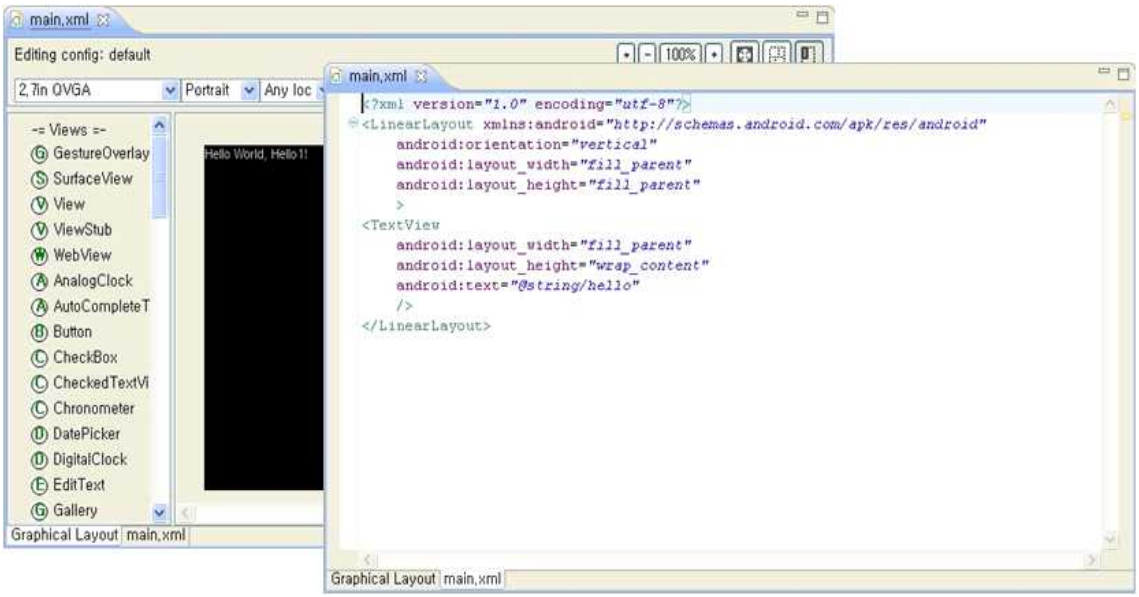


그림 3.12

-리소스

■ 기타 여러 가지 값(/values)

- 문자열, 배열 등 기타 애플리케이션에서 사용하는 여러 가지 값들을 보관
- 모든 프로젝트에는 문자열을 저장하는 strings.xml 파일을 포함

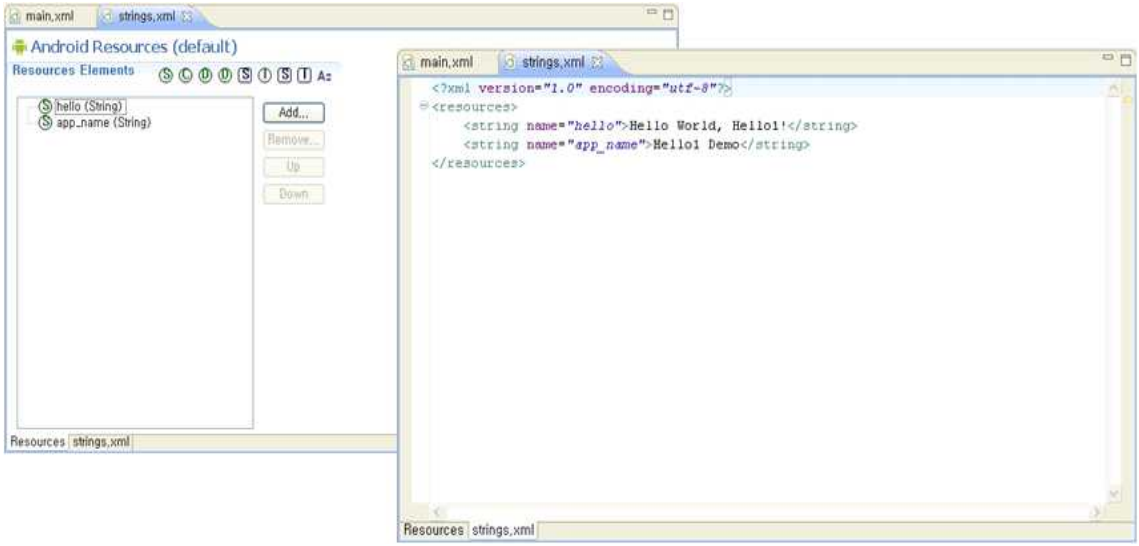


그림 3.13

-매니페스트 파일(AndroidManifest.xml)

■ 애플리케이션의 프로필과 같은 역할

■ 애플리케이션의 이름과 버전, 컴포넌트(액티비티, 서비스 등) 정의, 사용 권한, 사용하는 라이브러리 등 애플리케이션의 뼈대를 구성하는 정보를 포함

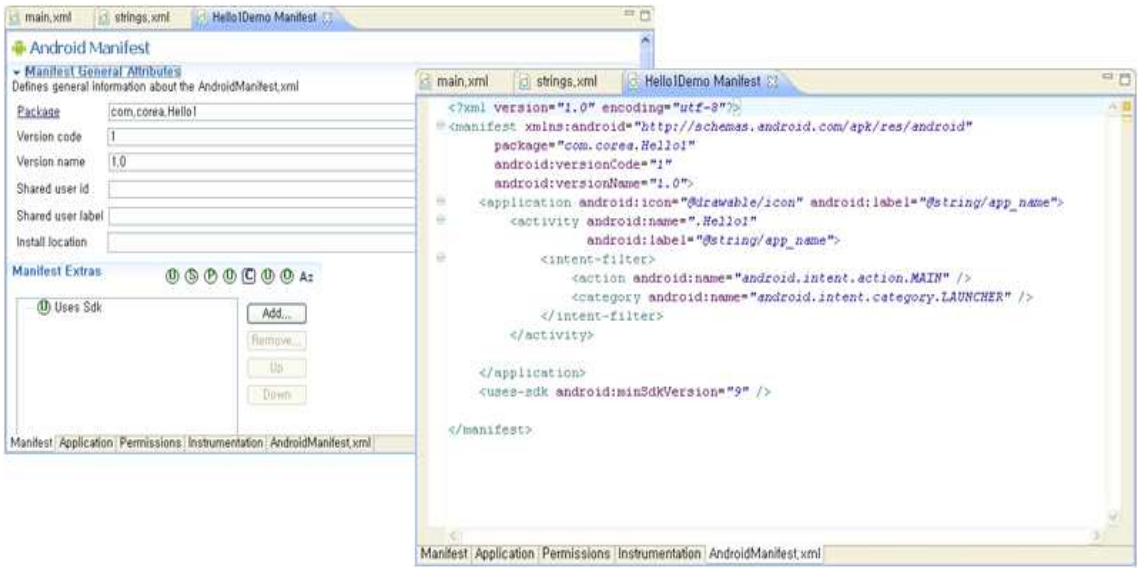


그림 3.14

3.2.2 애플리케이션 빌드

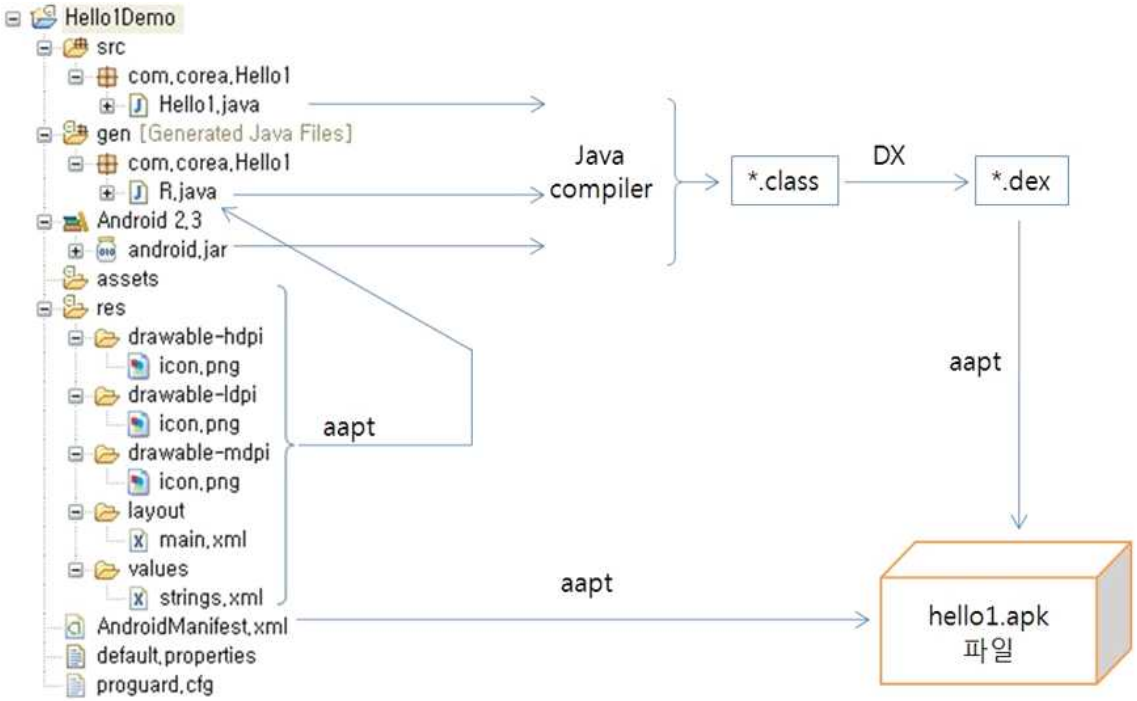


그림 3.15

3.2.3 프로그램 소스 분석

-프로그램 소스

```
Hello1.java
1  package com.corea.Hello1;
2
3  import android.app.Activity;
5
6  public class Hello1 extends Activity {
7      /** Called when the activity is first created. */
8      @Override
9      public void onCreate(Bundle savedInstanceState) {
10         super.onCreate(savedInstanceState);
11         setContentView(R.layout.main);
12     }
13 }
```

```
1  package com.corea.Hello1;
2
3  import android.app.Activity;
4  import android.os.Bundle;
5
6  public class Hello1 extends Activity {
7      /** Called when the activity is first created. */
8      @Override
9      public void onCreate(Bundle savedInstanceState) {
10         super.onCreate(savedInstanceState);
11         setContentView(R.layout.main);
12     }
13 }
```

그림 3.16

```
package com.corea.Hello1;

import android.app.Activity;
import android.os.Bundle;

public class Hello1 extends Activity {
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
    }
}
```

```
#include <stdio.h>

int main()
{
    printf(Hello World, Hello1!₩n");
    return 0;
}
```

그림 3.17

-import android.app.Activity

■ Activity란 응용 프로그램에서 하나의 화면을 구성하는 단위

■ Activity란 사용자 인터페이스 컴포넌트를 화면에 표시하고 시스템이나 사용자의 반응을 처리

■ 애플리케이션 = 다수의 Activity의 집합

-import android.os.Bundle

■ Bundle은 프로그램 실행에 필요한 정보(예를 들면, toString, getString, getShort 등)를 포함하는 클래스

■ C 언어의 stdio.h와 유사?

-onCreate(Bundle savedInstanceState)

■ 액티비티가 실행될 때 호출

■ 액티비티의 UI를 적용하기 위한 setContentView()와 같은 메소드를 호출

■ Bundle 객체는 애플리케이션의 실행/종료에 따른 상태 정보를 저장하기 위함

-setContentView(R.layout.main)

■ R 클래스의 내부 클래스인 layout 클래스의 main 필드를 화면에 출력

■ R.java 파일은 프로젝트 내의 여러 가지 리소스 파일의 주소를 포함

-R.layout.main

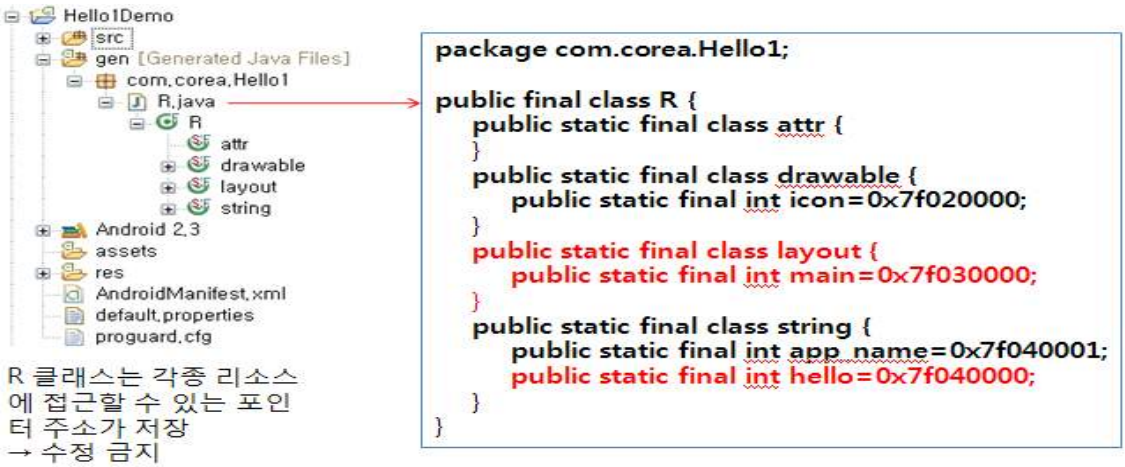


그림 3.18

-레이아웃

■ 뷰(view)는 화면 상에서 UI를 구성하는 기본 빌딩블록. 예를 들면 텍스트, 에디트, 그림, 버튼 등과 같은 기본적인 화면 구성 요소

■ 레이아웃(layout)은 각 뷰를 화면 상에 배치하고 구성하는 일종의 뷰그룹(View Group)

■ 레이아웃은 일반적으로 XML로 구성. 코드로도 구성 가능

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    >
<TextView
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:text="@string/hello"
    />
</LinearLayout>
```

-문자열 hello

◉ **문자열 hello**


```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string name="hello">Hello  World, Hello1!</string>
    <string name="app_name">Hello1  Demo</string>
</resources>
```

그림 3.19

-AndroidMenifest.xml

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
      package="com.corea.Android"
      android:versionCode="1"
      android:versionName="1.0">
    <application android:icon="@drawable/icon"
        android:label="@string/app_name">
        <activity android:name=".Android"
                  android:label="@string/app_name">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>

    </application>
    <uses-sdk android:minSdkVersion="7" />

</manifest>
```

-AndroidMenifest.xml

- 매니페스트 파일은 안드로이드가 애플리케이션을 구동할 때 필요한 구성 정보 – 주 액티비티, 라이브러리 및 모듈, 퍼미션 등 – 를 기술
- 기호 @는 리소스 파일에서 접근할 수 있는 정보를 의미
- Android 애플리케이션이 하나의 액티비티로 구성
- 인텐트 필터는 액티비티, 서비스, 브로드캐스트 리시버가 반응할 수 있는 인텐트의 형식을 기술
- android.intent.action.MAIN은 애플리케이션의 진입점(entry point)을 의미
- android.intent.category.LAUNCHER는 액티비티를 런처(launcher)에 노출
- android.intent.action.MAIN과 android.intent.category.LAUNCHER의 조합은 해당 액티비티를 애플리케이션 기동 시 최초에 수행될 액티비티로 지정

-문자열 출력 과정

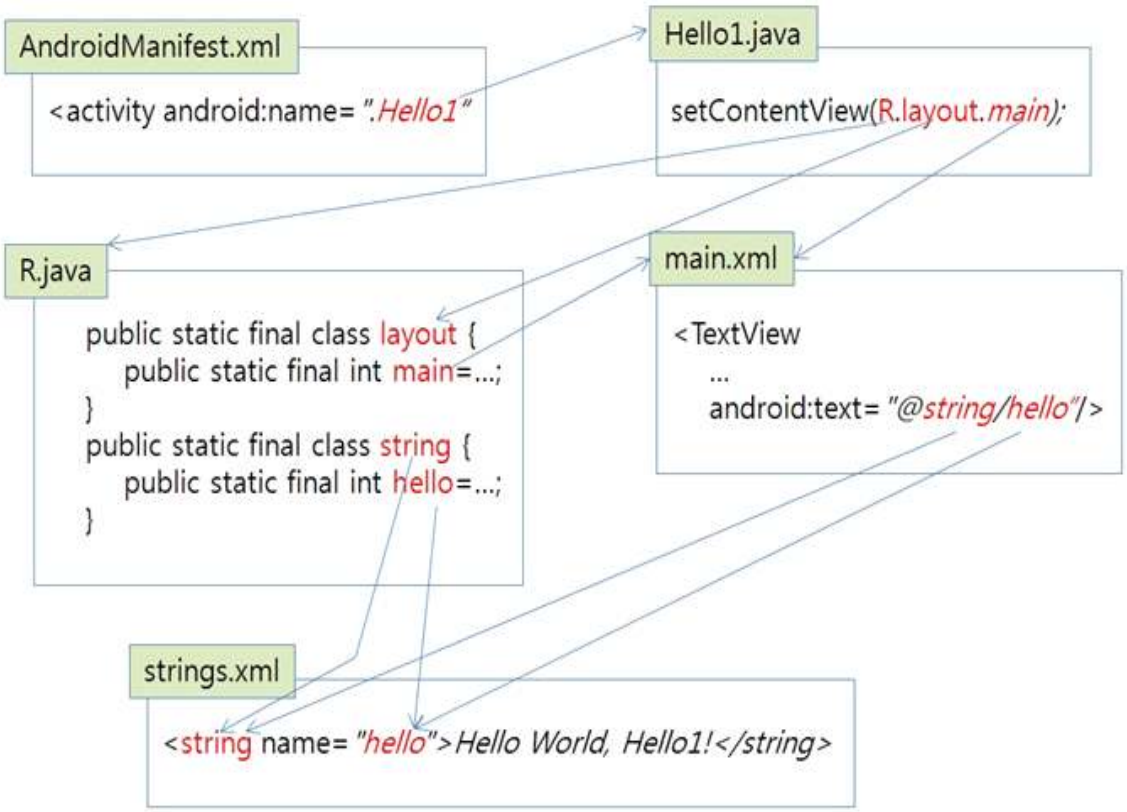


그림 3.20

3.3 코드로 문자열 표시하기

3.3.1 프로그램 소스 수정

-실행

- 첫 번째 프로젝트와 동일한 방법으로 Hello2Demo 프로젝트를 생성
- 동일한 결과
- 그러나 리소스 클래스인 R.java를 사용하지 않음
- 프로그램 소스와 사용자 인터페이스를 분리하지 않음

-앞 프로젝트의 소스 코드를 수정

Hello1.java

```
1  package com.corea.Hello1;
2
3  import android.app.Activity;
5
6  public class Hello1 extends Activity {
7      /** Called when the activity is first created. */
8      @Override
9      public void onCreate(Bundle savedInstanceState) {
10         super.onCreate(savedInstanceState);
11         setContentView(R.layout.main);
12     }
13 }
```

*Hello2.java

```
1  package com.corea.Hello2;
2
3  import android.app.Activity;
5
6  public class Hello2 extends Activity {
7      /** Called when the activity is first created. */
8      @Override
9      public void onCreate(Bundle savedInstanceState) {
10         super.onCreate(savedInstanceState);
11         TextView hello = new TextView(this);
12         hello.setText("Hello World, Hello2!");
13         setContentView(hello);
14     }
15 }
```

그림 3.21

-자동으로 필요한 import 문장 추가
-ctrl + shift + o

*Hello2.java

```
1  package com.corea.Hello2;
2
3  import android.app.Activity;
4  import android.os.Bundle;
5  import android.widget.TextView;
6
7  public class Hello2 extends Activity {
8      /** Called when the activity is first created. */
9      @Override
10     public void onCreate(Bundle savedInstanceState) {
11         super.onCreate(savedInstanceState);
12         TextView hello = new TextView(this);
13         hello.setText("Hello World, Hello2!");
14         setContentView(hello);
15     }
16 }
```

그림 3.22

-필요한 메소드의 선택

그림 3.23

-수정된 소스 코드

```
package com.corea.Android;

import android.app.Activity;
import android.os.Bundle;
import android.widget.TextView;

public class Android extends Activity {
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        TextView hello = new TextView(this);
        hello.setText("Hello World, Android!");
        setContentView(hello);
    }
}
```

-TextView

■ 화면에 문자열을 표시하는 역할

■ 생성자

```
Public TextView(Context context)
Public TextView(Context context, AttributeSet attrs)
Public TextView(Context context, AttributeSet attrs, int defStyle)
```

■ TextView(this)에서 this는 Activity

■ Activity 클래스의 상속도

```
Java.lang.Object
    android.content.Context
        android.content.ContextWrapper
            android.view.ContextThemeWrapper
                android.app.Activity
```

-setText()

■ 형식

```
public void setText(CharSequence text)
public void setText(int resid)
```

-setContentView()

■ 레이아웃 혹은 뷰(view)를 activity의 화면으로 표시하는 역할

■ 형식

```
public void setContentView(int layoutResID)
public void setContentView(View view)
```

■ hello는 TextView

■ TextView는 View의 서브클래스

3.4 문자열 출력 프로그램 응용

3.4.1 다양한 색상과 크기 문자열

-레이아웃 소스 변경

■ TextView 필드를 다음과 같이 수정한다면?

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:background="#ffffffff">
<TextView
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:text="빨강"
    android:background="#ffff0000" />
<TextView
    android:layout_width="wrap_content"
    android:layout_height="100dp"
    android:text="초록"
    android:textColor="#ff000000"
    android:textSize="50sp"
    android:background="#ff00ff00" />
</LinearLayout>
```

그림 3.24

-TextView 필드의 속성

■ background

■ textSize

■ textColor

■ textStyle

■ gravity: 뷰 안에서의 텍스트 위치

- top, bottom, left, right

- center, center_vertical, center_horizontal
- fill, fill_vertivcal, fill_horizontal

■ visibility
- visible
- invisible(안 보이나 공간 차지), gone(안보이고 공간 차지하지 않음)

-색상

■ 색상 값은 항상 #로 시작

■ 비트 수와 알파(투명도) 여부에 따라 4가지 형태
- #RGB
- #ARGB
- #RRGGBB
- #AARRGGBB
- 여기서 알파 값은 클수록 불투명

-크기

■ 픽셀: px

■ 인치: in

■ 밀리미터: mm

■ 포인트: pt 1dp는 160dpi화면에서 1px에 대응

■ 밀도 독립 픽셀: dp(density-independent pixel)

■ 축척 독립 픽셀: sp(scale-independent pixel)

3.4.2 이미지 출력

-이미지를 res₩drawable 아래에 복사

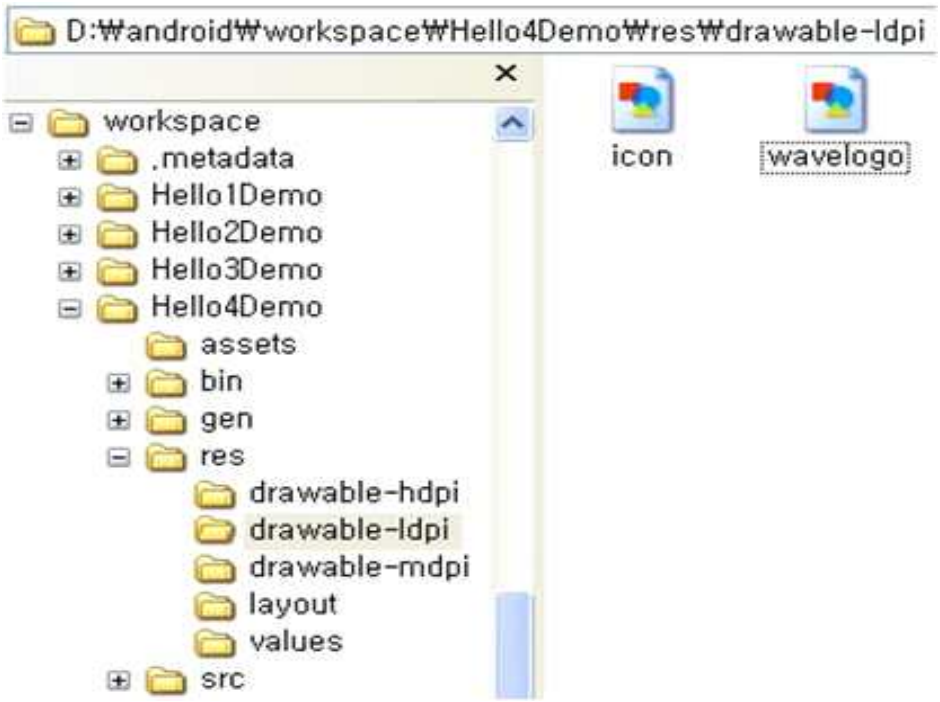


그림 3.25

-레이아웃 소스 변경

```
<?xml version="1.0" encoding="utf-8"?>
<ImageView xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/icon"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:adjustViewBounds="true"
    android:src="@drawable/wavelogo"/>
```

혹은

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent">
<ImageView
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:src="@drawable/wavelogo"/>
</LinearLayout>
```

-해상도에 따른 이미지

■ drawable-hdpi, drawable-ldpi, drawable-mdpi

■ 어떤 폴더에도 무관

-지원하는 이미지 형식

■ png

■ 9-patch

■ jpg

■ gif

■ …

3.4.3 레이아웃 내부에 레이아웃

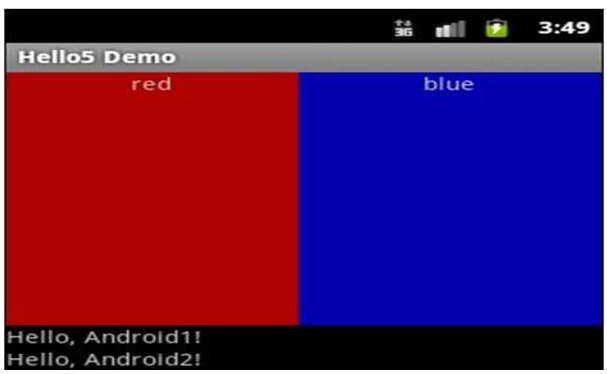


그림 3.26

제4장 액티비티와 리소스

4.1 액티비티 이해

4.1.1 안드로이드 애플리케이션

-자신의 리눅스 프로세스 내에서 실행

■각 프로세스는 자신의 자바 가상머신을 가짐 → 각 프로세스는 다른 프로세스와 격리되어 실행

■진입점(entry point)이 없고 인텐트(intent)에 의하여 컴포넌트 활성화

-고유한 리눅스 ID가 부여

■기본적으로 애플리케이션을 구성하는 파일들은 해당 사용자와 해당 애플리케이션에게만 접근 허용

■시스템 자원을 절약하기 위하여 2개의 애플리케이션에 대하여 동일한 사용자 ID 공유 가능 → 동일한 리눅스 프로세스 안에서 실행되며 동일한 가상머신을 공유

-느슨하게 결합된 컴포넌트로 구성

■애플리케이션 컴포넌트

- 액티비티(activity)
- 서비스(services)
- 컨텐트 공급자(content provider)
- 브로드캐스트 리시버(broadcast receiver)

■안드로이드 매니페스트 파일

- 각 컴포넌트의 상호 작용하는 방법을 기술
- 애플리케이션 구성 요소 선언 → APK 외부에서 패키지 구성 요소를 알 수 있도록 함
- 활용하고자 하는 라이브러리 지정
- 애플리케이션에게 필요한/허용한 접근 권한(permission) 등록
- 인텐트 필터(intent-filter)를 통한 애플리케이션의 주 진입점을 액티비티에 지정

-액티비티와 태스크

■액티비티는 같은 애플리케이션 내에 존재하는 액티비티 뿐만 아니라 다른 애플리케이션 내에 존재하는 액티비티까지 호출 가능

■한 애플리케이션에서 다른 애플리케이션의 컴포넌트를 거의 자유자재로 사용 가능 → 파일의 구성으로만 보면 애플리케이션의 경계가 뚜렷하지만 실제로 애플리케이션의 실행 면에서는 애플리케이션 사이에는 경계가 없음

■태스크(Task)는 사용자가 실질적으로 “하나의 애플리케이션처럼” 느끼는 액티비티들의 집합, 즉 액티비티 스택(Activity Stack)

-태스크와 프로세스

■태스크

- 연관된 액티비티의 집합
- 다수의 프로세스와 APK에 걸쳐 존재 가능
- 다른 APK의 액티비티 호출 가능

■프로세스

- 커널 프로세스

• 기본적으로 APK는 하나의 프로세스에서 동작
• 하나의 APK에서 다수의 프로세스 매핑 가능

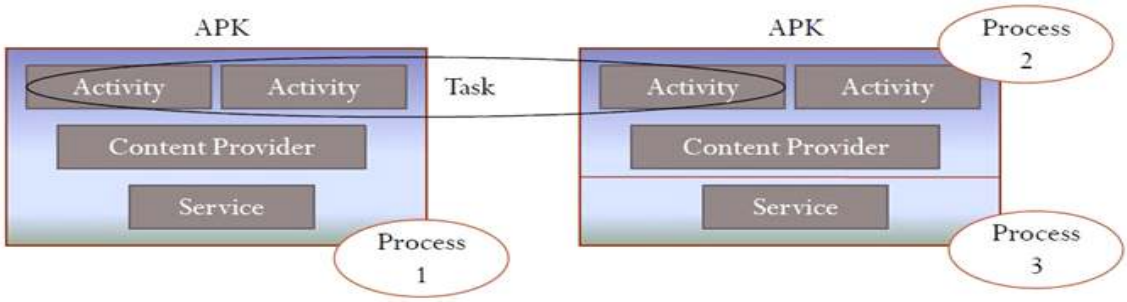


그림 4.1

-액티비티 스택

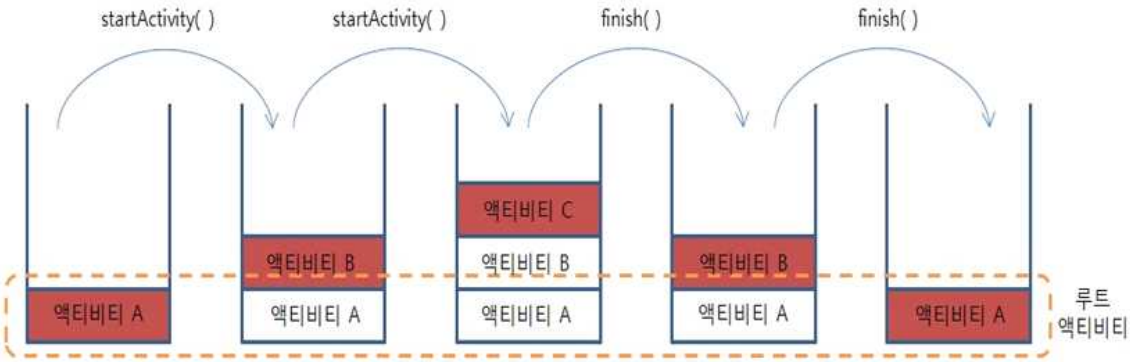


그림 4.2

-프로세스 생명주기

메모리가 부족할 경우 중요하지 않은 순서대로 메모리에서 프로세스를 제거

■ 전경(foreground) 프로세스: 현재 포커스를 가지고 있는 프로세스. 가장 마지막에 제거
■ 가시(visible) 프로세스: 포커스는 없지만 화면에 보이는 프로세스
■ 서비스(service) 프로세스
■ 배경(background) 프로세스: 화면에 보이지 않는 프로세스
■ 공백(empty) 프로세스: 액티비티를 가지고 있지 않은 프로세스. 메모리가 부족할 경우 바로 제거

4.1.2 액티비티 상태와 콜백 메소드

-개요

■ 안드로이드는 모바일 기기에서의 구동을 목적으로 하기 때문에 데스크탑 애플리케이션에 비하여 더욱 효율적 메모리 관리가 필요
■ 애플리케이션 컴포넌트의 중요 요소인 액티비티도 효율적인 메모리 관리를 위하여 액티비티 생성 및 소멸 과정인 생명주기가 있음
■ 액티비티가 중지 혹은 정지 상태일 경우 해당 프로세스가 메모리에서 제거될 수 있음. 사용자에게 다시 보여지게 될 때 이전 상태로 복구

■ 액티비티는 액티비티 스택(activity stack)에 의하여 관리

액티비티는 사용자와 상호 작용하는 단위이며 일반적으로 전체 화면을 차지

-액티비티의 3가지 상태

■ 활성(active) 혹은 실행(running) 상태
- 전경 화면에 있을 경우
- 해당 액티비티가 사용자의 동작에 대한 포커스를 가짐
- 사용자와 상호 작용 가능

■ 중지(paused) 상태
- 포커스를 보유하지 않았지만 사용자에게 일부 보임
- 메모리가 극도로 부족할 경우 시스템에 의하여 강제 종료
- 사용자와 상호 작용 불가

■ 정지(stopped) 상태
- 사용자에게 전혀 보이지 않지만 여전히 모든 상태와 멤버 정보는 유지
- 다른 컴포넌트가 메모리를 요청하면 시스템에 의하여 강제 종료

-액티비티를 위한 메소드

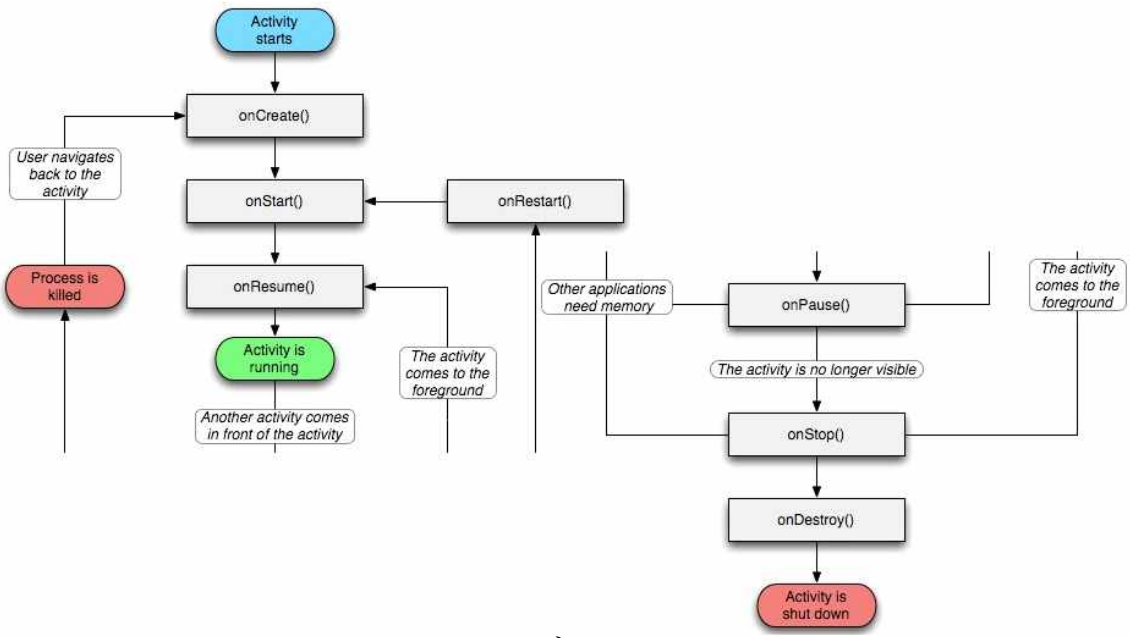


그림 4.3

-액티비티를 위한 콜백 메소드

■ onCreate()
- 액티비티가 생성될 때 처음으로 호출되는 함수
- 전역 상태의 모든 리소스를 초기화. 예를 들면 layout과 data binding 등
- 액티비티는 기본적으로 버튼, 리스트, 체크박스, 입력 표시줄 등과 같은 위젯들이 배치되어 있는 레이아웃을 구성해야 하고, 그러한 웨젯들이 사용자와 상호작용을 할 수 있도록 해주는 코드를 포함

■ onStart()

• 액티비티가 최기화 과정을 마친 후 사용자에게 보여줄 준비가 되었을 때 호출

■ onResume()
• 액티비티가 사용자에게 보여지고 사용자의 입력을 처리할 수 있음
• 액티비티 스택의 최상위에 위치

■ onPause()
• 절전 상태 혹은 새로운 액티비티가 시작될 경우
• 포커스 상실
• 재개(resume)되기 전 데이터 저장, 애니메이션 중지, 프로세서를 소비하는 작업 중단

■ onStop()
• 더 이상 액티비티가 사용자에게 보이지 않음
• 더 이상 액티비티 스택의 최상위에 위치하지 않음

■ onDestory()
• 존재하는 모든 리소스를 해제
• 시스템 내에 액티비티가 존재하지 않게 됨

-액티비티 상태 저장하기

■ 일반적으로 정지된 액티비티는 사용자가 다시 사용할 것을 대비하여 메모리에 상주. 그러다가 메모리가 부족하게 되면 강제로 종료

■ 액티비티를 다시 호출하면
• 강제 종료된 상태: 다시 액티비티를 생성한 후 액티비티를 실행
• 강제 종료되지 않은 상태: 액티비티를 다시 만들 필요 없이 다시 화면에 표시하면 되므로, 액티비티를 다시 시작

■ 액티비티가 중지되는 것은
• 화면에 표시되지 않음
• 사용자와 상호작용 하지 않음
• 현재 액티비티의 상태를 Bundle 형태로 저장 → 사용자가 텍스트 등을 입력하다가 액티비티가 종료될 경우 작업 내용이 날아가는 것을 방지하기 위함
• 새로 액티비티가 시작할 경우 Bundle 객체에는 null
• 활성 상태로 변할 때 저장된 UI 상태를 복원

■ onSaveInstanceState()
• onPause() 혹은 onStop() 이후 메모리가 부족할 경우 프로세스가 메모리에서 제거될 수 있음
• 메모리에서 제거되기 전에 액티비티 상태를 저장
• 매개 변수로 액티비티의 동적 상태를 기록할 수 있는 번들(Bundle) 오브젝트

■ onRestoreInstanceState()
• onCreate() 혹은 onStart() 이후 저장된 액티비티 상태를 복원

■ onSaveInstanceState()와 onRestoreInstanceState()는 생명주기 메소드가 아니기 때문에 항상 호출되지 않음. 개발자가 해당 코드의 메소드를 오버라이드하여 구현

■ 매개변수로 사용되는 savedInstanceState는 저장된 인스턴스의 상태, 즉 액티비티의 UI 상태를 의미

4.2 리소스 이해

4.2.1 리소스 생성

-리소스란?

■ 애플리케이션 = 기능 + 리소스
■ 기능은 애플리케이션 실행에 관계되는 모든 알고리즘을 포함하는 코드
■ 리소스는 애플리케이션이 사용하는 자산
  • 텍스트 문자열, 이미지, 아이콘, 오디오, 동영상 등
  • 레이아웃
■ 리소스를 소스 코드와 분리하면?
  • 유지 보수 용이
  • 언어와 문화권에 맞는 애플리케이션의 현지화(localization) 가능

-리소스의 저장
■ 안드로이드 프로젝트에서 리소스는 Java 소스 코드와는 별도로 외부 파일로 저장
■ 일반적인 리소스 파일은 대부분 XML 파일로 저장
■ 이미지와 같은 원본 자료 파일은 그 자체로 저장
■ 모든 리소스는 res 디렉토리의 하위 디렉토리에 저장되며, 하위 디렉토리 이름은 소문자 + 숫자+ 밑줄로만 구성
■ 그래픽 및 애니메이션 파일 등 일부 리소스 파일은 파일 이름과 동일한 이름의 변수로 참조되기 때문에 Java 식별자 형태의 파일 이름으로 명명
■ aapt(Android Asset Packaging Tool)가 리소스를 모두 파악하여 자원을 접근하기 위한 변수의 정의를 담은 gen/R.java 파일을 생성

-리소스 형식과 파일 이름

표 4.1

리소스 형식	디렉토리	권장 파일 이름	엘리먼트 이름
문자열	values	strings.xml	<string>
문자열 배열		arrays.xml	<string-array>
색상 값		colors.xml	<color>
크기		dimens.xml	<dimen>
단순 표시		drawables.xml	<drawable>
스타일 및 테마		styles.xml, themes.xml	<style>
그래픽	drawable	drawables.xml	
애니메이션	anim		<set>, <alpha>, <scale> 등
메뉴	menu		<menu>
XML	xml		
원본	raw		
레이아웃	layout		

4.2.2 리소스 접근

-코드에서 문자열 참조

■ R.java 파일에 정의된 R 클래스와 하위 클래스를 이용

■ 코드에서 리소스를 접근하려면 R 클래스와 하위 클래스의 멤버 변수를 통하여 접근.

■ 예를 들어 hello라는 문자열을 접근하려면

```
String myString = getResources().getString(R.string.hello);
```

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string name="hello">Hello World, Android!</string>
    <string name="app_name">Android</string>
</resources>
```

-코드에서 문자열 배열 참조

■ 문자열 배열 fruits의 접근

```
String[] fruits = getResources().getStringArray(R.array.fruits);
```

■ 문자열 배열 리소스 파일(~/res/values/arrays.xml)

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string-array name="fruits">
        <item>apple</item>
        <item>banana</item>
    </string-array>
    <string-array name="animal">
        <item>tiger</item>
        <item>lion</item>
    </string-array>
</resources>
```

-코드에서 색상 참조

■ 문자열 배열 fruits의 접근

```
int myColor = getResources().getColor(R.color.textColor);
```

■ 색상 리소스 파일(~/res/values/colors.xml)

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <color name="backgroundColor">#006600</color>
    <color name="textColor">#ffeecc</color>
</resources>
```

-코드에서 크기 참조

■ 텍스트 크기를 위한 textPointSize의 접근

```
float myTextSize = getResources().getDimension(R.dimen.textPointSize);
```

■ 크기 리소스 파일(~/res/values/dimens.xml)

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <dimen name="smallSize">6pt</dimen>
    <dimen name="textPointSize">11pt</dimen>
    <dimen name="largeSize">20pt</dimen>
</resources>
```

-코드에서 레이아웃 참조

■ 레이아웃 내부에 정의된 ImageView01의 접근

```
ImageView iv = (ImageView)findViewById(R.id.ImageView01);
```

■ 레이아웃 파일(~/res/layout/main.xml)

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:background="#ffffffff">
<ImageView
    android:id="@+id/ImageView01"
    android:layout_width="fill_parent"
    ...
...
</LibearLayout>
```

-코드에서 이미지 참조

■ res/drawable 디렉토리에 추가한 flag.png 파일의 접근

```
ImageView iv = (ImageView)findViewById(R.id.ImageView01);
iv.setImageResource(R.drawable.flag)
```

■ 레이아웃 파일(res/layout/main.xml)

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:background="#ffffffff">
<ImageView
    android:id="@+id/ImageView01"
    android:layout_width="fill_parent"
    ...
```

```
…
</LibearLayout>
```

-리소스에서 리소스 참조

■ 참조 방법

```
@[패키지이름:]리소스형식/리소스이름
```

-예제: 애플리케이션 이름을 hello 문자열과 동일하게 하려면 strings.xml을 수정

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string name="hello">Hello World, ColorSize!</string>
    <string name="app_name">@string/hello</string>
</resources>
```

-시스템 리소스 참조

■ android.R의 하위 클래스에는 다음과 같은 시스템 리소스 보유

- 표준 시스템 색상
- 시스템 스타일과 테마
- 애플리케이션 프로그램 섬네일(thumbnail) 이미지와 아이콘
- 오류 문자열과 표준 버튼 텍스트
- 페이드인/아웃을 위한 애니메이션 시퀀스
- …

■ 참조 방법

- 코드에서 참조: R 대신 android.R
- 리소스에서 참조: 패키지 이름으로 android 사용

-선택적 리소스

■ 애플리케이션의 현지화 가능

■ 선택적 리소스를 위한 수식어

표 4.2

수식어	내용
언어	kr(한글), en(영어), fr(불어) 등
화면 크기	small, normal, large
화면 방향	port(세로), land(가로)
화면 픽셀 밀도	ldpi(120dpi), mdpi(160dpi), hdpi(240dpi)

■ 리소스와 참조

- 리소스: MyApp/res/drawable-mdpi/myImage.png
- 코드: R.drawable.myImage
- XML: @drawable/myImage

4.3 이벤트와 토스트

4.3.1 이벤트

-이벤트를 준비하기 위한 2가지 방법

■ View 클래스에 이미 존재하는 Interface를 이용하여 Listener를 만든 후 View에 연결

- View.OnClickListener
- View.OnTouchListener
- View.OnKeyListener
- 등등

■ 클래스를 만들 때 이미 정의되어있는 Listener들을 수정(override)하여 사용

-방법 1

```
public class ExampleActivity extends Activity {
    protected void onCreate(Bundle savedValues) {
        Button button1 = (Button)findViewById(R.id.btn1);
        Button button2 = (Button)findViewById(R.id.btn2);
        button1.setOnClickListener(buttonListener);
        ~~~ 중략 ~~~
    }

    private OnClickListener buttonListener = new OnClickListener() {
        public void onClick(View v) {
            // 클릭이 발생하면 할 일을 추가
        }
    };
    ~~~ 중략 ~~~
}
```

-방법 2

```
public class ExampleActivity extends Activity implements OnClickListener {
    protected void onCreate(Bundle savedValues) {
        Button button1 = (Button)findViewById(R.id.btn1);
        Button button2 = (Button)findViewById(R.id.btn2);
        button1.setOnClickListener(this);
        ~~~ 중략 ~~~
    }
    public void onClick(View v) {
        // 클릭이 발생하면 할 일을 추가
    }
    ~~~ 중략 ~~~
}
```

4.3.2 토스트

-의미

■ 설정 변경 혹은 이벤트 등이 발생했을 때 유용

■ 활성 액티비티의 포커스를 뺏지 않고, 잠시 표시되었다가 사라짐

■ 다른 방법인 Notification에 비하여 매우 간단

-사용 방법

■ makeText() 메소드로 인스턴스를 생성 후 show() 메소드를 사용하여 화면에 표시

■ API

- public static Toast makeText (Context context, int resId, int duration)
- public static Toast makeText (Context context, CharSequence text, int duration)

■ 토스트를 표시할 시간

- Toast.LENGTH_SHORT, Toast.LENGTH_LONG

4.3.3 리소스 응용

• 이미지 출력

그림 4.4

• 버튼 + 이벤트

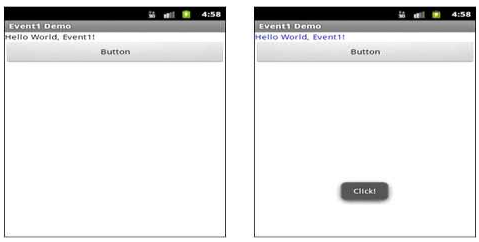


그림 4.5

• 설정에 의한 다국어 지원

그림 4.6

4.4 애플리케이션 디버깅

4.4.1 개요

-오류의 종류

■ 문법 오류(syntax error)

■ 내용 오류(semantic error)

■ 런타임 오류(runtime error)

■ 예제: n2에는 n2, n3에는 n3, m에는 n / d를 저장

```
int n, n2, n3, d;

n2 = n * n:                    // syntax error
n3 = n2 * n2;                  // semantic error
m  = n / d;                    // if d == 0, runtime error
```

-디버깅 도구

■ 아무리 훌륭한 프로그래머도 오류에서 자유롭지 못함

■ 이클립스 기반 안드로이드 개발 환경을 위한 디버깅 도구

- 이클립스 디버거
- 로그캣
- 안드로이드 디버그 브릿지
- DDMS
- 트레이스 뷰

4.4.2 DDMS

-개요

■ Dalvic Debug Monitoring Service

■ ADT 플러그인에서 DDMS 퍼스펙티브를 지원해주므로 이클립스 내에서 DDMS를 바로 사용 가능

■ 역할

- 프로세스 관리
- 에뮬레이터 제어
- 로그 관리
- 파일 관리
- 화면 캡쳐

-DDMS 퍼스펙티브 추가

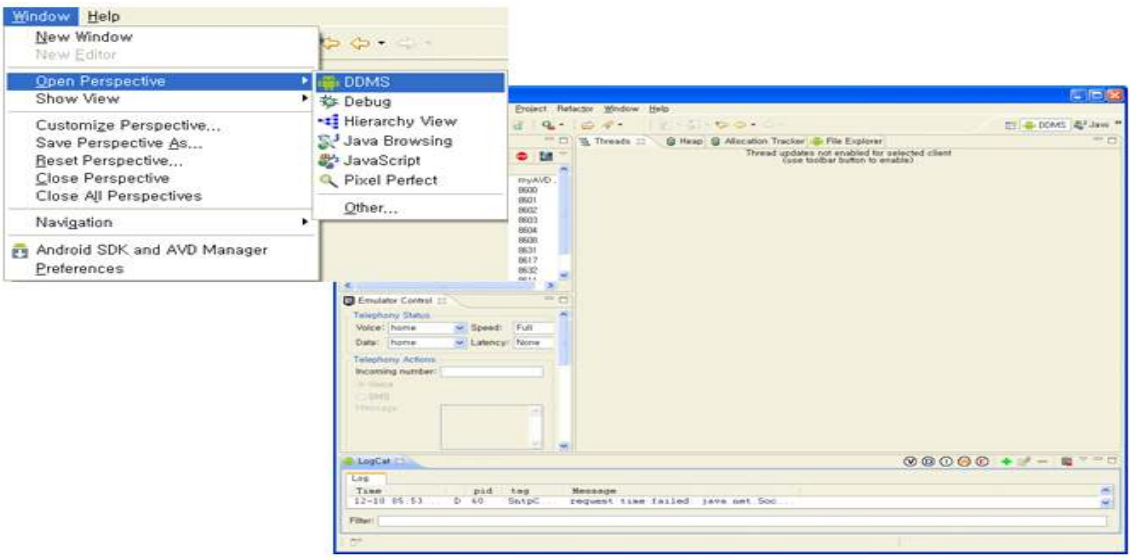


그림 4.7

-로그캣이란

■ 안드로이드가 제공하는 디버깅을 위한 범용 로그 패키지

■ DDMS 퍼스펙티브를 변환하면 하단부에 로그캣 창이 나타남

그림 4.8

■ 애플리케이션을 실행하면 로그캣 창에 부팅을 포함한 모든 실행 과정에 대한 로그 메시지를 출력

-로그캣 열기

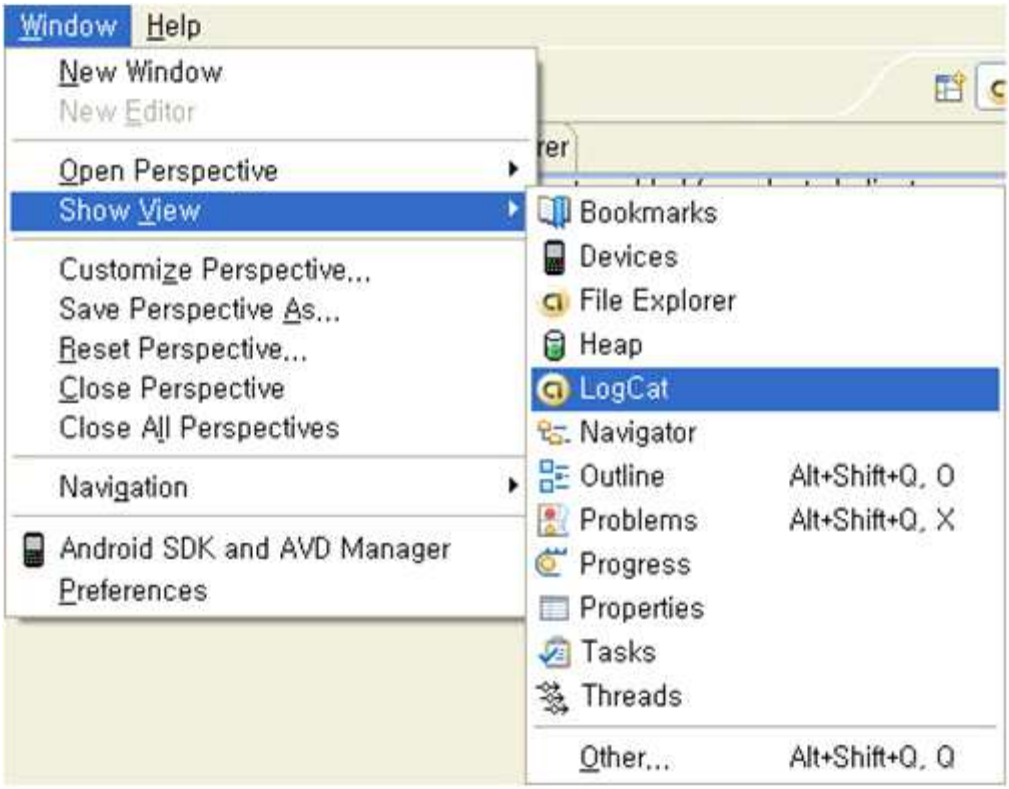


그림 4.9

-로그캣 이용

■ 수준별 로그 메시지를 출력하려면 툴바에 있는 V, D, I, W, E를 사용

• IV(Verbose): 모든 메시지 보기

- I D(Debug): Debug, Information, Warning, Error 수준의 메시지만 보기
- I(Information): Information, Warning, Error 수준의 메시지만 보기
- W(Warning): Warning, Error 수준의 메시지만 보기
- E(Error): Error 수준의 메시지만 보기

■ 의미 있는 로그 메시지를 출력하려면 하단부의 Filter 필드를 사용하여 원하는 문자만 포함하는 로그 메시지만 걸러내어 출력

-로그 메시지의 내용

- 시간(time)
- 우선 순위(priority)
- 프로세스 식별자(process identifier)
- 태그(tag)
- 로그 메시지(log message)

-로그캣에 직접 쓰기

■ 로그캣에 임의의 메시지를 출력하려면 안드로이드가 제공하는 다음 메소드를 사용

```
log.x(String tag, String message, [Throwable exception])
```

■ 여기서 x는 수준별 로그 메시지 출력을 위한 툴바에 있는 V, D, I, W, E 중의 하나

제5장 위젯과 레이아웃

5.1 뷰

5.1.1 뷰와 뷰 그룹

-뷰

■ 사용자 인터페이스 컴포넌트를 위한 기본 빌딩블록

■ 화면에서 사각형 영역을 차지하며 그리기, 포커스 변화, 스크롤, 이벤트 처리 등을 담당

-뷰 그룹

■ 다수의 자식 뷰를 담을 수 있는 뷰 클래스의 확장

■ 다른 뷰들을 모아 관리하기 위하여 사용되는 특별한 뷰

■ 다른 뷰나 뷰 그룹을 하부에 포함할 수도 있고 독자적으로도 존재 가능

■ ViewGroup에서 파생된 클래스들은 크게 두 가지 범주인 레이아웃과 어댑터 뷰로 구별

-뷰 계층구조

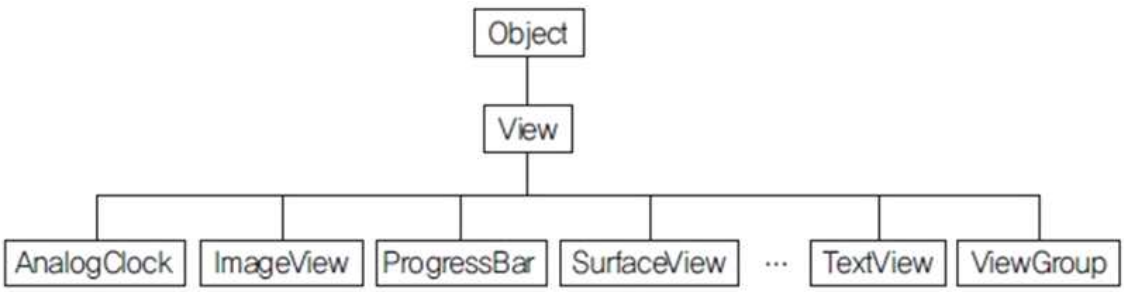


그림 5.1

5.1.2 뷰 속성과 메소드

-식별자 ID

■ 자신을 식별하기 위하여 정수 ID를 정의

■ ID는 일반적으로 레이아웃 XML 파일에서 배정되며 뷰 트리 내에서 자신을 찾기 위하여 사용

■ ID는 트리 전체에 걸쳐 유일할 필요는 없지만 찾고자 하는 뷰 내에서는 유일

```
<Button id="@+id/my_button"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="@string/my_button_text"/>
```

```
Button myButton = (Button) findViewById(R.id.my_button);
```

-크기와 위치

■ 뷰의 기하학적 모습은 폭과 높이로 표시되는 2차원의 직사각형으로 좌측 상단을 기준으로 위치를 픽셀 단위로 표시

■ layout_width와 layout_height라는 XML 레이아웃 속성을 사용하거나 ViewGroup.LayoutParams() 메소드를 사용

-여백

■ padding

■ paddingTop, paddingBottom, paddingLeft, 그리고 paddingRight

-XML 속성 일부

■ android:visibility – 위젯의 화면 표현 여부 결정.

■ android:background – 16진수 RGB(#FFFFFF 형식)로 위젯의 배경 색깔 지정.

-메소드 일부

■ setEnable() – 위젯을 활성화/비활성화 시킴.

■ IsEnable() – 위젯의 활성화 여부 check.

■ requestFocus() – method를 호출한 위젯으로 fucus를 이동.

■ getParent() – 자신이 포함된 부모 위젯이나 컨테이너 획득.

■ findViewById() – 컴파일된 Resource, R 객체를 이용해 xml layout 파일에 선언된 위젯 인스턴스 획득

■ getRootView() – 최상위 view를 획득

5.2 위젯

5.2.1 개요

-의미

■ 일반적으로 기계 혹은 장치 속성을 가지며 독특하고 신기한 소형 부품이나 도구를 통칭

■ 프로그래밍에서는 툴바, 트리, 슬라이더와 같이 사용자와 시스템 사이의 상호작용을 위한 인터페이스, 상호작용 방식에 적합하게 구조화된 컨트롤 요소를 의미

■ View의 서브클래스로 존재하며 완전한 모습을 갖춘 텍스트 필드나 버튼 같은 객체를 의미

■ 사용자에게 정보를 표시하거나 사용자의 입력을 받아들이는 컴포넌트

■ DatePicker, TimePicker, 및 ZoomControls은 뷰 그룹으로부터 파생되지만 위젯과 같이 사용

-위젯의 종류 - 1

표 5.1

클래스	개요
TextView	텍스트를 표시함
EditText	편집 가능한 텍스트를 표시함
AutoCompleteTextView	사용자의 입력에 맞춰 텍스트를 완성함
ImageView	아이콘이나 사진 등 이미지를 표시함
VideoView	비디오를 재생함
Button	기본적인 누름 버튼
CompoundButton	체크와 해제 상태를 갖는 누름 버튼
CheckBox	체크와 해제 상태를 갖는 체크박스
RadioButton	체크와 해제 상태를 갖는 라디오버튼

-위젯의 종류 - 2

표 5.2

클래스	개요
RadioGroup	복수의 라디오버튼 중 하나를 선택하는 버튼 그룹
ImageButton	이미지를 표시할 수 있는 누름 버튼
AnalogClock	현재 시각을 표시하는 아날로그 시계
DigitalClock	현재 시각을 표시하는 디지털 시계
Chronometer	경과 시간을 표시함
ProgressBar	기다리는 시간이나 경과 시간을 표시함
DatePicker	연월일을 달력에서 설정함
TimePicker	시각을 설정함
ZoomControls	이미지나 지도의 줌 레벨을 설정함

5.2.2 대표적인 위젯

-TextView

■ Label 표시

■ 문자열의 입력, 출력, 편집, 포맷에 관련된 대부분의 기능을 포함
■ android:typeface – 활자체(폰트) normal, sans, serif, monospace 중에서 지정.
■ android:textstyle – bold, italic 지정. 복수 적용 시 | 사용. (ex. "bold|italic")
■ android:textColor
■ android:text – Label 내용

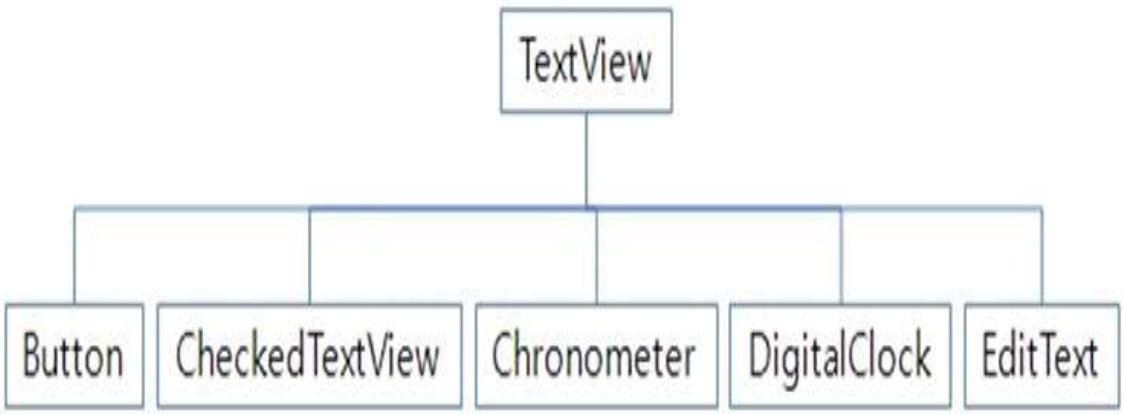


그림 5.2

-EditText
■ 사용자로부터 텍스트를 입력받을 때 사용.
■ TextView에서 파생된 것으로 EditText의 기능은 대부분 TextView에 구현되어 있으며, 편집과 관련된 속성들이 활성화
■ android:autoText – true일 경우 일반적인 스펠링 error를 잡아낸다.
■ android:capitalize – none, characters, words, sentences값을 사용할 수 있으며 각각 사용자 결정, 모든 문자, 모든 단어의 시작 문자, 모든 문장의 시작 문자를 자동으로 대문자로 변환
■ android:digits – String 형식으로 지정된 토큰만 입력 가능
■ android:singleLine – false일 경우 복수 줄 허용

-버튼
■ 사용자가 누르거나 클릭하여 명령을 내릴 수 있는 위젯

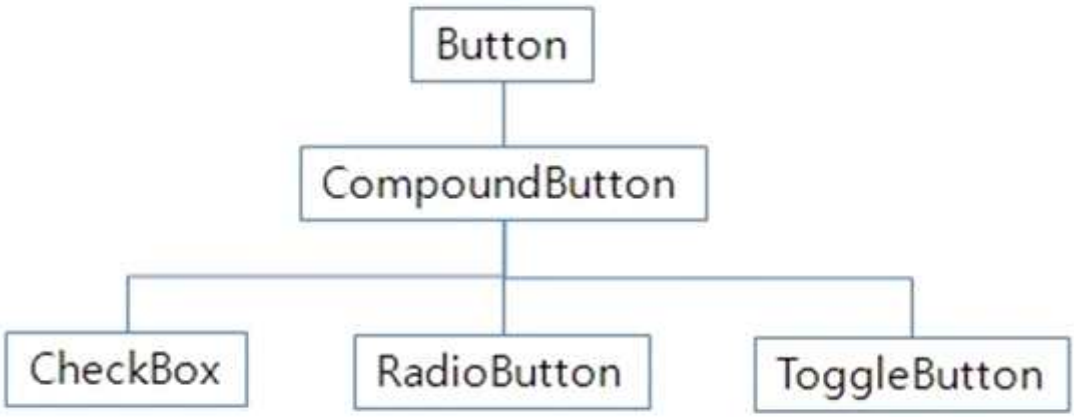


그림 5.3

■ CheckBox
- 한 목록에서 여러 개를 선택할 때 흔히 사용
- isChecked() – checkbox의 상태를 반환
- setChecked(true/false) – checkbox를 check/unckeck 함.
- toggle() – checkbox를 toggle

■ RadioButton
- 사용자가 여러 항목 중에서 하나만을 선택할 때 사용
- CompoundButton을 상속. 사용 방법은 CheckBox와 거의 동일
- RadioButton은 RadioGroup에 의해 여러 개가 한 묵음으로 사용
- 같은 RadioGroup 내부의 RadioButton은 한 번에 택일 가능.
- RadioButton은 CheckBox와 다르게 한번 check되면 사용자가 직접 unckeck 할 수 없음.
- RadioGroup 내부의 다른 RadioButton을 선택 하는 것으로 uncheck

-ImageView와 ImageButton
■ 아이콘과 같은 임의의 이미지를 표시하기 위한 위젯

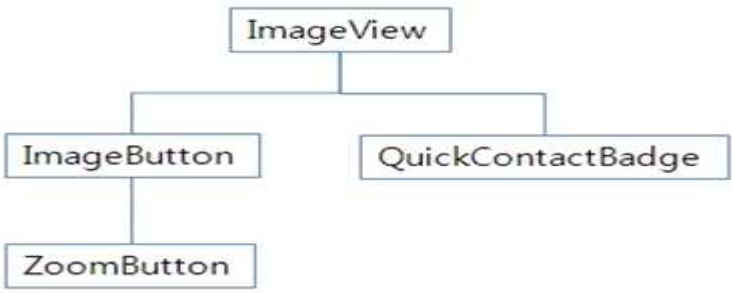


그림 5.4

■ ImageView의 대표적 속성
- src
- maxWidth와 maxHeight
- tint

5.2.3 응용
-Widget1Demo

그림 5.5

-Widget2Demo

그림 5.6

5.3 레이아웃

5.3.1 개요

-의미

■ 필요한 위젯을 효과적으로 사용하고 개성있는 화면을 만들려면 적절한 레이아웃을 사용
■ 뷰 혹은 위젯을 담고 위치를 제어하는 역할을 담당
■ View의 자손인 ViewGroup 클래스를 통하여 구현
■ 뷰의 서브 클래스로 계층 구조 내의 하부에서 뷰 혹은 컨트롤들의 위치를 제어하는 역할을 담당
■ 자바 프로그램을 통해서 정의될 수도 있지만 대부분 XML 레이아웃 파일에 따로 정의

-레이아웃의 계층도

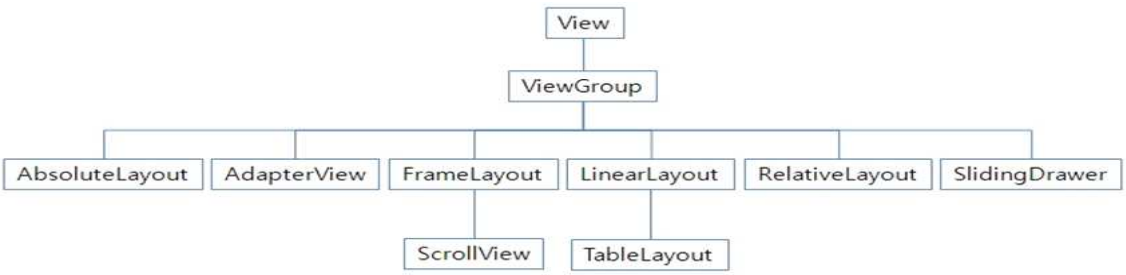


그림 5.7

-레이아웃 속성

■ 뷰의 자손이므로 뷰의 속성을 상속
■ 위젯을 화면에 배치하는 대부분의 속성은 모든 레이아웃에서 공통적으로 사용된다.
■ 레이아웃의 속성은 일반적으로 layout_로 시작. 자식 뷰가 사용하는 경우 자신의 속성을 -지정하기 보다는 부모 뷰인 레이아웃에게 자신의 위치를 지정
■ 대표적인 속성:
  • layout_weight
  • layout_gravity
  • layout_margin 등

5.3.2 선형 및 테이블 레이아웃

-LinearLayout

■ 위젯이나 다른 하위 컨테이너를 하나의 행 혹은 열 방향으로 나열

■ 단순하지만 가장 사용 빈도가 높음

■ 선형 레이아웃으로 개발자가 원하는 작업을 하려면 orientation, weight, gravity, padding 등의 특성을 활용

■ 특히 orientation 속성은 뷰의 배치 방향을 결정하며 vertical과 horizontal로 지정할 수 있으며 기본값은 horizontal

-TableLayout

■ 열과 횡의 테이블 모습을 이용하여 나열

■ LinearLayout의 서브클래스

-속성

■ <TableRow> 는 테이블의 한 행을 의미

■ <TextView> 는 문자열을 출력하는 역할

■ <view>는 가로 구분선을 표시

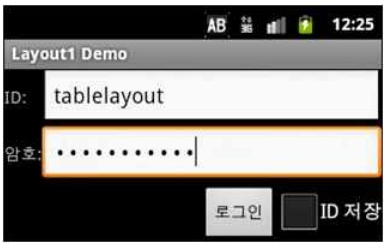


그림 5.8

5.3.3 절대 레이아웃

-AbsoulteLayout

■ 자식 뷰의 위치를 관계나 순서에 상관없이 절대 좌표로 표시하는 레이아웃

■ 자식 뷰의 위치를 layout_x와 layout_y 속성을 사용하여 명시

■ 단말기의 해상도나 방향 설정에 대하여 유연하지 못함

■ 화면 전체가 아니라 일부 영역에 대한 세밀한 표시가 요구되는 상황에 한정하여 사용

그림 5.9

5.3.4 상대 레이아웃

-RelativeLayout

■ 특정 내부 요소를 다른 내부 요소에 관련되어 위치

-속성

■ android:orientation

■ android:layout_width
■ android:layout_height
■ android:layout_above
■ android:layout_below
■ android:layout_alignBaseline
■ android:layout_alignBottom
■ android:layout_alignLeft
■ android:layout_alignRight
■ android:layout_alignTop
■ android:layout_toLeftOf
■ android:layout_toRightOf
-예

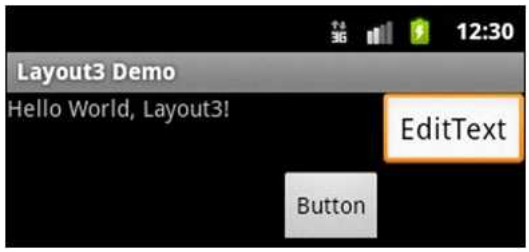


그림 5.10

5.3.5 기타 레이아웃

-FrameLayout

■ 가장 간단한 레이아웃으로 내부에 있는 뷰나 콘트롤들이 좌측 상단에 겹쳐서 차곡차곡 저장. 주로 탭(Tab)을 이용할 때 자주 사용

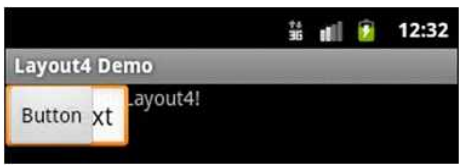


그림 5.11

-ScrollView

■ RelativeLayout 클래스의 서브클래스
■ 뷰 가장자리에 스크롤바가 생성

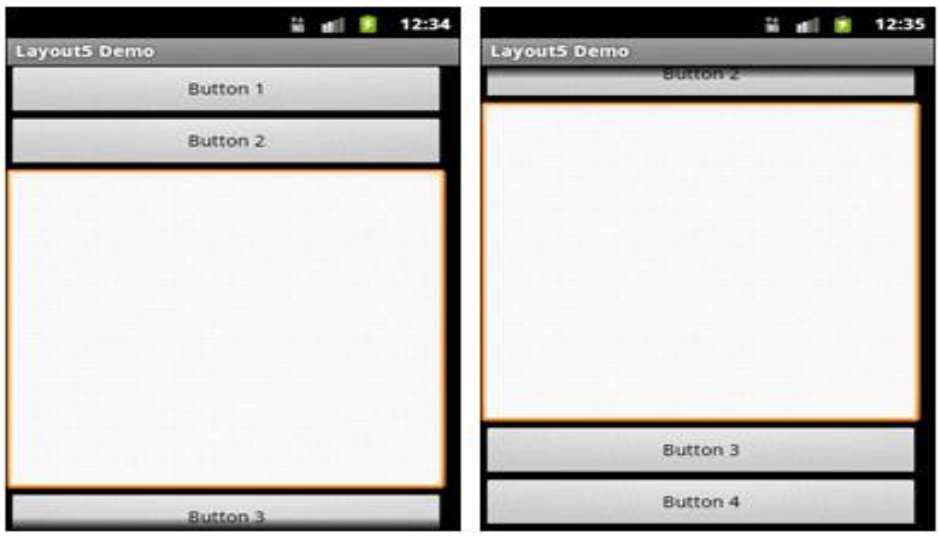


그림 5.12

제6장 메뉴와 다이어로그

6.1 재정의 메소드 추가와 …

6.1.1 재정의 메소드 추가

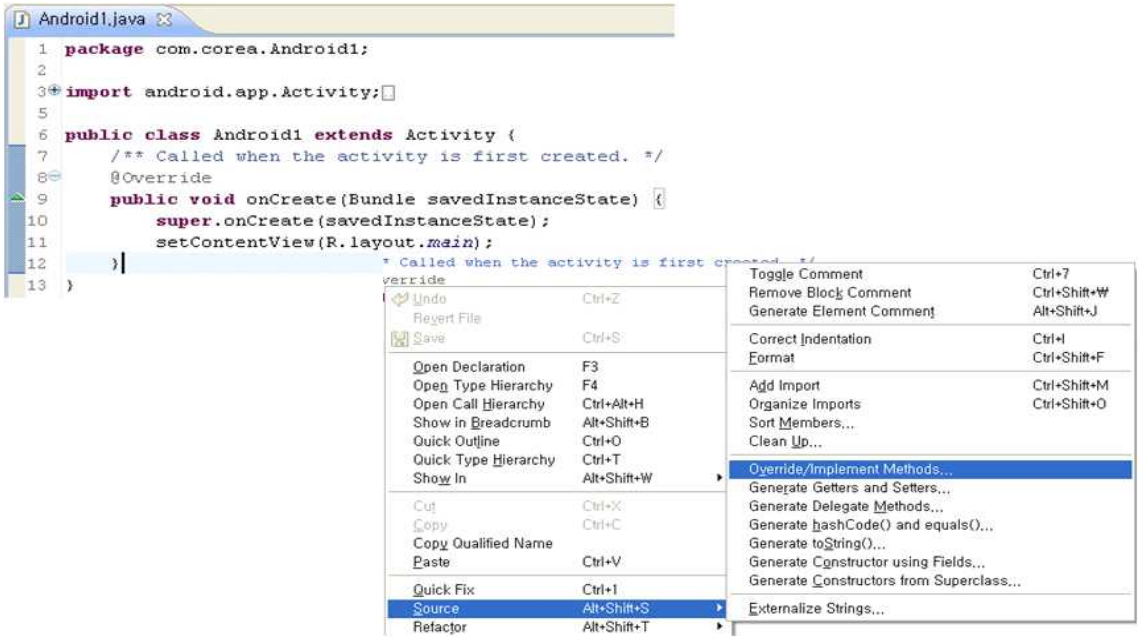


그림 6.1

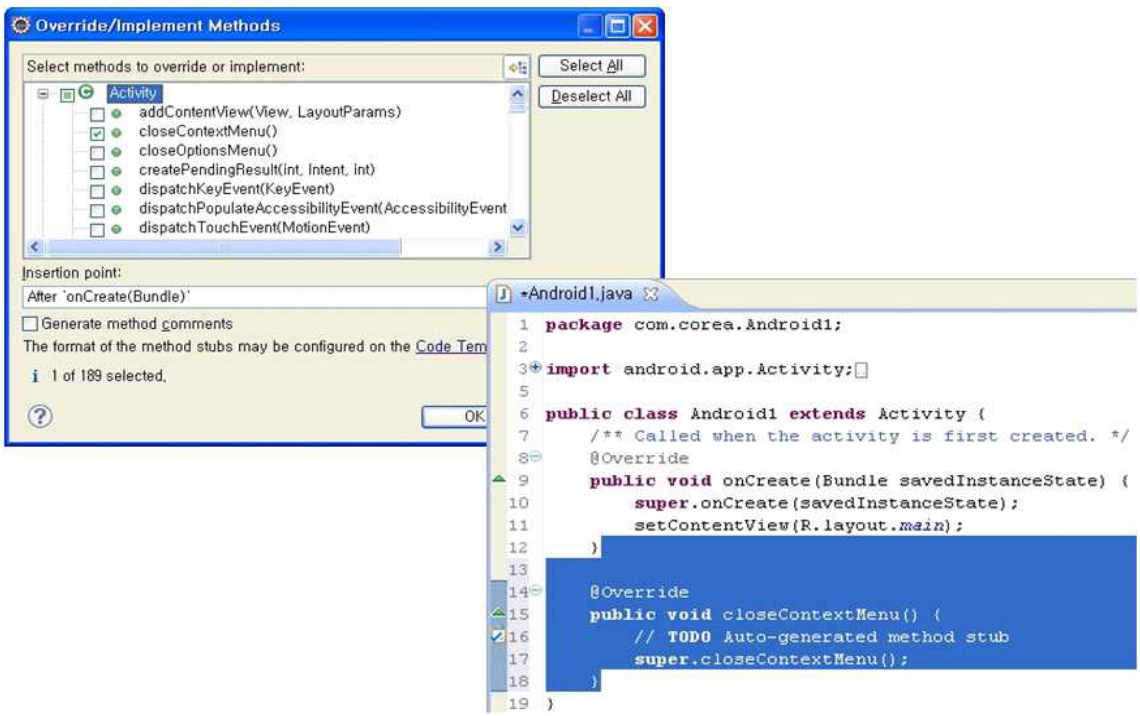


그림 6.2

6.1.2 폴더 추가

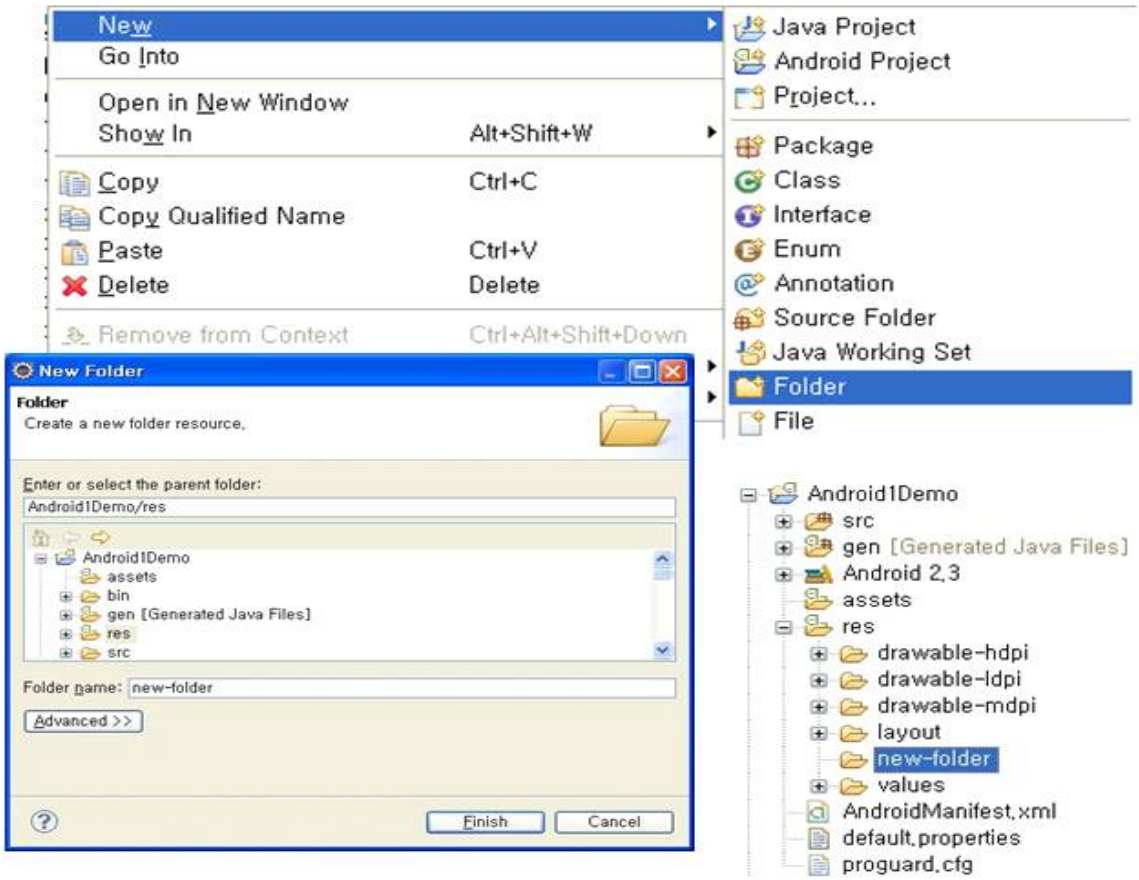


그림 6.3

6.1.3 XML 파일 생성

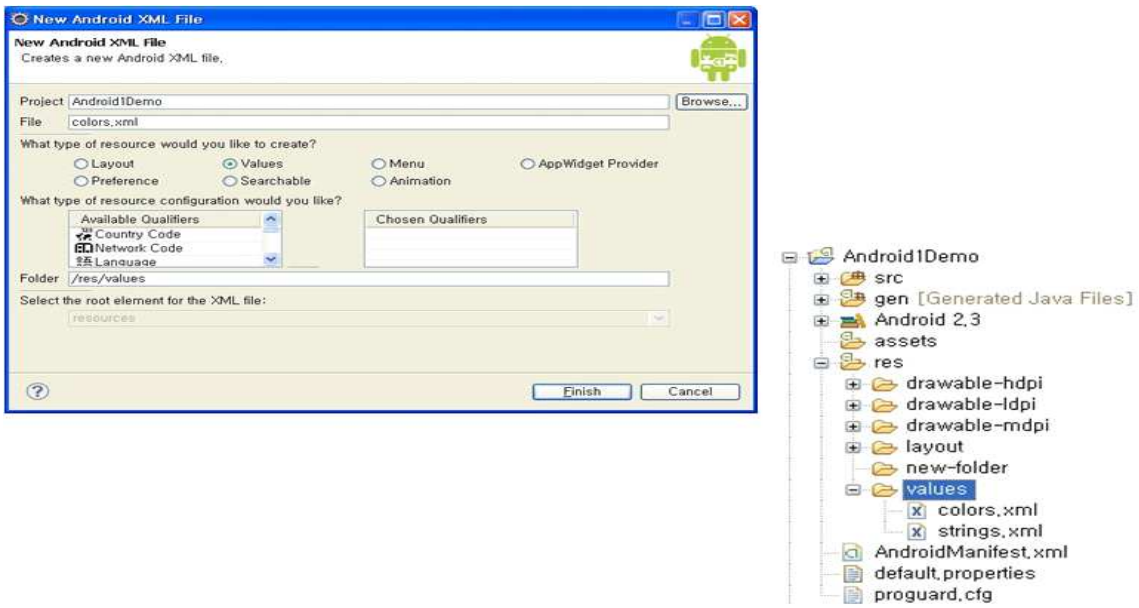


그림 6.4

6.2 메뉴

6.2.1 개요

-의미

■ 숨겨두었다가 필요할 때 화면에 표시할 수 있는 UI
■ 애플리케이션 수준에서 옵션 · 컨텍스트 · 서브 메뉴를 제공
■ 옵션 메뉴
• 모바일 단말기의 MENU 버튼을 누르면 화면 하단에 나타나는 메뉴를 의미
• 아이콘 메뉴와 확장 메뉴로 세분
■ 컨텍스트 메뉴
• 화면의 특정 뷰를 길게 누르는 동작에 의하여 나타나는 메뉴를 의미
• 옵션 메뉴와 달리 컨텍스트에 민감한 메뉴 항목을 뷰에 제공하고자 할 때 주로 사용
■ 서브 메뉴
• 옵션 메뉴나 컨텍스트 메뉴에 의한 메뉴 항목이 또 다른 항목을 포함하는 일종의 팝업창으로 나타나는 메뉴
• 서브 메뉴는 또 다른 서브 메뉴를 포함할 수 없다.

6.2.2 옵션 메뉴

-아이콘 메뉴:

■ 애플리케이션 실행 도중 모바일 장치의 MENU 버튼을 클릭하면 화면 하단에 나타나는 메뉴
■ 최대 6개의 메뉴 항목을 지원
■ 아이콘을 지원하는 유일한 메뉴 방식이며 체크박스 혹은 라디오버튼을 지원하지 않는 유일한 메뉴 방식
■ 메뉴 항목이 7개 이상이 되는 경우에는 화면 하단에 이들을 모두 표시할 수 없다.

-확장 메뉴:

■ 아이콘 메뉴에서 More라는 메뉴 항목을 누르면 노출되는 메뉴
■ 옵션 메뉴가 7개 이상의 항목을 포함하면 6번째 메뉴 항목부터는 자동적으로 확장 메뉴에 나타난다.

-메뉴의 생성

■ onCreateOptionsMenu(Menu menu)
■ 사용자가 MENU 버튼을 누르면 호출되는 액티비티 메소드
■ 안드로이드는 표준 옵션 메뉴의 내용을 초기화하기 위하여 호출
■ 메뉴가 만들어질 때 한번만 호출

-메뉴 항목 추가

■ Menu 객체에 메뉴 항목이나 서브 메뉴 항목 추가
■ public abstract MenuItem add(int groupId, int itemId, int order, CharSequence title 혹은 int titleRes)
■ public abstract SubMenu addSubMenu (int groupId, int itemId, int order, CharSequence title 혹은 int titleRes)

-실행중 메뉴 항목 변화

■ onPrepareOptionsMenu(Menu menu)
■ 실행 중에 메뉴 항목을 변경하고자 한다면 다음 메소드를 재정의

■ 애플리케이션의 상태에 따라 메뉴 내용을 동적으로 수정하거나 메뉴 항목을 효율적으로 활성화 혹은 비활성화

-메뉴 항목 선택

■ onOptionsItemSelected(MenuItem item)

■ 옵션 메뉴에서 메뉴 항목을 선택하면 호출되는 메소드

-예제

그림 6.5

6.2.3 컨텍스트 메뉴

-의미

■ 마우스 우측 버튼을 클릭할 때 나타나는 메뉴와 개념적으로 비슷

■ 뷰를 길게 누르면 뷰와 관련된 기능을 표시하는 플로팅 메뉴

■ 컨텍스트 메뉴를 생성하는 방법은 구조적으로 옵션 메뉴와 유사. 그러나 컨텍스트 메뉴는 액티비티 혹은 특정 뷰에 종속된 메뉴이므로 뷰와 컨텍스트 메뉴를 연관시켜야 함

■ 컨텍스트 메뉴는 ListView 항목을 취급할 때 자주 사용하지만 어떤 뷰를 위해서도 생성 가능

-메뉴 생성

■ onCreateContextMenu (ContextMenu menu, View v, ContextMenu. ContextMenuInfo menuInfo)

-메뉴 항목 선택

■ onContextItemSelected(MenuItem item)

-컨텍스트 메뉴의 뷰에 등록

■ registerForContextMenu(View view)

■ setOnCreateContextMenuListener(View.OnCreateContextMenuListener l)

-예제

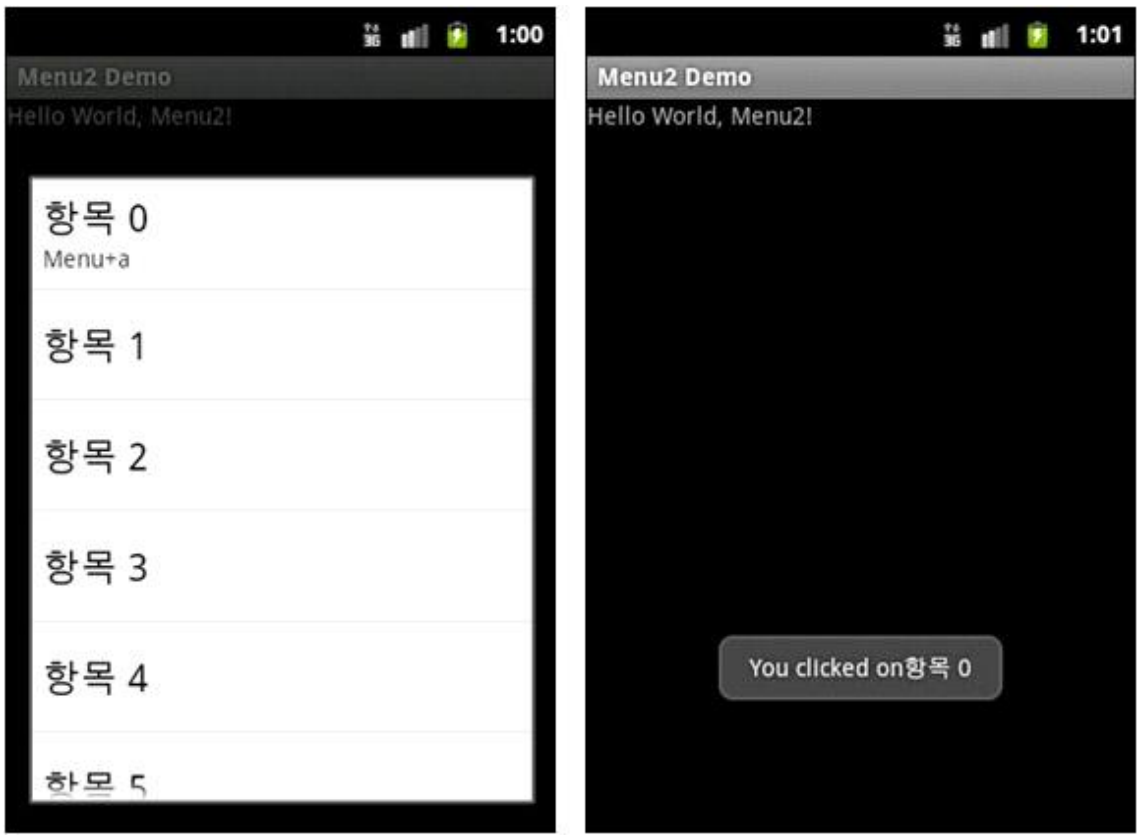


그림 6.6

6.2.4 메뉴 전개

-의미

■ 리소스를 사용하여 정의한 메뉴를 코드로 정의

■ 메뉴도 리소스이기 때문에 코드가 아닌 XML 파일을 사용하면 훨씬 쉽게 정의

■ 메뉴 구조를 정의한 리소스 XML 파일을 다른 리소스처럼 프로젝트의 res/menu 폴더 혹은 res/string 폴더에 저장 가능

■ 메뉴의 종류가 많으면 res/menu 폴더를 생성하여 메뉴를 위한 XML 파일을 관리

-XML 파일로 메뉴 정의

■ 3개의 유효한 엘리먼트 - menu, group, 그리고 item - 가 존재

■ 루트 엘리먼트는 반드시 menu

■ Item과 group 엘리먼트는 menu의 자식
■ 또한 item 엘리먼트는 group 엘리먼트의 자식이 될 수 있음
■ 서브 메뉴를 생성하기 위하여 또 다른 menu 엘리먼트가 item의 자식이 될 수도 있음
-메뉴 전개자 생성과 메뉴 전개
■ 메뉴 전개자 생성
  • public MenuInflater getMenuInflater ()
  • 현재 액티비티에서 MenuInflater 객체를 생성하여 반환하는 메소드
■ 메뉴의 전개
  • MenuInflater 객체를 사용하여 inflate() 메소드를 호출

• 예제

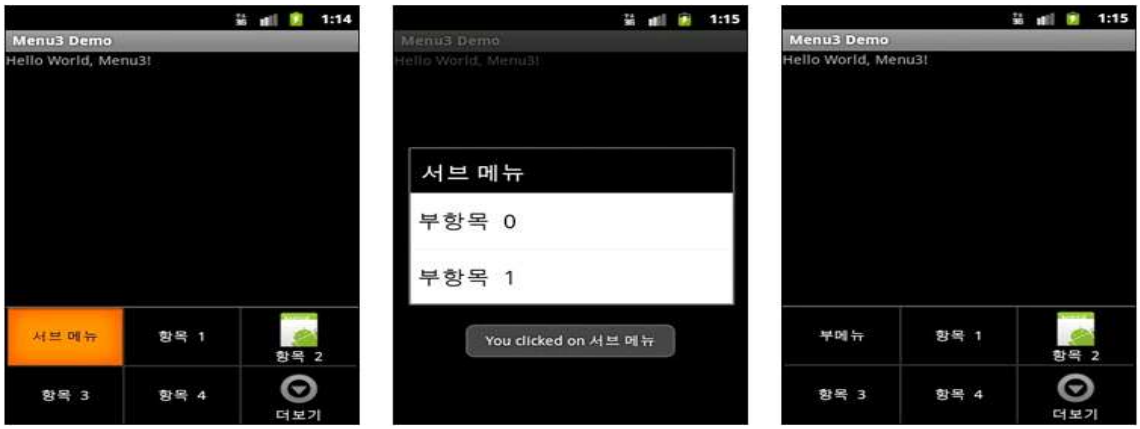


그림 6.7

6.3 다이어로그

6.3.1 개요

-의미
■ 문자 그대로 시스템과 대화하기 위한 윈도우
■ 사용자에게 시스템의 현재 상태 및 오류 사항 등을 알려주기 위하여 경고 메시지를 띄우거나 질문 사항을 보내어 사용자의 선택을 받아들이는 기본적인 통신 수단
-대표적인 Dialog 클래스

표 6.1

종류	의미
AlertDialog	아이콘, 메시지, 버튼 3개를 가지며 가장 많이 사용됨
DatePickerDialog	달력에서 날짜를 제공하여 날짜 설정을 가능하게 함
ProgressDialog	실행 상태를 통지하는 진행 바를 포함
TimePickerDialog	시간을 제공하여 시간 설정을 가능하게 함
ZoomDialog	줌 레벨을 선택함. 주로 지도에서 사용

6.3.2 경고 다이어로그

-의미

■ 윈도우즈의 모달 다이어로그 방식과 유사

■ 경고 다이어로그에서 버튼을 누르지 않으면 진행하던 작업으로 돌아가거나 다른 작업으로 진행할 수 없음

■ 작업의 결과나 경고 등 중요한 메시지를 사용자에게 전달할 때 경고 다이어로그를 사용

■ Dialog 클래스의 확장 클래스

-생성과 표시

■ AlertDialog.Builder(Context) 메소드를 사용하여 Builder 객체를 생성한 후 경고 다이어로그의 세부 속성을 설정

■ AlertDialog.Builder() 메소드의 매개변수인 Context를 위해서는 다이어로그를 생성하는 부모 액티비티를 사용

■ 세부 속성을 설정한 후 show() 메소드를 사용하면 Builder 객체를 생성하고 화면에 표시

-세부 속성 설정

■ public AlertDialog.Builder setIcon (int iconId 혹은 Drawable icon)

■ public AlertDialog.Builder setMessage (int messageId 혹은 CharSequence message)

■ public AlertDialog.Builder setTitle (int titleId 혹은 CharSequence title)

-예제 1

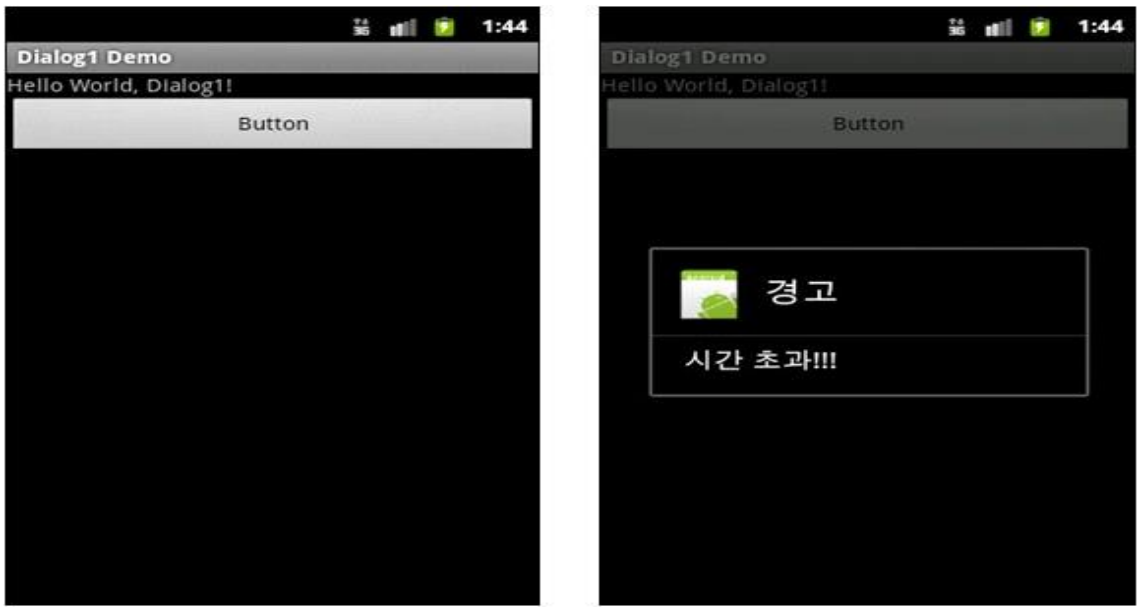


그림 6.8

-경고 다이어로그의 3가지 버튼 설정

■ public AlertDialog.Builder set...Button
(CharSequence text 혹은 int textId, DialogInterface.OnClickListener listener)

■ …은 Negative, Neutral, Positive 중 하나

■ Back 버튼에 의한 다이어로그 취소 금지

■ public AlertDialog.Builder setCancelable (boolean cancelable)

- 예제 2

그림 6.9

-3개의 버튼 외에 리스트 혹은 체크박스와 라디오버튼의 추가

■ public AlertDialog.Builder setItems
(CharSequence[] items, DialogInterface.OnClickListener listener)

-예제 3

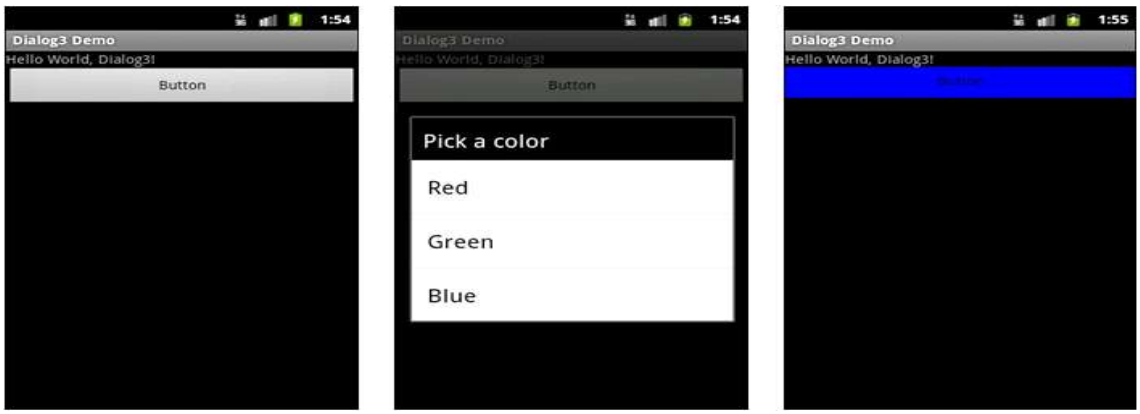


그림 6.10

-예제 4

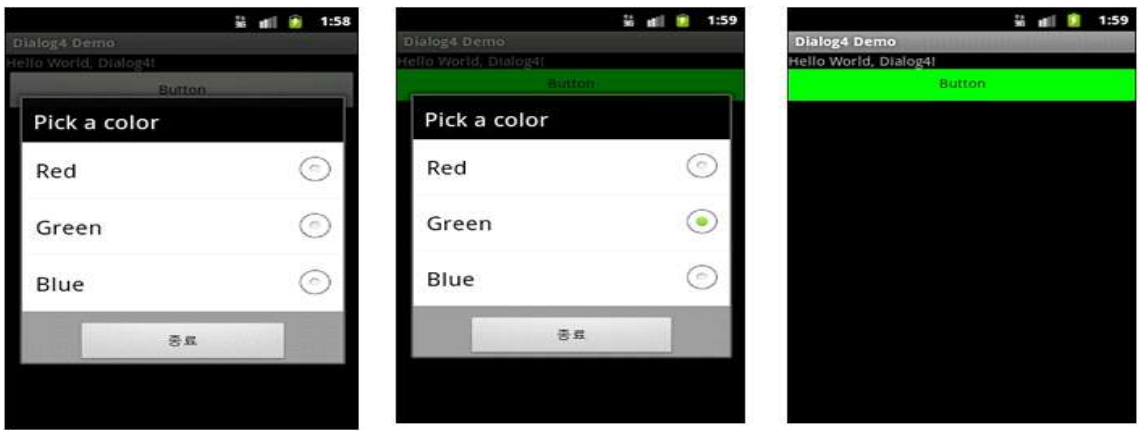


그림 6.11

6.3.3. 날짜 선택 다이어로그

-의미

■ 달력을 기준으로 월 · 일 · 연도을 선택하는 위젯

■ DatePicker를 포함하는 간단한 다이어로그

■ 날짜 선택 다이어로그에서 사용자가 선택한 날짜 값을 받아올 수 있도록 OnDateSetListener()를 설정

■ 날짜 선택 다이어로그에서 연월일 값을 선택하면 onDateSet() 메소드를 호출하며 이 메소드를 통해 사용자가 선택한 연월일 값을 화면에 표시

-예제

그림 6.12

제7장 어댑터와 어댑터뷰

7.1 어댑터

7.1.1 개요

-의미

■ 두 개의 상이한 부분을 연결시키는 데 사용하는 장치

■ 외부 데이터 소스와 어댑터뷰 사이의 연결 수단

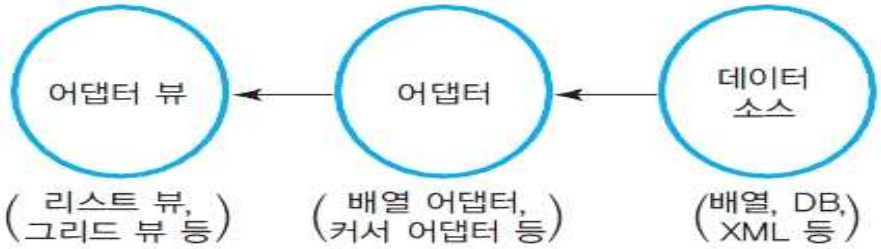


그림 7.1

■ 유사한 데이터 집단을 편리하게 화면에 표시할 수 있는 인터페이스
-역할
■ 데이터 소스를 접근하여 데이터 항목을 읽어오고,
■ 각 데이터 항목을 위한 뷰를 생성한다.

-어댑터 계층구조

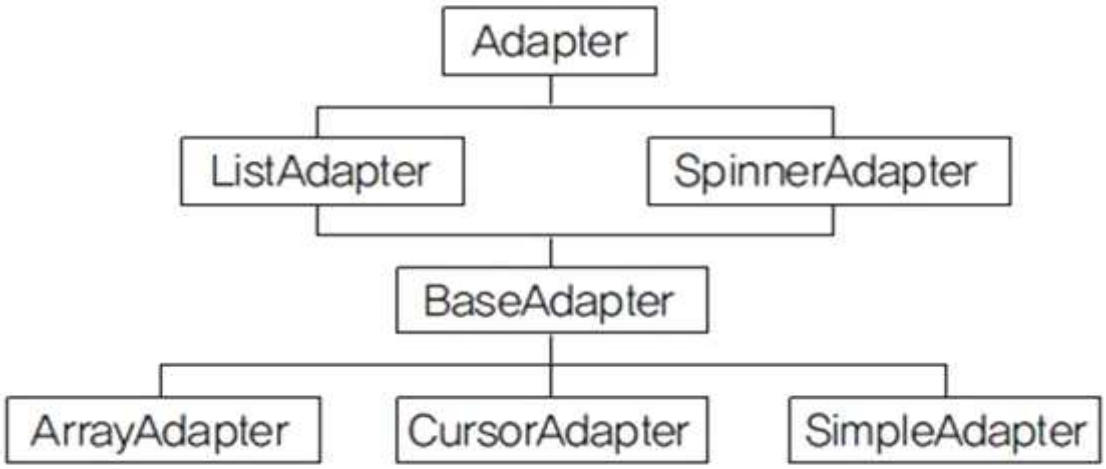


그림 7.2

7.1.2.어댑터 메소드

-데이터 소스를 접근하여 데이터 항목을 읽어오는 역할
■ public abstract int getCount ()
■ public abstract Object getItem (int position)
■ public abstract long getItemId (int position)
-뷰를 생성하기 위한 메소드
■ public abstract View getView (int position, View convertView, ViewGroup parent)

7.1.3 배열 및 단순 어댑터

-배열 어댑터
■ ArrayAdapter
■ 배열 어댑터는 문자열 집합과 같은 배열로 구성된 데이터 소스를 어댑터뷰와 연관시키기 위한 어댑터
■ 객체 생성
  • 생성자public ArrayAdapter (Context context, int textViewResourceId, List<T> objects 혹은 T[] objects)
  • 데이터 소스가 외부 리소스로 구성된 배열이라면public static ArrayAdapter<CharSequence> createFromResource (Context context, int textArrayResId, int textViewResId)

-배열 어댑터
■ 대표적인 레이아웃 리소스 ID: platforms₩android-<버전>₩data₩res₩layout 폴더에 있음

표 7.1

리소스 ID	의미
simple_list_item_1	하나의 텍스트 뷰로 구성된 레이아웃
simple_list_item_checked	체크가 표시되는 레이아웃
simple_list_item_single_choice	라디오 버튼이 표시되는 레이아웃
simple_list_item_multiple_choice	체크 버튼이 표시되는 레이아웃

■ 데이터 소스의 변화가 발생하면 다음과 같은 메소드를 사용하여 어댑터뷰에게 알리고 재생 필요public void notifyDataSetChanged ()

-단순 어댑터

■ SimpleAdapter

■ 정적인 데이터를 XML 파일에서 정의한 뷰로 사상해주는 어댑터

■ 키와 값의 쌍으로 구성된 Map 객체를 포함하는 ArrayList가 데이터 소스인 경우에 적합

■ 객체 생성: 생성자 이용public SimpleAdapter (Context context, List<? extends Map<String, ?>> data, int resource, String[] from, int[] to)

7.2 어댑터뷰

7.2.1 개요

-의미

■ ViewGroup의 서브클래스로써 다른 뷰를 담는 컨테이너 역할을 수행하는 것은 당연

■ ViewGroup의 자손인 레이아웃과는 달리 일반적인 위젯처럼 사용자와 상호작용 가능

-역할

■ 어댑터로부터 공급받은 데이터로 레이아웃을 채우고

■ 또한 사용자에 의한 항목 선택 이벤트를 처리한다.

-어댑터뷰 계층구조

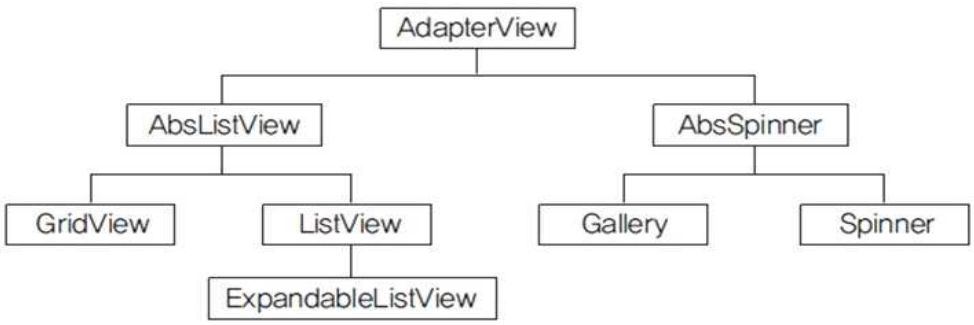


그림 7.3

7.3 어댑터뷰 메소드

-어댑터와 바인딩

■ 어댑터는 코드에서 제공하는 리스트나 디바이스의 데이터베이스에 대한 쿼리 결과와 같은 외부 리소스로부터 데이터를 가져온다

■ public abstract void setAdapter (T adapter)
-사용자의 데이터 항목 선택에 대한 이벤트를 처리하기 위한 핸들러 등록
■ public void setOnItemClickListener (AdapterView.OnItemClickListener listener)
■ public void setOnItemLongClickListener (AdapterView.OnItemLongClickListener listener)

7.4 리스트뷰

7.4.1 개요
-의미
■ 어댑터뷰 중에서 가장 자주 사용되는 위젯
■ 화면에 표시할 내용들이 리스트 형태로 되어 있는 경우 수직으로 펼쳐서 표시하기 위하여 사용
■ 데이터 항목은 ListAdapter사 공급
-속성
■ choiceMode: 리스트뷰의 선택 특성을 의미. 속성값으로 none, singleChoice, 그리고 multipleChoice를 지정.
■ divider: 리스트 항목 사이를 구분하기 위하여 그려질 Drawable 객체나 색상을 의미.
■ dividerHeight: 구분선의 높이를 의미
-메소드
■ public void setChoiceMode (int choiceMode)
■ public void setDivider (Drawable divider)
■ public void setDividerHeight (int height)
■ public void clearChoices ()
■ public int getCheckedItemPosition ()

7.4.2 ListActivity
-개요
■ 리스트뷰는 매우 자주 사용되며, 또한 리스트뷰를 사용하는 프로젝트는 대부분 거의 비슷한 구조를 가짐
■ 리스트뷰 + 액티비티
■ 배열이나 커서와 같은 데이터 소스를 바인딩함으로써 데이터 항목의 리스트를 표시하고 하나의 항목을 선택하면 이벤트 핸들러를 노출하는 액티비티
■ ListView를 포함하며 관련된 이벤트 리스너가 이미 등록되어 있음
-주요 메소드
■ public ListView getListView ()
■ public long getSelectedItemId ()
■ public int getSelectedItemPosition ()
■ public void setListAdapter (ListAdapter adapter)
■ protected void onListItemClick (ListView l, View v, int position, long id)

7.4.3 응용
-실습 1 & 2
■ ListView1Demo: 코드에서 배열 선언
■ ListView2Demo: XML 파일로 배열 선언

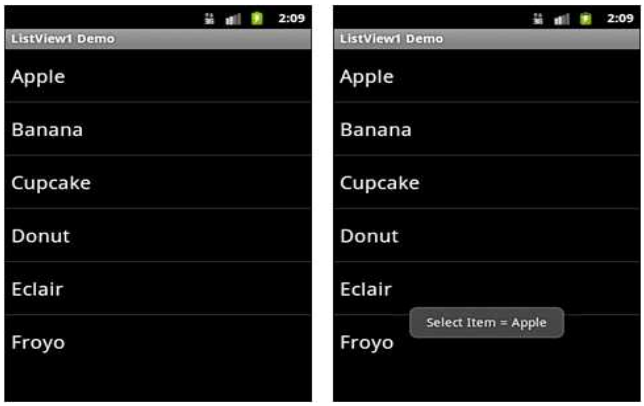


그림 7.4

-실습 3

■ ListView3Demo: 데이터 추가 기능

그림 7.5

-실습 4

■ ListView1Demo와 동일한 결과를 나타내는 ListActivity1Demo

그림 7.6

-실습 5

■ ListView3Demo와 유사한 결과를 나타내는 ListActivity2Demo

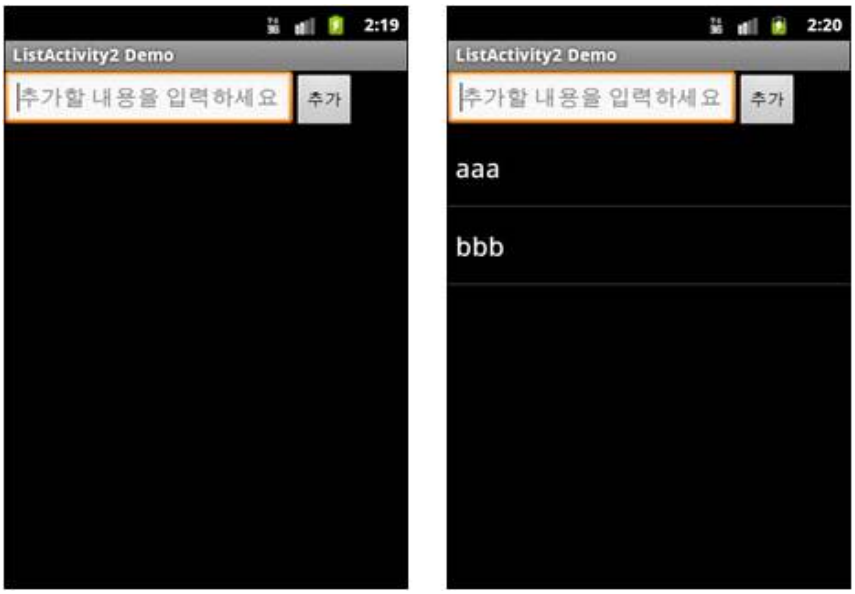


그림 7.7

-실습 6

■ 데이터 추가 및 삭제 기능 추가한 ListActivity3Demo

그림 7.8

7.5 어댑터 응용

7.5.1 개요

-리스트뷰 외에도 스피너, 갤러리, 그리드뷰 등과 같은 어댑터뷰가 존재

- 일반적으로 리스트뷰를 가장 흔히 사용
- 제한된 항목에 대한 선택 혹은 사진 이미지의 관리 등과 같은 경우 다른 종류의 어댑터뷰를 사용
- 리스트뷰는 배열 어댑터와 매칭 양호
- 다른 어댑터뷰는 다른 종류의 어댑터 - SimpleAdapter 혹은 BaseAdapter - 를 사용하는 경우가 다양

7.5.2 응용

-Spinner

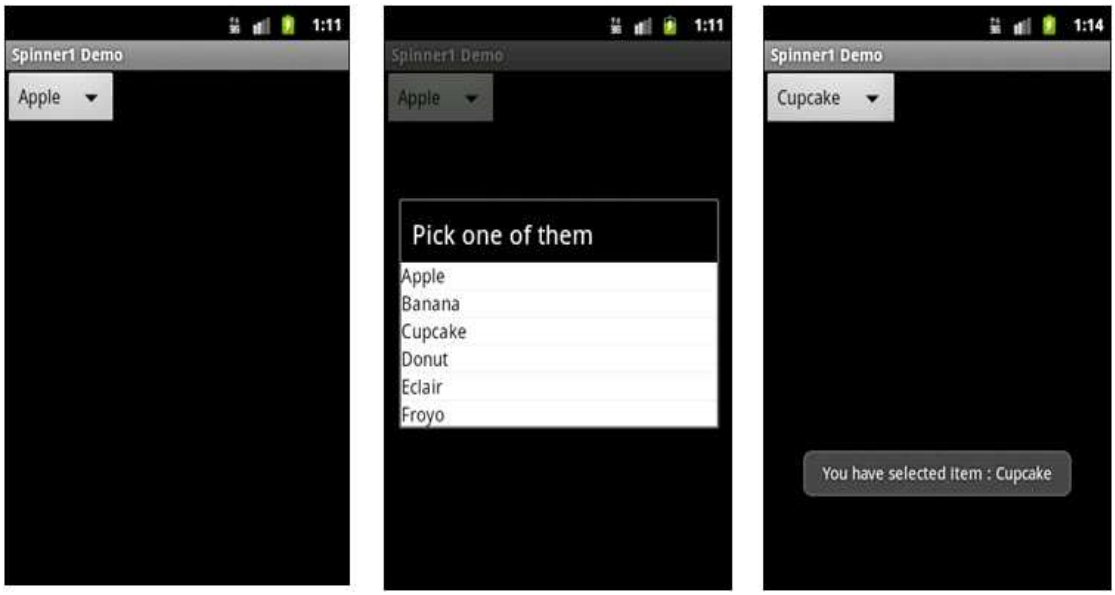


그림 7.9

-SimpleAdapter를 이용한 리스트뷰

그림 7.10

-Gallery

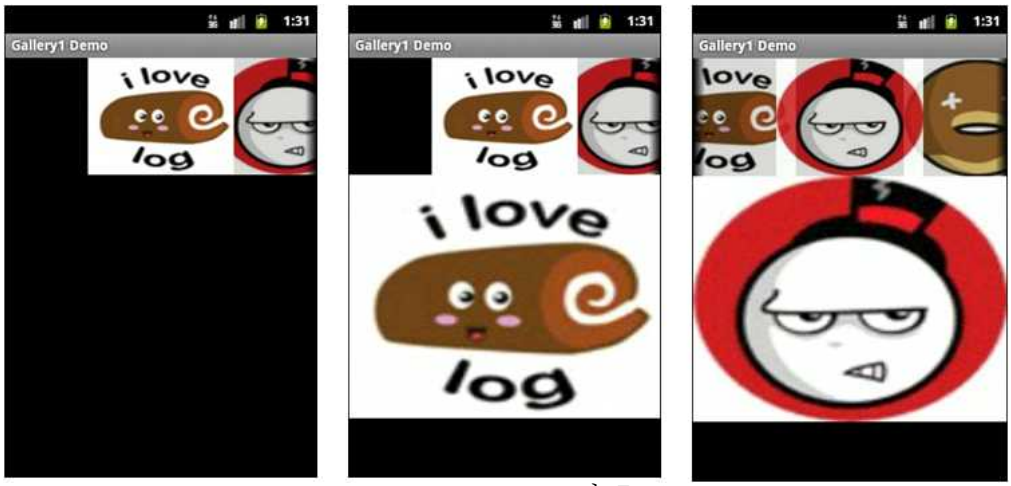


그림 7.11

-GridView

그림 7.12

제8장 인텐트

8.1 액티비티 및 퍼미션 추가

8.1.1 개요

-애플리케이션

■ 대부분 하나 이상의 애플리케이션 컴포넌트로 구성

■ 하나의 컴포넌트에서 다른 컴포넌트를 호출하려면 "다른 컴포넌트를 호출하고 싶다"는 의사 표현이 필요

■ 의사표현의 수단으로 인텐트 객체를 사용

-인텐트

■ 2개 이상의 애플리케이션 컴포넌트를 연결시키는 일종의 아교 역할

■ 인텐트를 실습하기 위하여 자바 파일이나 액티비티를 프로젝트에 추가
■ 인텐트는 다른 애플리케이션의 컴포넌트 호출 가능. 일부 애플리케이션은 권한이 부여되어야 컴포넌트를 실행 가능 → 안드로이드 매니페스트 파일에 퍼미션 추가 필요

8.1.2 자바 파일 추가

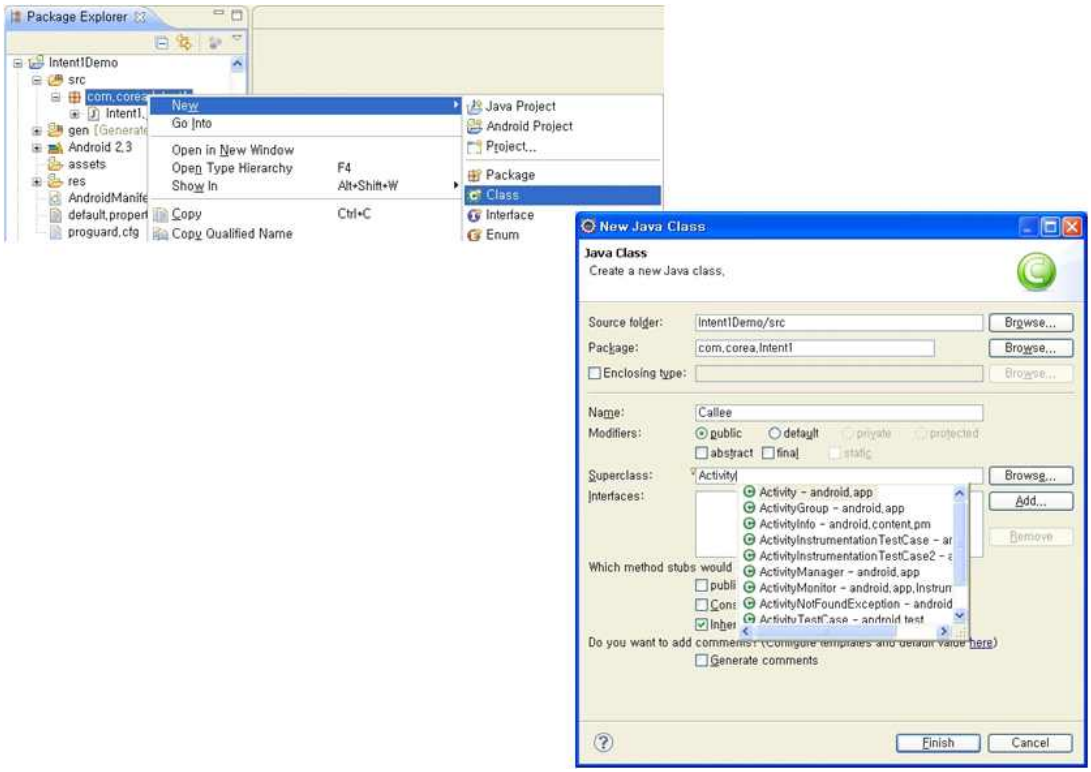


그림 8.1

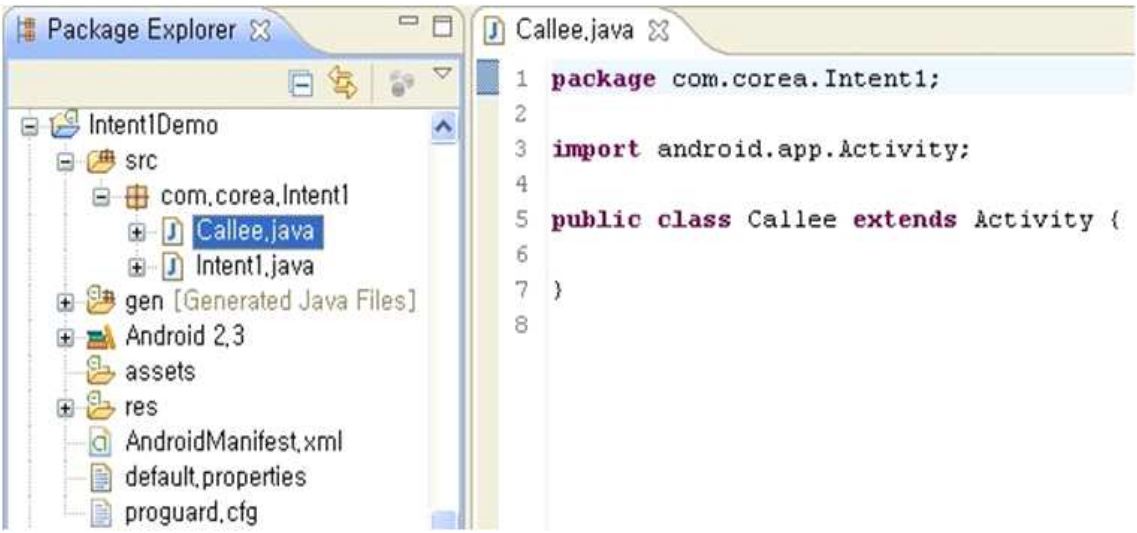


그림 8.2

-매니페스트 파일은?

8.1.3 액티비티 추가

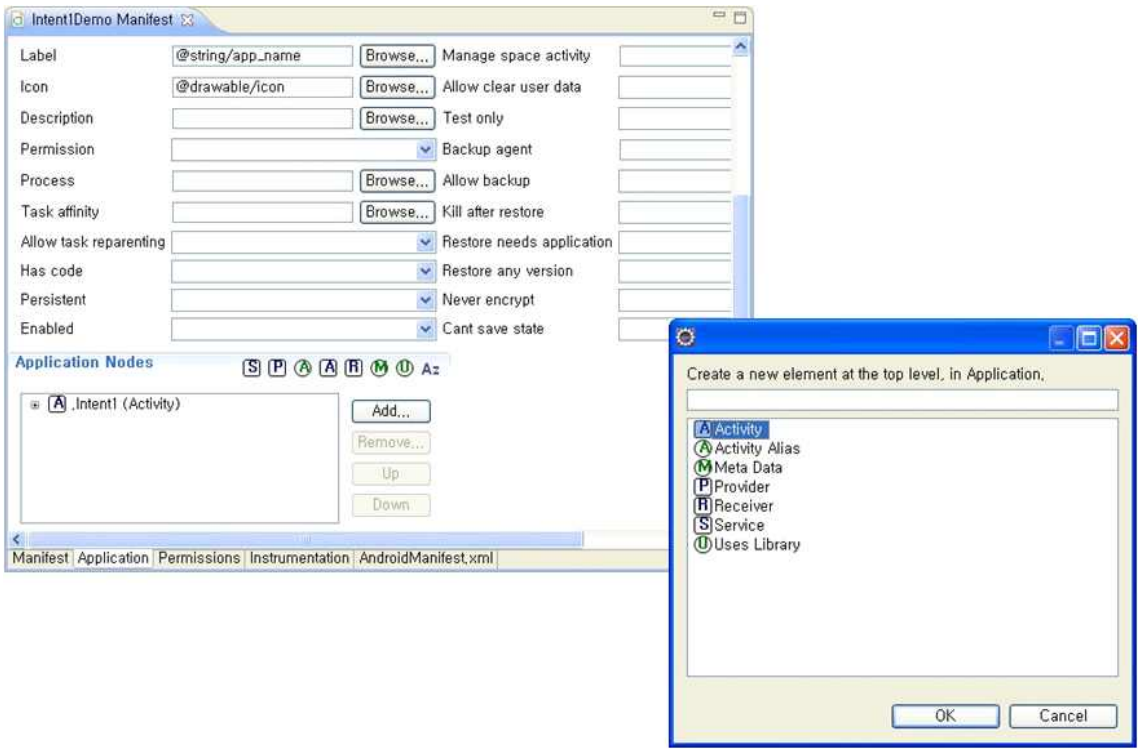


그림 8.3

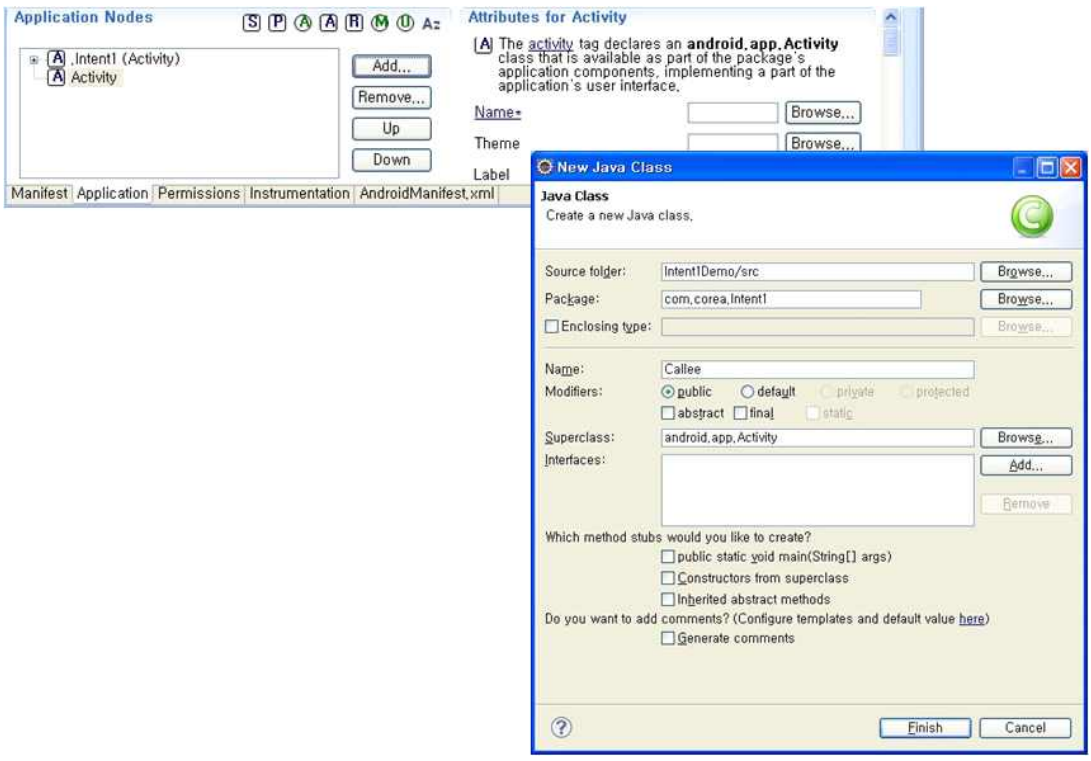


그림 8.4

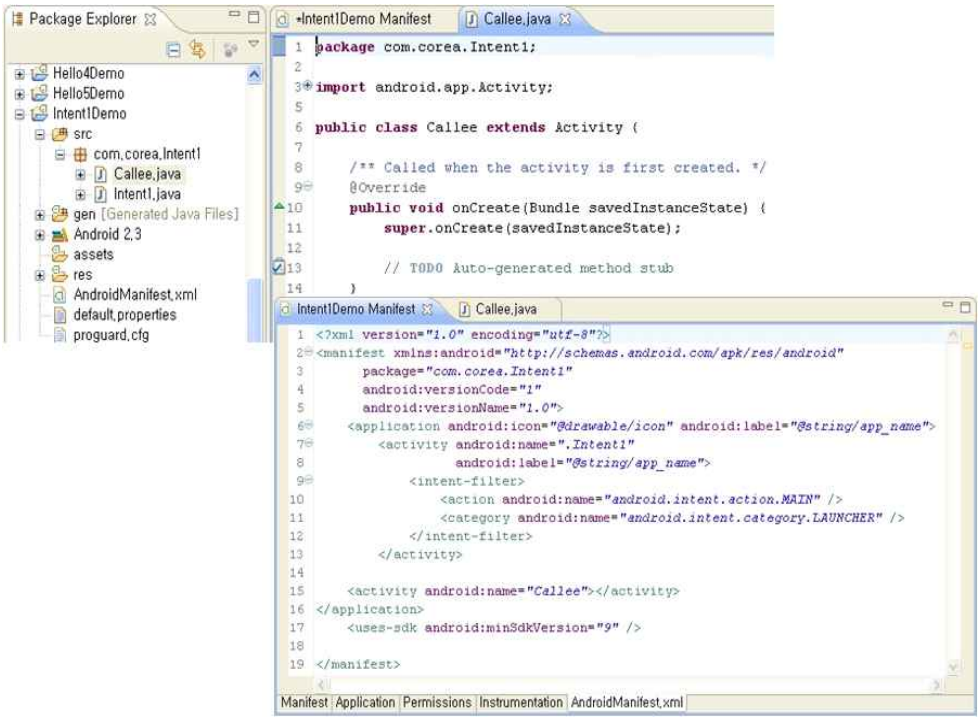


그림 8.5

8.1.4 퍼미션 추가

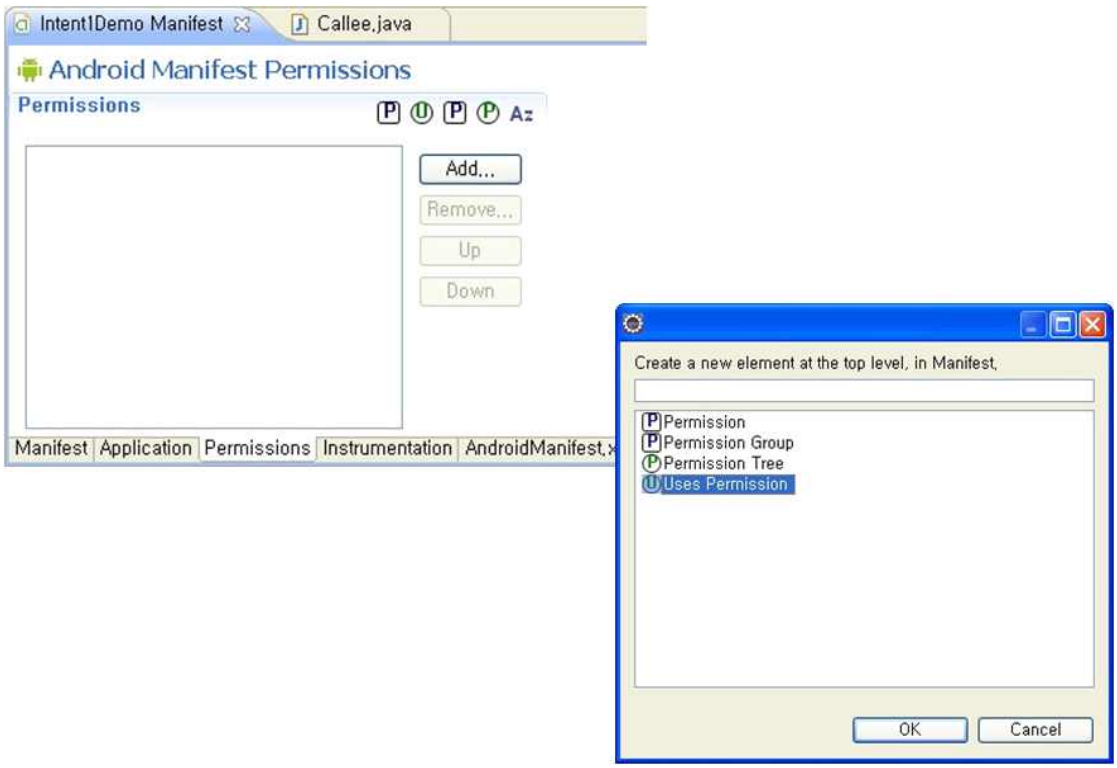


그림 8.6

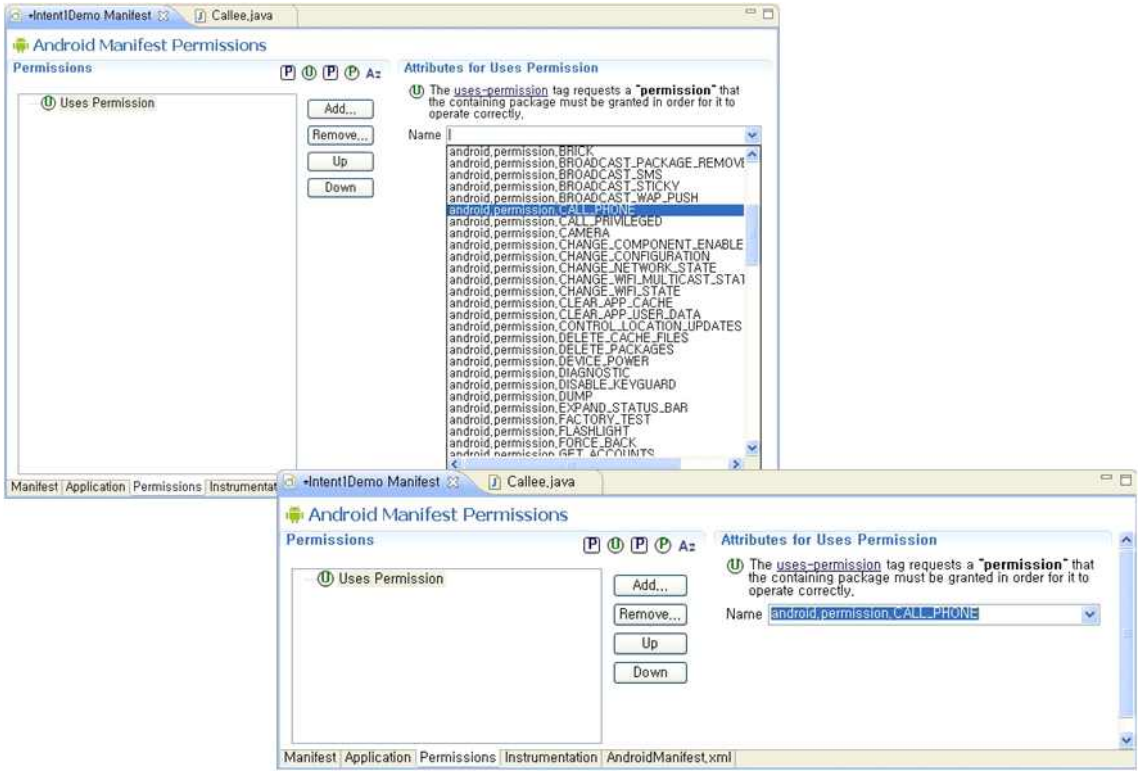


그림 8.7

8.2 인텐트 개요

8.2.1 개요

-애플리케이션 컴포넌트

- 액티비티, 서비스, 컨텐트 공급자, 그리고 방송수신자
- 컴포넌트들은 애플리케이션에서 느슨하게 결합
- 결합된 컴포넌트의 내역은 안드로이드 매니페스트 파일에 포함
- 느슨한 결합은 다른 시스템과 차별화할 수 있는 안드로이드 시스템의 중요한 특징
- 비록 애플리케이션끼리 철저하게 격리하지만 권한만 있다면 안드로이드는 하나의 애플리케이션에 포함된 컴포넌트를 다른 애플리케이션의 컴포넌트로 결합 허용

-인텐트란?

- 사전적 의미는 '의도'
- 기능적으로 말하면 컴포넌트를 상호 연결하기 위한 메시지
- 경계 없는 애플리케이션 컴포넌트를 상호 연결해주며, 또한 액티비티, 서비스, 그리고 방송수신자라는 3가지 애플리케이션 컴포넌트를 활성화시키는 메시지
- 메시지는 안드로이드의 애플리케이션 컴포넌트 사이에 통신 수단으로 동작
- 메시지 자체가 작업을 할 수 있는 것이 아니라 단지 수단.
- 메시지를 읽고 메시지의 내용을 직접 수행하는 것은 애플리케이션 컴포넌트

8.2.2 인텐트 생성

-생성자

■ Intent()
■ Intent(Intent o)
■ Intent(String action)
■ Intent(String action, Uri uri)
■ Intent(Context packageContext, Class<?> cls)
■ Intent(String action, Uri uri, Context packageContext, Class<?> cls)
-예제
■ Caller 액티비티가 Callee 액티비티를 호출하는 메시지를 가진 인텐트 객체를 생성한다. 그리고 arg라는 키에 “test"라는 값을 가진 엑스트라를 인텐트에 추가
Intent intent = new Intent(Caller.this, Callee.class);intent.putExtra("arg", "test");

8.2.3 간단한 예제

-Intent1Demo 프로젝트

그림 8.8

8.2.4 인텐트 종류

- 명시적 인텐트(Explicit Intent)
■ 호출 대상 컴포넌트의 이름이 명시된 경우
■ 메세지 자체에 해당 메세지를 전달해야 하는 수신처(컴포넌트 이름)가 명확하게 명시된 경우
-암시적 인텐트(Implicit Intent)
■ 호출 대상 컴포넌트의 특성만 나열된 경우
■ 주어진 단서(Action/Data/Category)를 기반으로 가장 적합한 하나의 컴포넌트를 찾아내야 하는 경우
■ 라우팅 규칙이 있음. 예를 들어, 인텐트에 액션 값이 명시되어 있다면 인텐트 필터에 해당 액션 값이 명시되야 함 등.

8.3 명시적 인텐트

8.3.1 개요

-생성 방법

■ Intent intent = new Intent(this, SubActivity.class)

-API

■ public Intent (Context packageContext, Class<?> cls)

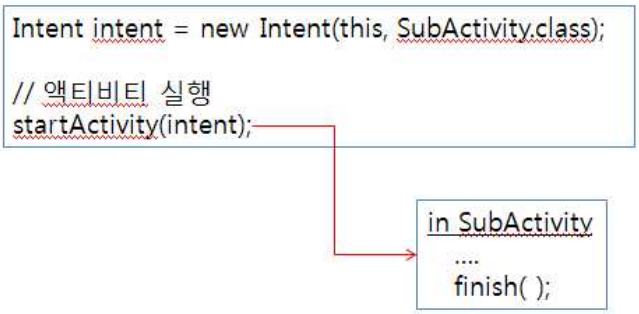
-전달 방법

■ startActivity() 혹은 startService() 등의 메소드를 호출

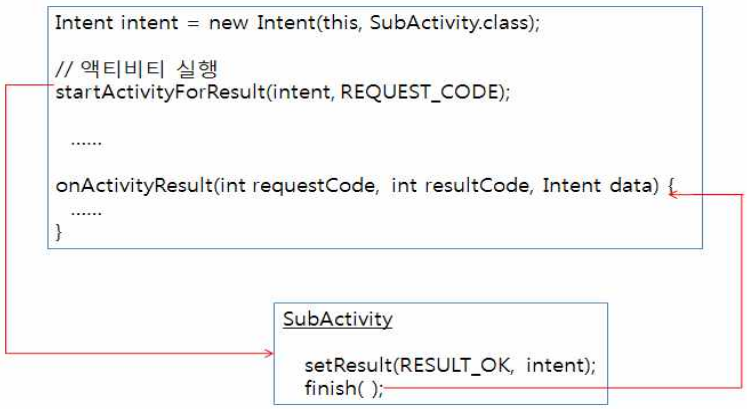
그림 8.9

8.3.2 결과값

-결과 값이 없는 경우



-결과 값이 있는 경우



8.4 암시적 인텐트

8.4.1 개요

-인텐트 라우팅 혹은 인텐트 확정

■ intent routing 혹은 intent resolving

■ 특정 인텐트가 주어지면 해당 인텐트의 의도에 가장 잘 부합하는 인텐트 필터를 가진 단 하나의 컴포넌트를 찾아주는 과정

■ 패키지 관리자

• 애플리케이션 컴포넌트의 인텐트 필터를 분석하여 수신하고자 하는 메시지의 내용을 수집하여 관리
• 분석한 필터의 내용을 바탕으로 최적의 액티비티를 찾아 액티비티 관리자에게 통보

■ 액티비티 관리자
• 수신한 인텐트에 가장 잘 부합하는 액티비티를 찾기 위하여 패키지 관리자에게 문의
• 인텐트를 수신해야할 대상 액티비티를 활성화

-인텐트 객체와 인텐트 필터의 관계

■ 인텐트는 다양한 형태로 존재
■ 예를 들어 전화와 관련된 인텐트의 경우: 인텐트를 보내는 컴포넌트는 전화 번호를 메시지로 사용. 인텐트를 받아야 할 컴포넌트는 다이얼을 돌리고 통화하는 애플리케이션
■ 메시지에 인텐트를 받아야 할 대상 컴포넌트가 지정되어 있지 않다면 안드로이드는 적절한 대상 컴포넌트를 찾아 활성화
■ 모든 애플리케이션은 안드로이드 매니페스트 파일의 인텐트 필터에 잠재적인 대상을 표시.

-인텐트 필터

■ 안드로이드 시스템 내부에서는 액티비티, 서비스, 그리고 방송수신자와 같은 애플리케이션 컴포넌트에 의하여 다양한 인텐트 발생 가능
■ 이와 같은 애플리케이션 컴포넌트는 자신들이 할 수 있는 일을 시스템에게 알리기 위하여 하나 이상의 인텐트 필터를 사용
■ 특정 인텐트를 수신하고자 하는 컴포넌트는 인텐트 필터를 사용하여 자신이 받고 싶은 인텐트 메시지를 정의
■ 인텐트 필터는 특정 의도를 가진 인텐트 메시지를 수신하겠다는 선언

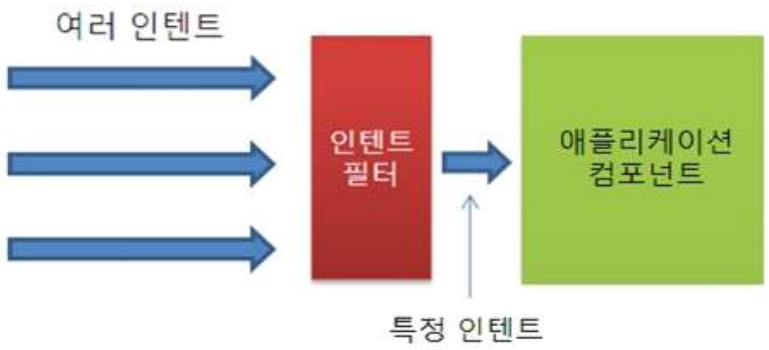


그림 8.10

-인텐트 필터의 예

```
01 <intent-filter android:label="@string/resolve_edit">
02    <action android:name="android.intent.action.VIEW" />
03    <action android:name="android.intent.action.EDIT" />
04    <category android:name="android.intent.category.DEFAULT"/>
05    <data android:mimeType="vnd.android.cursor.item/vnd.google.note" />
06 </intent-filter>
07 <intent-filter>
08    <action android:name="android.intent.action.INSERT" />
```

```
09    <category android:name="android.intent.category.DEFAULT" />
10    <data android:mimeType="vnd.android.cursor.dir/vnd.google.note" />
11 </intent-filter>
```

8.4.2 인텐트 객체

-의미

- 정보의 묶음
- 인텐트 객체는 인텐트를 수신하는 컴포넌트에게 관심 있는 정보의 묶음
- 수행되어야 할 액션과 처리할 데이터, 컴포넌트의 범주 등
- 대개의 경우 액션과 데이터만 있다면 충분. 경우에 따라 더 정확하고 상세한 처리를 위하여 추가적인 정보도 필요

-컴포넌트 이름

- 선택 사항
- 명시적으로 지정되면 인텐트 객체가 지정된 클래스의 객체로 전달
- 컴포넌트 이름이 있다면 호출 대상 컴포넌트가 고정되어 있어 인텐트 객체의 다른 정보는 별 의미가 없음
- 동일한 애플리케이션 내의 서브 액티비티를 호출할 때 주로 사용
- 다른 애플리케이션의 액티비티도 권한만 있다면 명시적으로 호출가능
- 컴포넌트 이름이 명시적으로 지정되지 않을 경우
  - 주로 다른 애플리케이션에 속한 컴포넌트를 호출할 때 사용
  - 안드로이드는 인텐트 객체의 다른 정보를 사용하여 가장 적합한 대상 컴포넌트를 탐색. 즉, 안드로이드는 모든 애플리케이션의 메니페스트 파일에 있는 인텐트 필터를 참조.

-대표적인 표준 액션

표 8.1

액션을 위한 상수	대상 컴포넌트	의미
ACTION_CALL	액티비티	통화 시작
ACTION_DIAL		전화 걸기
ACTION_EDIT		데이터를 편집하기 위하여 표시
ACTION_MAIN		태스크를 위한 초기 액티비티 시작. 입출력을 위한 데이터 없음
ACTION_PICK		데이터로부터 한 항목을 선택하기
ACTION_VIEW		보여주기
ACTION_SYNC		서버와 모바일 단말기의 데이터 동기화
ACTION_BATTERY_LOW	방송 수신자	배터리 부족
ACTION_HEADSET_PLUG		헤드셋 장비 접속/해제
ACTION_SCREEN_ON		화면 켜짐
ACTION_TIMEZONE_CHANGED		시간대timezone 변경

-데이터

■ 별도의 데이터가 필요 없는 액션도 있지만 많은 액션이 대응하는 데이터 명세를 가짐

- 액션이 ACTION_CALL 혹은 ACTION_DIAL이라면 데이터 필드는 통화하고자 하는 전화 번호인 tel:이라는 접두어가 있는 URI
- 액션이 ACTION_EDIT이라면 데이터 필드는 편집을 위하여 보여질 문서에 대한 URI
- 액션이 ACTION_VIEW이라면 데이터 필드는 http:를 포함한 URI

■ 데이터를 처리할 컴포넌트를 인텐트와 매치할 때를 URI 외에 데이터의 형식을 알 필요

• 범주

표 8.2

범주를 위한 상수	의미
CATEGORY_BROWSABLE	이미지 혹은 이메일과 같은 링크에 의하여 참조되는 데이터를 표시하기 위하여 브라우저에 의해 대상 액티비티를 안전하게 호출할 수 있다.
CATEGORY_DEFAULT	
CATEGORY_GADGET	액티비티는 가젯을 보유하고 있는 다른 액티비티에 내장될 수 있다.
CATEGORY_HOME	단말기를 켤 때나 HOME 키를 눌렀을 때 액티비티가 홈 화면을 보여준다.
CATEGORY_LAUNCHER	액티비티는 태스크의 시작 액티비티가 될 수 있으며 애플리케이션 런처에 표시된다.
CATEGORY_PREFERENCE	대상 액티비티가 설정 화면이다.

-엑스트라

■ 인텐트를 처리할 컴포넌트에 전달되어야 할 추가적인 정보

■ 키와 값의 쌍으로 구성

■ 주로 액티비티 사이에 매개변수와 반환값을 전달하는 도구로 사용

■ 키와 값의 쌍으로 정보를 전달하기 위한 메소드public Intent putExtra (String name, int value 혹은 각종 형식)public Intent putExtras (Bundle extras)

■ 엑스트라에 저장된 값을 읽기 위한 메소드public <type> get<Type>Extra (String name, <type> defaultValue)public String getStringExtra (String name)public Bundle getExtras ()

■ 번들을 위한 메소드

public <type> get<Type> (String key) public <type> get<Type> (String key, <type> defaultValue) public void put<Type> (String key, <type> value)

■ 보기

```
Intent i = new Intent();
i.putExtra("TEST", "Test string"); // String 추가

// 인텐트를 받은 다른 컴포넌트가 Extras 데이터를 받아옴
String str = getIntent().getExtras().getString("TEST");
```

8.5 인텐트 응용

8.5.1 명시적 인텐트

-예제

그림 8.11

8.5.2 명시적 인텐트

-예제

그림 8.11

8.5.3 메뉴와 인텐트

-예제

■ Intent3Demo와 동일한 결과

■ 다음 메소드 이용

• public abstract MenuItem setIntent (Intent intent)

제9장 그래픽

9.1 개요

9.1.1 이미지

-개요

■ 이미지를 출력할 때 레이아웃 위에서 View 객체를 사용

■ 간단한 이미지를 표현하는 애플리케이션인 경우 ImageView를 주로 사용

■ 애플리케이션에서 이미지는 해당 프로젝트의 res/drawable/ 폴더에 위치

-안드로이드 SDK 1.5 버전 이후

■ 해상도마다 별도의 폴더 - drawable-hdpi/, drawable-mdpi/, 그리고 drawable-ldpi/ - 를 제공

■ R 클래스로 생성될 때나 XML 파일에서는 -hdpi와 같은 접미어 없이 모두 res/drawable/ 폴더로 인식

-지원 형식

■ 거의 모든 형식의 이미지를 지원

■ PNG 형식이 안드로이드에 가장 적합

■ JPG 혹은 BMP 형식의 이미지를 사용해도 무방

■ GIF 형식의 이미지는 사용가능하지만 부적합

-참조

■ 폴더에 저장된 이미지는 소스 코드나 XML 파일에서 파일 확장자를 제외한 파일 이름만으로 참조

9.1.2 그래픽

-개요

■ 커스텀 2차원 그래픽 라이브러리와 고사양의 3차원 그래픽을 위한 OpenGL ES 1.0에 의하여 구동

■ android.graphics.drawable 패키지가 가장 일반적인 2차원 그래픽 API를 지원

■ OpenGL API는 크로노스 OpenGL ES 패키지와 안드로이드의 OpenGL 유틸리티에 의하여 유용

■ 프로젝트를 시작할 때 그래픽에 대한 정확한 요구를 살펴보는 것이 중요

■ 그래픽 작업은 다양한 테크닉이 요구. 예를 들어 다소 정적인 애플리케이션은 인터랙티브 게임이나 3차원 렌더링을 요구하는 애플리케이션과는 다른 차원의 그래픽과 애니메이션으로 구현

-2차원 그래픽 그리기

■ 레이아웃의 뷰 객체에 그래픽과 애니메이션을 그리기

• 뷰 안에 그래픽이 포함되도록 정의함에 따라 시스템의 뷰 계층 구조 그리기 절차에 의하여 처리

• 동적으로 변화가 없고 성능 집약적인 게임에 속하지 않는 간단한 그래픽에 적합

■ 그래픽을 캔버스Canvas에 직접 그리기

• 적절한 클래스의 draw() 메소드를 호출하는 것으로 개발자의 코드로 그리기를 처리

• 규칙적으로 스스로 다시 그리기를 해야 하는 애플리케이션에 적합

9.2 캔버스에 그리기

9.2.1 캔버스

-의미

■ 그래픽이 그려지는 실제 화면의 인터페이스로써 사용

■ 그림을 그릴 때 사용하는 화폭을 의미하며 화면 영역에 무엇인가를 그리는 수단을 제공

-기하학적 도형 그리기 메소드

■ public void drawPoint (float x, float y, Paint paint)

■ public void drawPoints (float[] pts, Paint paint)

■ public void drawLine (float startX, float startY, float stopX, float stopY, Paint paint)

■ public void drawLines (float[] pts, Paint paint)

■ public void drawCircle (float cx, float cy, float radius, Paint paint)

■ public void drawRect (float left, float top, float right, float bottom, Paint paint)

-기타 메소드

■ public void drawRoundRect (RectF rect, float rx, float ry, Paint paint)

■ public void drawOval (RectF oval, Paint paint)

■ public boolean clipRect (float left, float top, float right, float bottom)

■ public void drawARGB (int a, int r, int g, int b)

■ public void drawColor (int color)

■ public void drawRGB (int r, int g, int b)

■ public void drawText (String text, float x, float y, Paint paint)

9.2.2 페인트

-의미

■ 붓의 두께 혹은 물감의 종류 등과 같은 그리기에 대한 속성 정보를 가지는 클래스

-생성자

■ public Paint ()

■ public Paint (int flags)

■ public Paint (Paint paint)

-메소드

■ public void setARGB (int a, int r, int g, int b)

■ public void setAntiAlias (boolean aa)

■ public void setColor (int color)

■ public void setDither (boolean dither)

■ public void setStyle (Paint.Style style)

■ public void setTextSize (float textSize)

■ public Typeface setTypeface (Typeface typeface)

9.2.3 패스

-의미

■ 직선 조각, 2차 곡선 및 3차 곡선를 구성하는 도형의 궤적 정보를 가지는 그래픽 클래스

■ 복잡한 도형을 그릴 때 미리 도형의 경로를 구성할 수 있기 때문에 효율적

■ 도형의 궤적 정보만 가지기 때문에 화면에 직접적으로 보이지 않는다. 따라서 다음과 같은 메소드를 호출 public void drawPath (Path path, Paint paint)

-유용한 메소드

■ public void addCircle (float x, float y, float radius, Path.Direction dir)
■ public void addOval (RectF oval, Path.Direction dir)
■ public void addRect (RectF rect, Path.Direction dir)
■ public void addRoundRect (RectF rect, float[] radii, Path.Direction dir)
■ public void lineTo (float x, float y)
■ public void moveTo (float x, float y)
■ public void rLineTo (float dx, float dy)
■ public void rMoveTo (float dx, float dy)
■ public void reset ()

9.2.4 응용

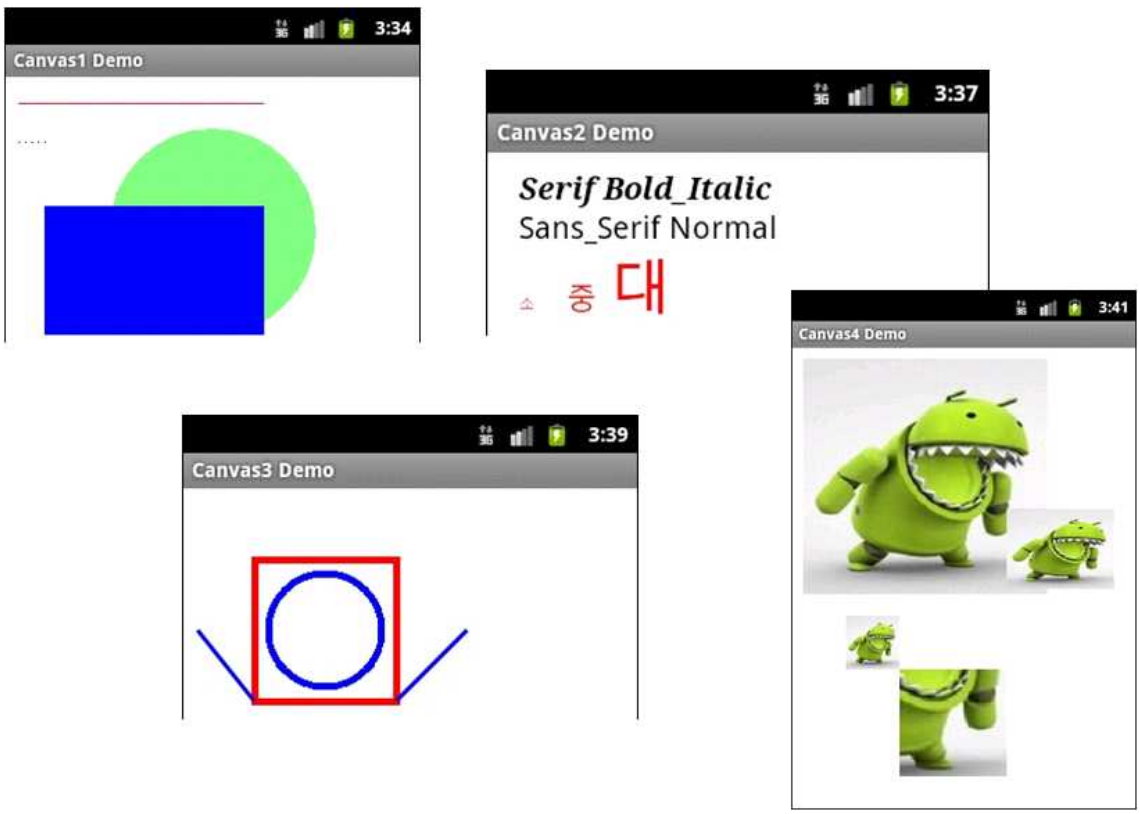


그림 9.1

9.3 뷰 객체에 그리기

9.3.1 Shader와 Drawable

-Shader

■ 도형의 표면을 자연스럽게 채우는 시각적 효과를 제공
■ 그래픽에 그래디언트gradient 효과를 설정하는 3개의 서브 클래스
  • LinearGradient
  • RadialGradient
  • SweepGradient

■ BitmapShader는 Shader의 서브클래스로써 비트맵을 도형의 표면에 반복적으로 표현

-Drawable
■ 그리기가 가능한 모든 것을 추상화한 클래스
■ 화면에 그리기 위하여 추출할 수 있는 리소스의 형식
■ 서브클래스
  • BitmapDrawable, ClipDrawable, ColorDrawable
  • ShapeDrawable, PictureDrawable, LayerDrawable 등
-ShapeDrawable
■ 동적으로 2차원 그래픽을 그리고자 하는 경우 적합
■ Shape 객체를 이용하여 도형 그리기에 적합
■ Shape 추상 클래스는 PathShape, RectShape, ArcShape, OvalShape, RoundRectShape 와 같은 파생 클래스

9.3.2 응용

-XML을 통한 ShapeDrawable 응용

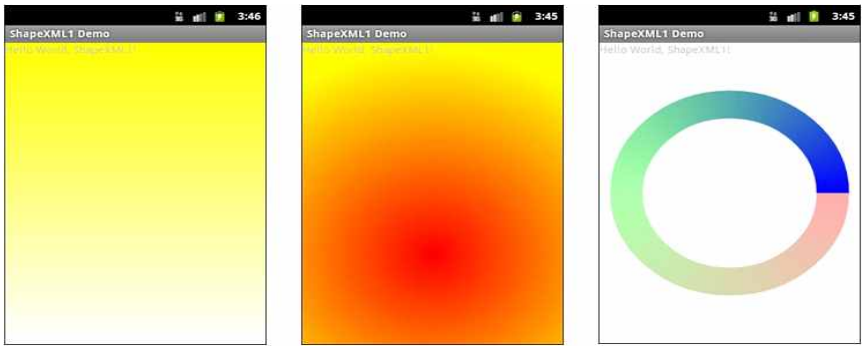


그림 9.2

-XML을 통한 ShapeDrawable 응용

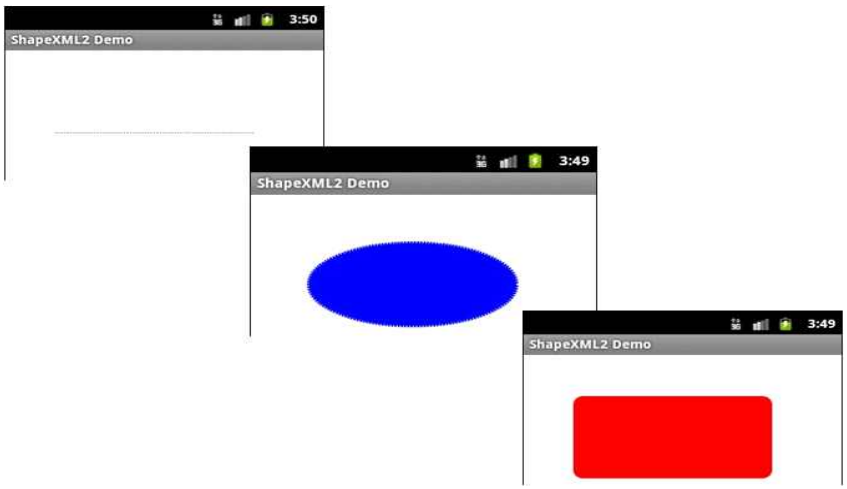


그림 9.3

-코드를 통한 ShapeDrawable와 Shader 이용

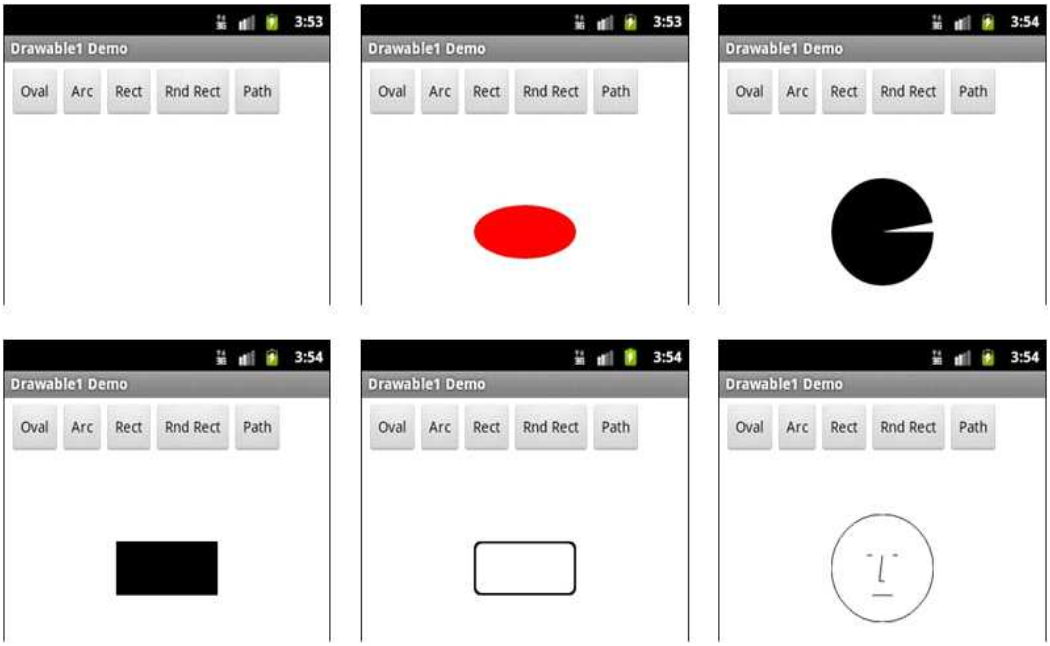


그림 9.4

-코드를 통한 ShapeDrawable와 Shader 이용

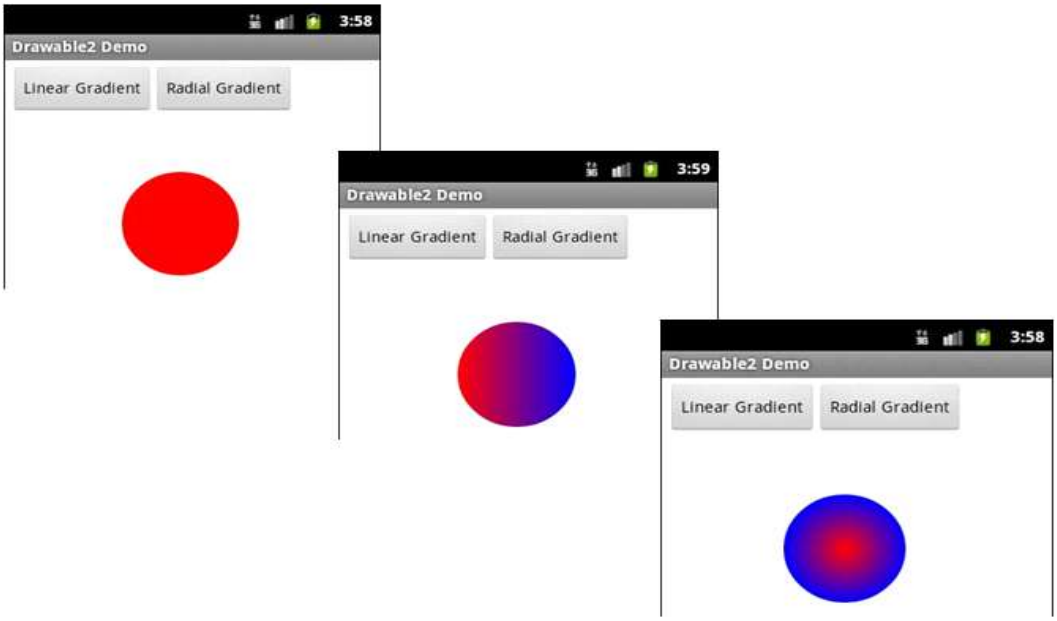


그림 9.5

9.4 위젯의 수정

9.4.1 개요

-안드로이드는 사용자 인터페이스를 위하여 세련되고 강력한 컴포넌트 기반의 모델을 제공
-미리 만들어진 다양한 위젯과 레이아웃 사용 가능
-안드로이드가 제공하는 위젯은 매우 보편적인 경우를 위한 기능과 모양
-커스텀 위젯 생성 방법
■기존 위젯의 도움이 없이 완전히 새롭게 생성하는 위젯
■기존 위젯이나 레이아웃을 확장하여 관련된 메소드를 재정의하여 만든 위젯
■다수의 위젯을 조합하여 만든 위젯

9.4.2 응용

-스마트한 에디트 텍스트

그림 9.6

-simple_list_item_1.xml의 변형

그림 9.7

제10장 스레드와 애니메이션

10.1 스레드

10.1.1 개요

-프로세스

■ 애플리케이션 프로그램의 실행 → 적어도 하나의 리눅스 프로세스가 동작

■ 프로세스는 주소 공간이나 레지스터와 같은 자원을 독립적으로 보유하는 실행중인 프로그램을 의미

■ 프로세스는 항상 CPU를 사용하지 않을 뿐만 아니라 또한 프로세스가 증가하면 시스템의 오버헤드도 증가

-스레드

■ CPU를 효과적으로 사용하고 다수의 프로세스에 의한 오버헤드를 줄이기 위하여 스레드 사용

■ 스레드는 프로세스에서 독자적인 명령어 제어권을 가지며 프로세스에 포함된 자원을 공유하는 프로그램의 실행 단위를 의미

■ 스레드를 지원하지 않는 시스템은 프로세스를 CPU의 스케줄링 단위로 사용하지만 스레드를 지원하는 시스템은 스레드를 스케쥴링 단위로 사용

■ 부분의 운영체제는 스레드를 지원

-안드로이드와 스레드

■ 자바에서 제공하는 스레드를 지원

■ 안드로이드 애플리케이션은 기본적으로 하나의 스레드를 가진 프로세스에 의하여 실행

■ 애플리케이션을 실행하면 안드로이드는 자동적으로 메인 스레드, 즉 UI 스레드를 생성

■ 메인 스레드는 사용자와 상호작용하기 위한 이벤트를 책임지기 때문에 매우 중요 → 사용자와 상호작용 외의 오랜 시간을 필요로 하는 작업을 처리하려면 별도의 스레드를 사용하는 것이 바람직

-스레드 생성자

■ public Thread ()

■ public Thread (Runnable runnable)

-스레드 사용법

■ Thread 클래스의 파생 클래스를 생성하고 run() 메소드를 재정의

■ Runnable 객체를 생성한 후 Thread 객체를 제공. Runnable은 run()이라는 추상 메소드를 가진 간단한 인터페이스
→ 두 경우 모두 run() 메소드를 정의해야 하며, run() 메소드는 스레드에서 수행할 코드를 포함하며 진입점 역할을 수행

10.1.2 핸들러

-의미

■ 스레드 사이에 메시지를 보내고 처리하기 위하여 사용하는 통신 메커니즘

■ 스레드는 프로세스와 달리 자원을 공유하기 때문에 두 개의 스레드가 동시에 하나의 자원을 수정한다면 동기화 문제가 발생

-스레드의 역할 분담

■ 메인 스레드: 사용자 인터페이스의 변경 사항을 전담

■ 서브 스레드: 시간 소모적인 작업을 담당

-핸들러의 역할

■ 두 개의 스레드 사이에 작업 내용 교환하게 함으로서 동기화 문제를 해결

-메시지

■ 생성된 핸들러 객체는 자신을 생성한 스레드 및 메시지 큐와 밀접하게 연관

■ 스레드가 메시지를 전달하면 메시지 큐에 적재

■ 메시지란 UI 등에서 실제로 처리를 담당하는 스레드로 데이터를 전송하거나 작업을 요청하기 위하여 전달하는 객체

■ 메시지가 도착하면 핸들러의 다음 메소드를 통하여 메시지를 수신가능public void handleMessage (Message msg)

■ Message 객체의 멤버 필드

- What: 사용자가 정의할 수 있는 메시지 코드로써 수신자가 메시지를 식별하기 위하여 사용
- arg1과 arg2: 정수형으로 된 정보를 저장하기 위한 변수
- obj: 수신자에게 보낼 수 있는 임의의 객체로써 Object 형식으로 정수 형식 이외의 데이터를 전달할 때 유용
- replyTo: 기타

-메시지 생성자

■ public Message ()

■ public static Message obtain ()

■ public static Message obtain (Handler h, int what, int arg1, int arg2)

■ public static Message obtain (Handler h, int what, int arg1, int arg2, Object obj)

-핸들러가 제공하는 스레드 사이에 메시지를 통신하기 위한 메소드

■ public final boolean post (Runnable r)

■ public final boolean postDelayed (Runnable r, long delayMillis)

■ public final boolean sendEmptyMessage (int what)

■ public final boolean sendMessage (Message msg)

10.1.3 스레드 응용

-2개의 추가적인 스레드를 생성하여 정수가 점차 다른 속도로 증가하는 멀티스레드 작업

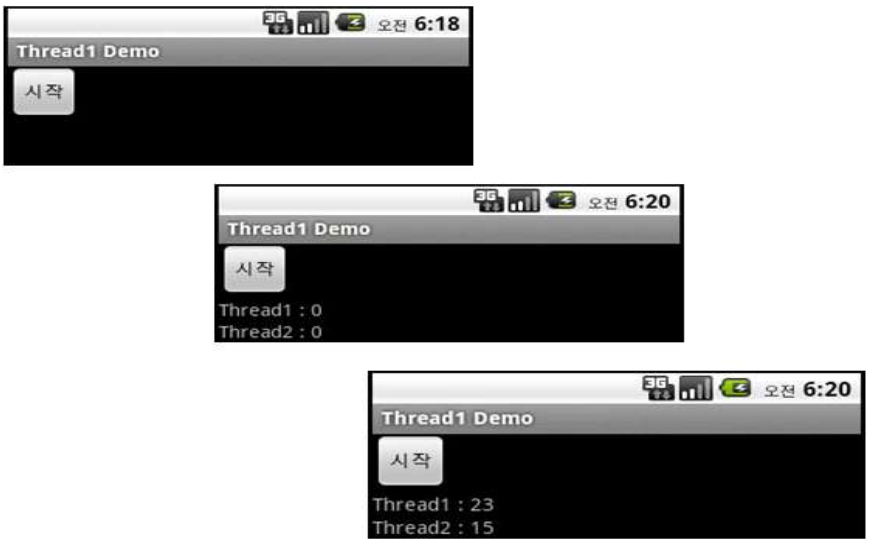


그림 10.1

-Runnable 인터페이스를 사용하면?

-스레드를 이용한 진행 다이어로그

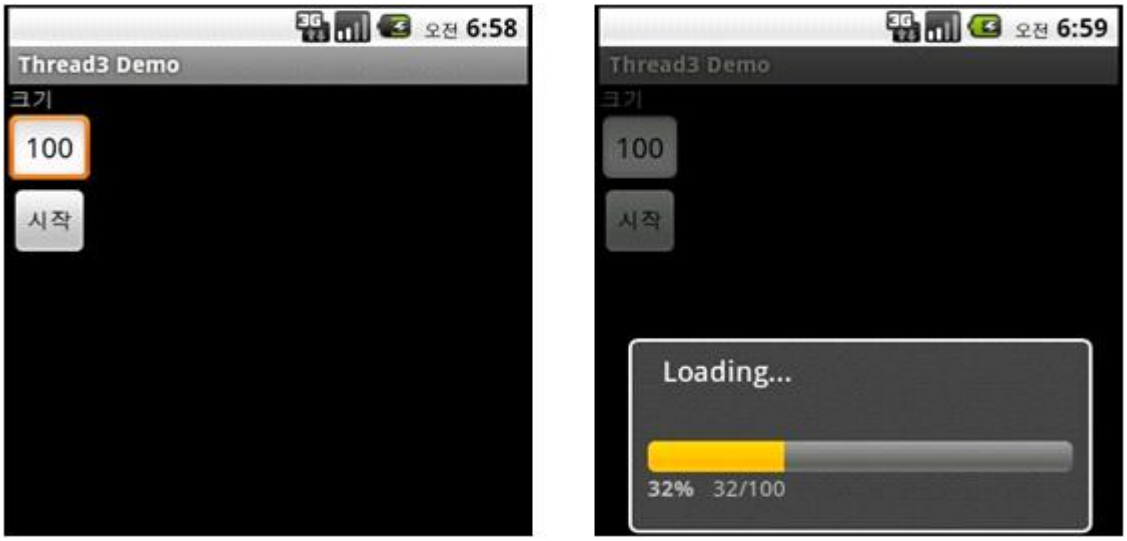


그림 10.2

10.1.4 루퍼

-의미

- 일반적으로 스레드가 보내는 메시지는 메시지 큐에 저장되고 순차적으로 처리
- 안드로이드 시스템은 메시지 큐를 관리하지만 메시지 큐에 저장된 메시지를 꺼내어 핸들러로 전달하지는 않는다.
- 루퍼(Looper)를 제공하여 메시지 큐와 핸들러 사이에서 메시지를 전달하는 작업을 담당
- UI 스레드인 메인 스레드는 기본적으로 루퍼를 소유
- 작업 스레드는 루퍼가 없음. 작업 스레드가 메시지 큐에서 메시지를 꺼내어 처리하려면 루퍼를 생성하여 동작하도록 해야 함

-전형적 루퍼 사용법

```
class LooperThread extends Thread {
    public Handler mHandler;
    public void run() {
        Looper.prepare();
        mHandler = new Handler() {
            public void handleMessage(Message msg) {
                // process incoming messages here
            }
        };
        Looper.loop();
    }
}
```

-루퍼 응용

■ 메인 스레드와 작업 스레드가 양방향으로 메시지를 통신하는 예제

■ 메인 스레드가 화면에서 입력한 수를 작업 스레드가 수신하여 제곱 연산을 수행

■ 작업 스레드가 연산 결과를 메인 스레드에게 다시 보내면 메인 스레드는 연산 결과를 화면에 출력

그림 10.3

10.2 애니메이션

10.2.1 프레임 애니메이션

-개요

■ 전통적인 애니메이션

■ 약간씩 다른 일련의 이미지를 필름의 롤처럼 차례대로 재생시킴으로써 이미지를 자연스럽게 움직이도록 만드는 기법

■ 인간의 잔상 효과를 이용

■ 상용 시스템인 경우에는 초당 24~30개의 프레임을 출력

■ 인터레이스 방식을 사용하는 시스템에서는 초당 60개의 프레임을 출력하여 고품질의 애니 메이션을 구현

■ 애니메이티드 GIF도 본질적으로는 프레임 애니메이션에 해당

■ 프레임의 수에 따라 용량 문제 발생. 그러나 단순하기 때문에 프레임 애니메이션은 코드로 정의하기 복잡한 그래픽 변형에 적합

-기본 클래스는 AnimationDrawable

-유용한 메소드

■ public void addFrame (Drawable frame, int duration)

■ public void setOneShot (boolean oneShot)

■ public void start ()

■ public void stop ()

-구현 방법

■ 자바 소스 코드로 구현: AnimationDrawable 클래스의 API를 사용하여 애니메이션 프레임을 정의하며 조금 복잡

■ XML 파일을 정의하여 구현: 애니메이션을 구성하는 각 프레임을 나열하는 XML 파일을 사용하기 때문에 간단. 애니메이션을 위한 XML 파일은 res/anim/ 폴더에 저장

-응용

■ 여러 개의 이미지를 가진 애니메이션

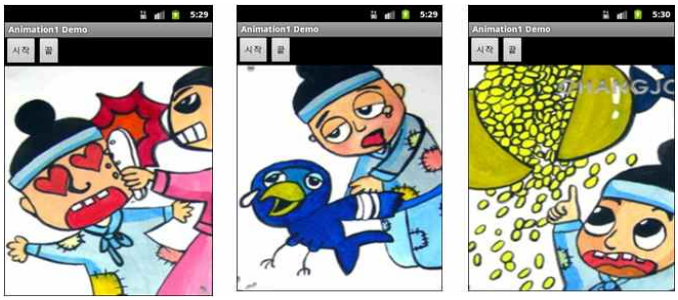


그림 10.4

10.2.2 트윈 애니메이션

-개요

■ 비트맵, 도형, 텍스트 같은 그래픽 객체나 임의의 뷰 위젯을 대상으로 이동, 확대 · 축소, 회전 등에 대한 변환을 통해 애니메이션 효과를 얻는 기법

■ 트랜지션 애니메이션이라고도 불림

■ 객체와 변환 방식을 설정한 후 수학의 보간법 계산을 통해 다음 프레임을 생성

■ CPU 사용량 ↑, 파일의 용량 ↓

■ 변환의 대상은 View 객체이므로 텍스트, 배경 이미지 등 모든 종류가 가능

■ XML 파일 혹은 자바 소스로도 코딩 가능

■ XML 파일로 정의하면 읽기 쉽고, 재사용 및 교체 가능

-기본 클래스는 Animation 클래스

■ 뷰, 서피스, 그리고 다른 객체에 적용될 수 있는 애니메이션을 추상화한 것

■ 관련된 하위 클래스로는 AlphaAnimation, AnimationSet, RotateAnimation, ScaleAnimation, TranslateAnimation

■ 속성과 메소드

표 10.1

속성	관련된 메소드	의미
detachWallpaper	setDetachWallpaper(boolean)	윈도우 애니메이션 옵션
duration	setDuration(long)	애니메이션 수행 시간
fillAfter	setFillAfter(boolean)	애니메이션 완료 후 변환 적용
fillBefore	setFillBefore(boolean)	애니메이션 완료 전 변환 적용
fillEnabled	setFillEnabled(boolean)	fillAfter 적용
interpolator	setInterpolator(interpolator)	애니메이션 보간 적용
repeatCount	setRepeatCount(int)	애니메이션 반복 횟수
repeatMode	setRepeatMode(int)	애니메이션 행동 정의
startOffset	setStartOffset(long)	애니메이션 수행 지연 시간(1/1,000초)
zAdjustment	setZAdjustment(int)	애니메이션 Z 순서 조정

-기본 클래스는 Animation 클래스

■관련된 상수

표 10.2

상수 이름	의미
ABSOLUTE	픽셀 개수
INFINITE	애니메이션 무한 실행
RELATIVE_TO_PARENT	부동소수점 수로서 현재 객체의 부모 객체의 폭 혹은 높이와 곱해진다.
RELATIVE_TO_SELF	부동소수점 수로서 현재의 애니메이션 객체의 폭 혹은 높이와 곱해진다.
RESTART	애니메이션이 마지막에 도달하고 INFINITE_REPEAT이거나 양수면 애니메이션은 처음부터 시작
REVERSE	애니메이션이 마지막에 도달하고 INFINITE_REPEAT이거나 양수면 애니메이션은 역방향으로 진행
START_ON_FIRST_FRAME	애니메이션 시작 시간
ZORDER_BOTTOM	애니메이션 Z 순서의 영향을 받도록 조정
ZORDER_NORMAL	애니메이션이 현재 Z 순서로 유지되도록 한다
ZORDER_TOP	애니메이션이 다른 내용의 최상위가 되도록 한다.

-응용

■정사각형에 대하여 성장, 이동 및 회전 변환을 수행

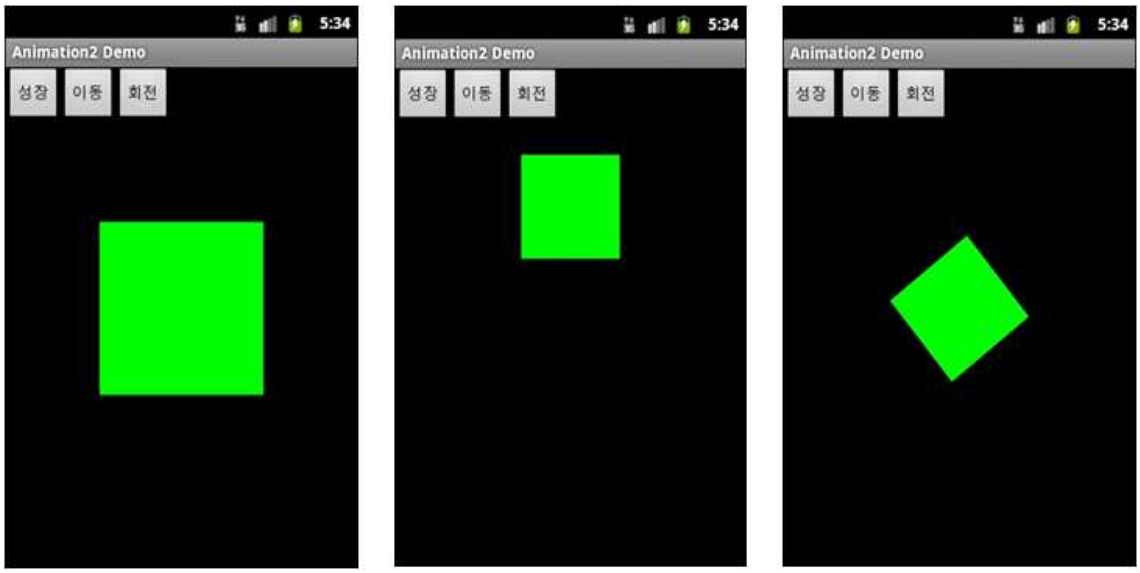


그림 10.5

10.3 서피스뷰

10.3.1 개요

-의미

■ 일반적으로 애플리케이션의 뷰는 메인 스레드에서 그려지며, 또한 사용자의 상호작용도 동일한 메인 스레드에서 수행

■ 문제점

- 복잡한 애니메이션 작업과 같은 렌더링 코드가 오랫동안 메인 스레드를 잡고 있다면 사용자와 상호작용 불가
- 운영체제는 작업 스레드에게 화면 리소스의 접근을 불허

■ 안드로이드는 서피스뷰를 제공하여 해결

■ 서피스뷰를 이용하면 작업 스레드가 화면 리소스의 접근을 위한 준비를 가능

■ 서피스뷰는 뷰 계층구조 내에서 드로잉 전용 공간을 제공하는 특별한 서브 클래스

-애플리케이션을 개발하려면 SurfaceView 클래스를 확장한 후 SurfaceHolder.Callback이라는 인터페이스를 구현

-SurfaceHolder.Callback이라는 인터페이스는 다음 메소드 포함

■ 서피스의 생성, 변화, 그리고 소멸에 대한 이벤트를 알려주는 메소드

■ public abstract void surfaceCreated (SurfaceHolder holder)

■ public abstract void surfaceChanged (SurfaceHolder holder,

■ int format, int width, int height)

■ public abstract void surfaceDestroyed (SurfaceHolder holder)

10.3.2 응용

-화면에 터치하는 곳마다 안드로이드 아이콘이 복사되는 예제

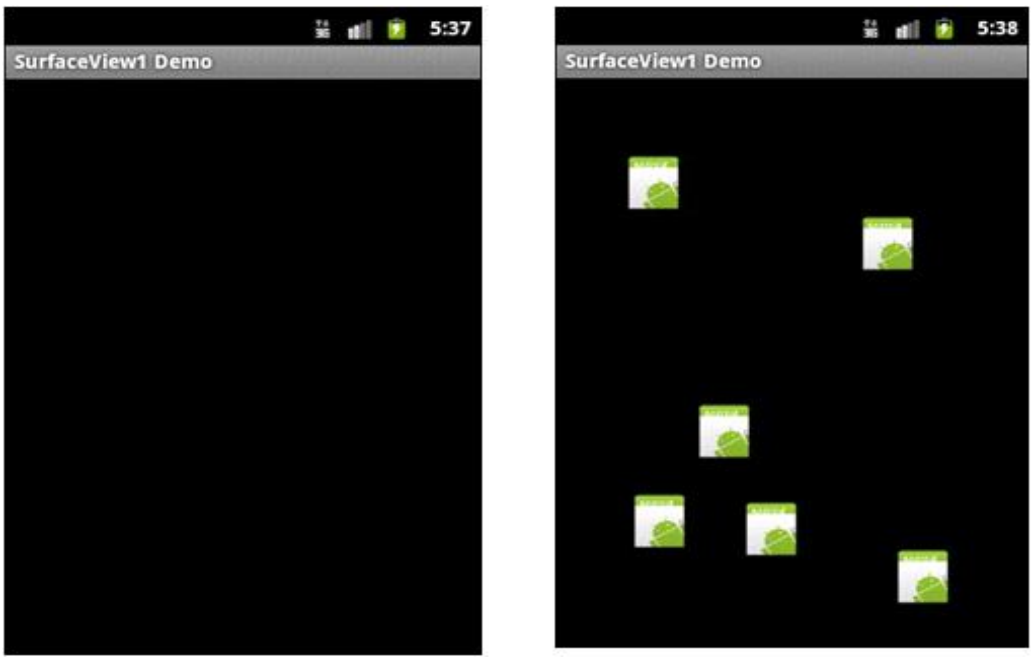


그림 10.6

제11장 데이터 관리

11.1 프레퍼런스

11.1.1 개요

-의미

■액티비티의 생명주기에 의하면 액티비티가 메모리에서 제거될 경우 사용자 인터페이스 상태가 사라짐

■프레퍼런스는 자동 로그인 혹은 비밀번호 저장 등과 같은 화면 이동 사이에 데이터 손실을 방지하기 위하여 주로 사용

■각 애플리케이션에 고유한 환경 설정에 대한 정보를 보관하기 위한 목적으로 주로 사용되며 대부분 소량의 정보로 구성

-프레퍼런스의 기본 데이터 형식

■키를 사용하여 값을 저장하고 가져오는 일반적인 프레임워크

■어떤 형식 - boolean, float, int, long, 그리고 string - 의 기본적인 데이터라도 SharedPreferences 클래스를 사용하여 저장 가능

■프레퍼런스를 이용하여 저장한 데이터는 사용자 세션 사이에서 영속적

-애플리케이션을 위한 프레퍼런스를 얻기 위한 메소드

■public abstract SharedPreferences getSharedPreferences (String name, int mode)

■public SharedPreferences getPreferences (int mode)

-프레퍼런스 파일의 위치

■data/data/<package_name>/shared_prefs/<preference_name>.xml

-프레퍼런스에 데이터 저장

■SharedPreferences.Editor 인터페이스를 구하여 edit() 메소드를 호출

■Editor는 영속적 저장소인 SharedPreferences에 모든 변경 사항을 commit() 메소드를 호출할 때 모아서 한꺼번에 적용

■데이터를 추가하기 위하여 putBoolean() 혹은 putString() 메소드를 사용

-유용한 메소드

■public abstract SharedPreferences.Editor putBoolean (String key, boolean value)

■public abstract SharedPreferences.Editor putInt (String key, int value)

■public abstract SharedPreferences.Editor putLong (String key, long value)

■public abstract SharedPreferences.Editor putFloat (String key, float value)

■public abstract SharedPreferences.Editor putString (String key, String value)

■public abstract boolean commit ()

■public abstract boolean getBoolean (String key, boolean defValue)

■public abstract int getInt (String key, int defValue)

■public abstract long getLong (String key, long defValue)

■public abstract float getFloat (String key, float defValue)

■public abstract String getString (String key, String defValue)

11.1.2 응용

-예제

■에디트텍스트와 체크박스 위젯을 가진 액티비티에 적절한 설정

■액티비티를 강제로 종료

■재실행하면 이전에 취한 설정 자국 유지?

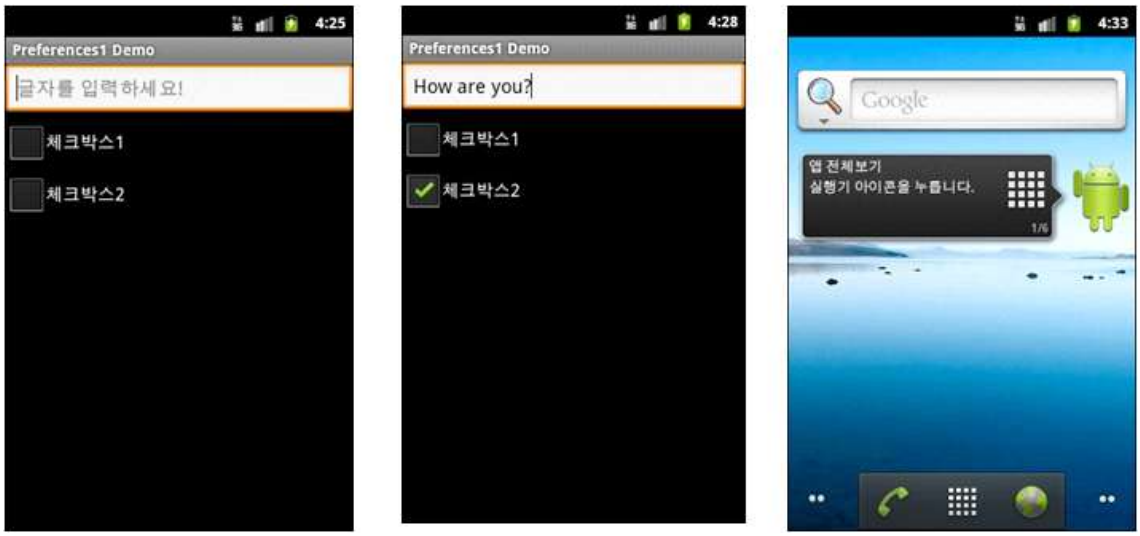


그림 11.1

-예제

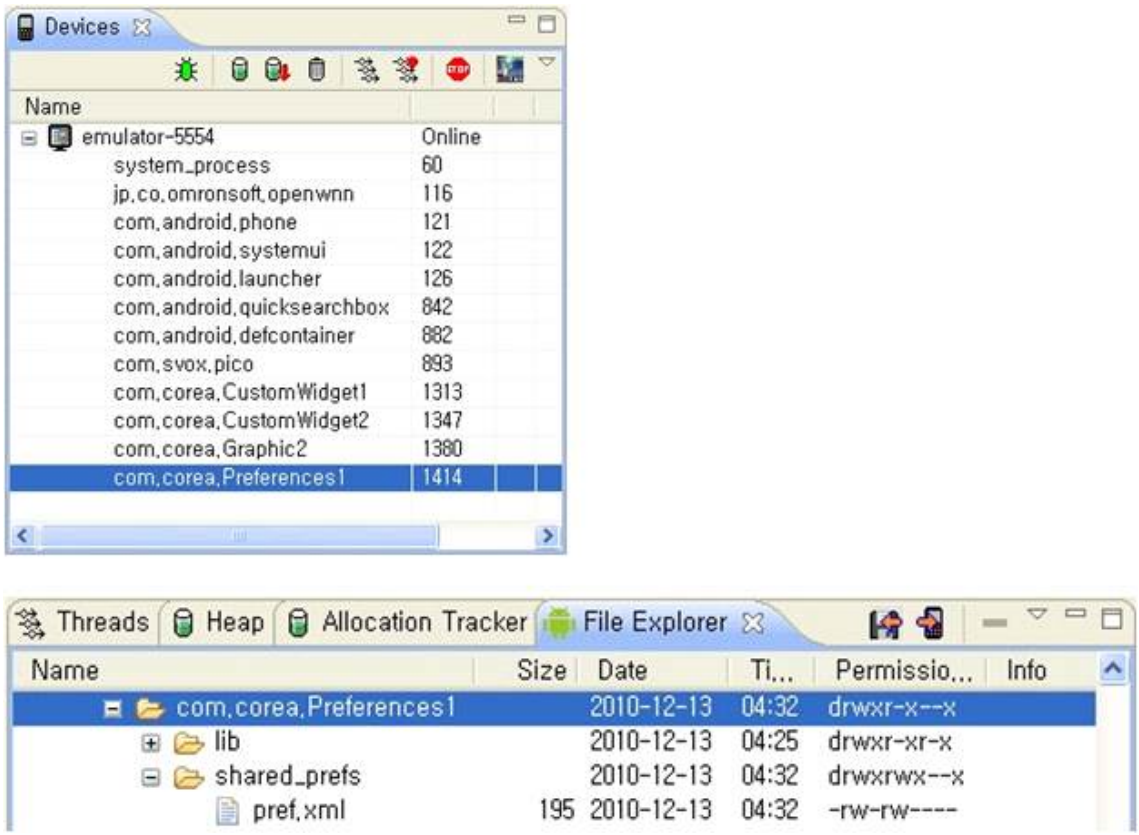


그림 11.2

11.2 데이터베이스

11.2.1 개요

-의미
■ 안드로이드는 SQLite라는 내장형 데이터베이스를 제공
■ 파일과 마찬가지로 DB는 기본적으로 생성된 애플리케이션에 종속
■ 가벼운 파일 기반인 SQLite DB는 내장형 디바이스에 적합
■ SQLite DB는 DB의 생성 및 오픈과 버전 관리를 담당하는 SQLiteOpenHelper 클래스와 DB에 대한 데이터의 추가 · 삭제 · 수정 및 질의 작업을 담당하는 SQLiteDatabase 클래스를 제공
■ 일반적으로 사용할 DB에 적합하도록 DB 어댑터를 생성한 후 어댑터를 통하여 DB를 관리
-DB의 위치
■ data/data/<package_name>/databases/<데이터베이스이름>.유
-DB 어댑터의 구성
■ 일반적으로 생성자
■ SQLiteOpenHelper 클래스를 확장한 내부 클래스
■ DB의 열기 및 닫기 메소드
■ 레코드의 추가 · 삭제 · 수정 및 질의 작업을 수행하는 메소드
-SQLiteOpenHelper 클래스의 확장
■ 생성자를 구현
■ 또한 상속된 다음 2개의 추상 메소드를 구현
  • public abstract void onCreate (SQLiteDatabase db)
  • public abstract void onUpgrade (SQLiteDatabase db, int oldVersion, int newVersion)

11.2.2 레코드 접근
-데이터베이스의 목적
■ 궁극적으로 레코드를 추가 · 삭제 · 수정
■ 질의한 결과인 레코드 내용을 화면에 출력
  • ContentValues와 Cursor

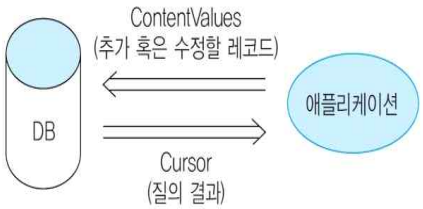


그림 11.3

-레코드의 추가 및 갱신
■ ContentValues는 레코드의 갱신 대상 필드의 값을 준비하기 위하여 사용하는 클래스
■ ContentValues의 빈 객체를 생성한 후 다음과 같은 put() 메소드를 사용하여 레코드를 추가 혹은 갱신할 준비
  • public void put (String key, Integer value)
  • public void put (String key, byte[] value)
  • public void put (String key, String value)
■ 메소드
  • public long insert (String table, String nullColumnHack, ContentValues values)
  • public int update (String table, ContentValues values,

• String whereClause, String[] whereArgs)
• public int delete (String table, String whereClause, String[] whereArgs)

-레코드의 질의
■ 커서 객체는 질의에 의하여 반환
■ 결과의 레코드 집합에 대한 무작위 접근을 제공
■ 메소드
• public Cursor query (String table, String[] columns, String selection, String[] selectionArgs, String groupBy, String having, String orderBy)
■ Cursor 인터페이스가 무작위 접근을 위하여 제공하는 메소드
• public abstract int getColumnCount ()
• public abstract int getCount ()
• public abstract int getInt (int columnIndex)
• public abstract int getPosition ()
• public abstract String getString (int columnIndex)
• public abstract boolean moveToFirst ()
• public abstract boolean moveToLast ()
• public abstract boolean moveToNext () 등

-커서와 관련된 어댑터
■ 커서를 어댑터에 바인딩한 후 어댑터뷰를 통하여 출력 가능
■ 안드로이드는 커서가 지정하는 열 혹은 필드를 텍스트뷰와 같은 위젯에 사상하는 SimpleCursorAdapter 라는 어댑터를 제공
■ 생성자
public SimpleCursorAdapter (Context context, int layout, Cursor c, String[] from, int[] to)

11.2.3 응용
-예제
■ 간단한 메모를 할 수 있는 note라는 DB를 생성

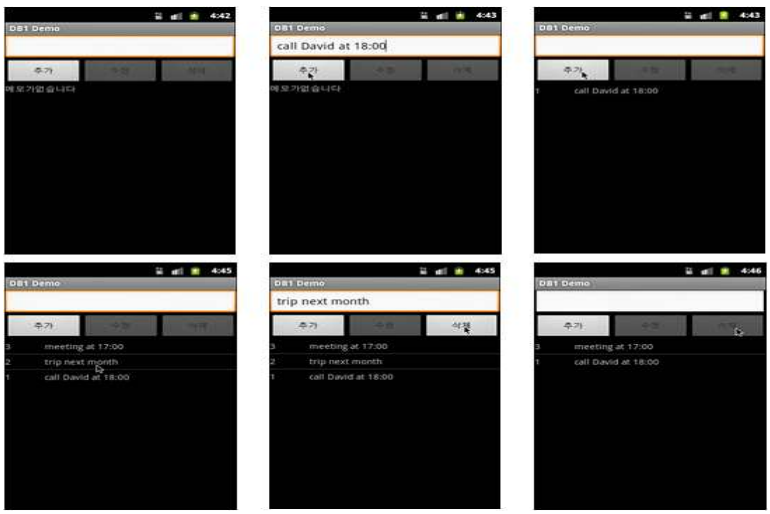


그림 11.4

11.3 컨텐트 공급자

11.3.1 개요

-의미

■ 안드로이드에서는 기본적으로 모든 애플리케이션은 다른 애플리케이션과 격리되어 실행

■ 따라서 하나의 애플리케이션에 포함된 데이터베이스 혹은 파일시스템과 같은 컨텐트 모델을 다른 애플리케이션에서 접근 불가

■ 컨텐트 공급자는 애플리케이션 사이에 컨텐트 모델을 공유하기 위한 메커니즘

■ 애플리케이션이 외부 컨텐트 모델을 접근하려면 컨텐트 리졸버가 필요

■ 컨텐트 공급자는 애플리케이션에 속한 컨텐트 모델을 외부에 노출시키는 역할을 수행

-안드로이드 애플리케이션과 파일 시스템

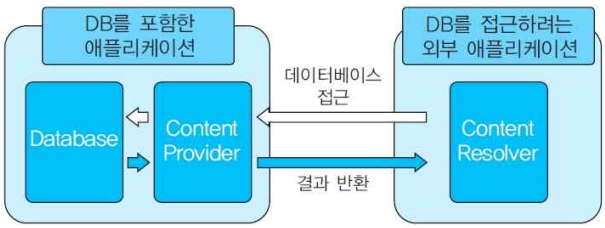


그림 11.5

-데이터 집합의 식별

■ 하나의 공적인 URI를 이용

■ URI 형식 <schema>://<authority>/<data_path>/<id>

■ URI 예 content://browser/bookmarkscontent:// contacts/peoplecontent:// contacts/people/3

11.3.2 내장 컨텐트 공급자

-android.provider 패키지

-대표적인 내장 컨텐트 공급자의 URI

표 11.1

컨텐트	URI
통화 로그	CallLog.Calls.CONTENT_URI
연락처 리스트	Contacts.People.CONTENT_URI
브라우저 북마크	Browser.BOOKMARKS_URI
브라우저 검색 이력	Browser.SEARCHES_URI
전화번호	Phone.CONTENT_URI
시스템 설정 값	Settings.System.CONTENT_URI
내장 미디어 이미지	MediaStore.Image.Media.INTERNAL_CONTENT_URI
외부 미디어 이미지	MediaStore.Image.Media.EXTERNAL_CONTENT_URI
내장 미디어 동영상	MediaStore.Video.Media.INTERNAL_CONTENT_URI
외부 미디어 동영상	MediaStore.Video.Media.EXTERNAL_CONTENT_URI
내장 미디어 오디오	MediaStore.Audio.Media.INTERNAL_CONTENT_URI
외부 미디어 오디오	MediaStore.Audio.Media.EXTERNAL_CONTENT_URI

-응용

■ 내장 컨텐트 공급자를 이용하여 전화번호를 접근한 후 데이터를 리스트뷰에 출력

그림 11.6

11.3.3 맞춤 컨텐트 공급자

-개요

■ 대부분의 컨텐트 공급자는 데이터베이스를 사용하지만 어떤 파일에 대해서도 가능

■ 애플리케이션에 속한 데이터베이스를 외부에 노출하려면 먼저 ContentProvider를 확장한 맞춤 컨텐트 공급자를 생성

■ ContentProvider의 6개 추상 메소드

- getType(), delete(), insert(), update(), query(), 그리고 onCreate()

■ 컨텐트 공급자의 주소인 URI와 데이터의 형식인 MIME 형식을 명시

-URI

■ 메소드

- public abstract List<String> getPathSegments ()
- public static Uri parse (String uriString)

■ UriMatcher

- URI 처리를 위하여 문자열로 구성된 정보를 분석하여 미리 분석한 후 정수 코드로 변환
- public UriMatcher (int code)
- public void addURI (String authority, String path, int code)
- public int match (Uri uri)

-MIME 형식

■ vnd.android.cursor.item/vnd.기관명.컨텐트형식

■ vnd.android.cursor.dir/vnd.기관명.컨텐트형식

-애플리케이션 컴포넌트 등록

■ 컨텐트 공급자의 authority를 안드로이드 시스템에 등록 필요

■ 매니페스트 파일의 <provider> 엘리먼트를 사용하여 애플리케이션 컴포넌트로 등록

■ 필요하다면 퍼미션도 추가

-응용
■ 메모장 처리를 위한 DB를 컨텐트 공급자로 외부에 노출

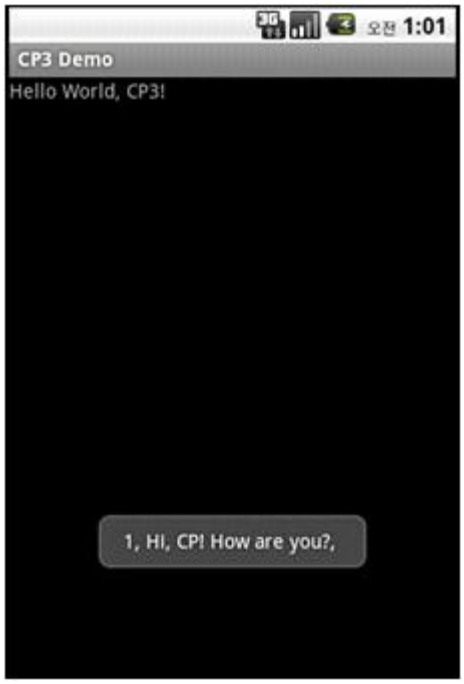


그림 11.7

제12장 노트패드

12.1 SQLite3 명령어

12.1.1 개요

-ASH
■ ADB가 포함하고 있는 쉘
■ 기본적인 리눅스 명령어를 사용 가능
■ DOS 명령창 → ADB 명령 → 다음 명령adb -s <에뮬레이터> shell
-SQLITE3 DB로 이동
■ # cd /data/data/com.corea.DB1/databases
■ # ls
■ memos.db
■ # sqlite3 memos.유
■ sqlite>
-SQLITE3 명령
■ 일반적인 SQL 명령
■ .help, .quit, .databases, .tables .schema
■ .help와 .quit

12.1.2 응용

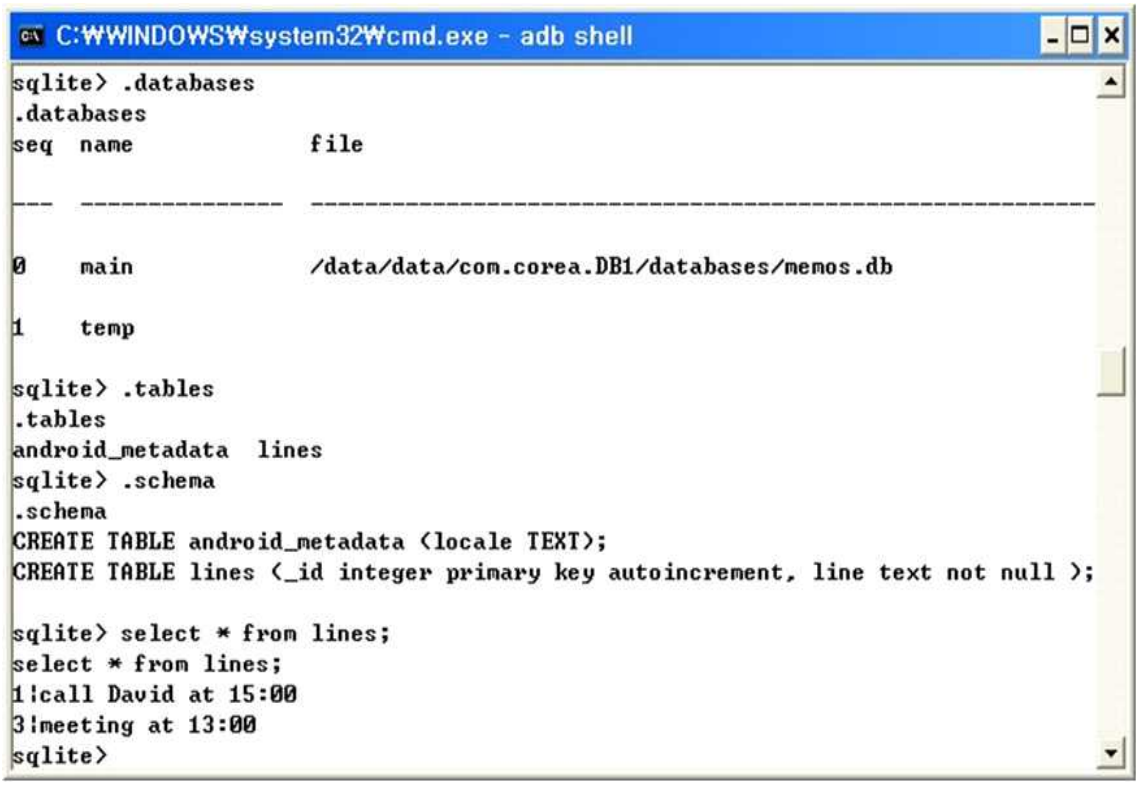


그림 12.1

12.2 노트패드

12.2.1 개요

-안드로이드는 개발자를 위하여 3가지 버전의 노트패트 프로젝트를 제공

-준비 사항

■ 안드로이드 투토리얼 사이트의 예제 코드 중 노트패트 프로젝트 연습용으로 보관된 파일 (http://developer.android.com/resources/tutorials/notepad/index.html)을 다운로드

■ 다운로드한 파일을 적절한 폴더에 압축을 해제한 후 푼다

■ NotepadCodeLab 폴더를 연다

-6개의 프로젝트 파일

■ Notepadv# 프로젝트는 연습용

■ Notepadv#Solution 프로젝트는 솔루션

12.2.2 Notepadv1

-포함된 내용

■ 리스트뷰의 기본 내용과 메뉴 옵션의 생성과 처리

■ 노트를 저장하기 위하여 SQLite 데이터베이스의 사용

■ SimpleCursorAdapter를 사용하여 데이터베이스 커서를 리스트뷰에 바인딩

■ 화면 레이아웃, 액티비티 메뉴에 메뉴 항목 추가 및 선택된 메뉴 항목의 처리

-프로젝트 구조

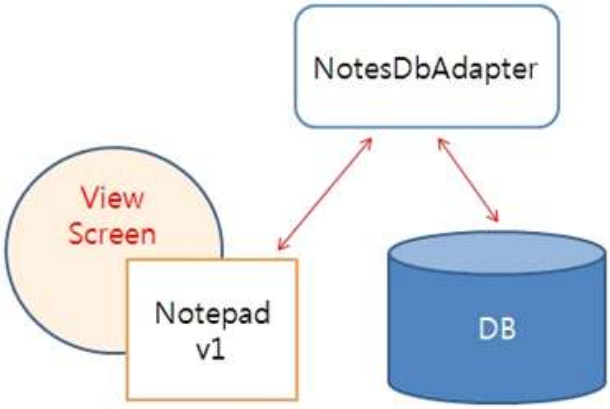


그림 12.2

■ 1개의 액티비티 Notepadv1으로 구성
■ 사용자가 직접 입력하는 내용이 아니라 자동적으로 생성된 문자열을 노트로 사용
■ 노트를 편집이나 삭제할 수 없고 단지 추가만 가능
■ DB 접근을 위한 도우미 클래스인 NotesDbAdapter를 이용하여 Notepadv1 클래스에서 생성한 노트를 저장

-실행 결과

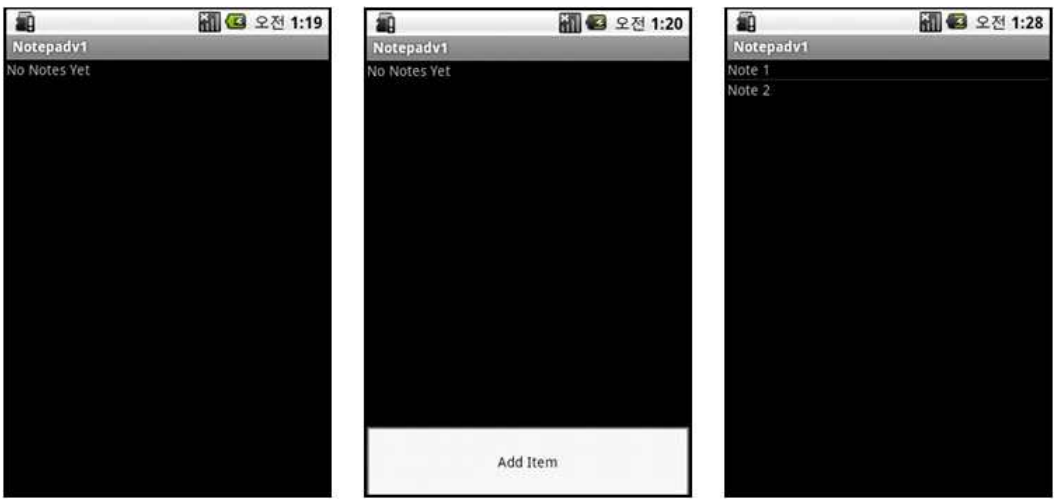


그림 12.3

12.2.3 Notepadv2

-포함된 내용

■ 새로운 액티비티를 생성하고 안드로이드 매니페스트 파일에 추가
■ startActivityForResult() 메소드를 사용하여 비동기적으로 다른 액티비티를 호출
■ Bundle 객체를 사용하여 액티비티 사이에 데이터 전달
■ 좀더 고급스러운 화면 레이아웃의 사용
■ 컨텍스트 메뉴의 생성

-프로젝트 구조

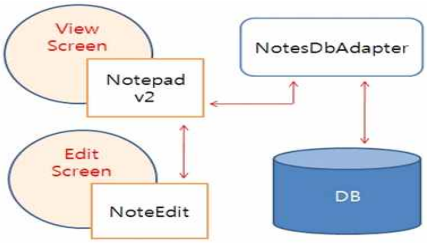


그림 12.4

■ 2개의 액티비티로 구성
■ Notepadv1 프로젝트와는 달리 사용자가 직접 노트를 생성
■ NoteEdit 클래스는 편집을 담당
■ Notepadv2 클래스는 DB 접근을 위한 도우미 클래스인 NotesDbAdapter를 이용하여 사용자가 편집한 노트를 관리

-추가 사항
■ 2개의 액티비티를 구분하기 위한 상수와 삭제 메뉴을 위한 상수 선언
■ 필요 없는 전역 변수 mNoteNumber 제거와 전역 변수로 사용할 커서 선언
■ mNoteNumber 변수 제거에 따른 fillData() 메소드의 수정
■ 컨텍스트 메뉴를 위한 onCreateContextMenu()와 onContextItemSelected() 메소드의 원형 추가
■ createNote() 메소드의 코드를 수정하기 위하여 기존 본문 코드 제거
■ 리스트뷰의 항목을 선택할 때 발생하는 이벤트를 처리하기 위한 콜백 메소드인 onListItemClick()의 원형 추가
■ 다른 액티비티에서 돌아온 경우 처리할 콜백 메소드인 onActivityResult()의 원형 추가

-실행 결과

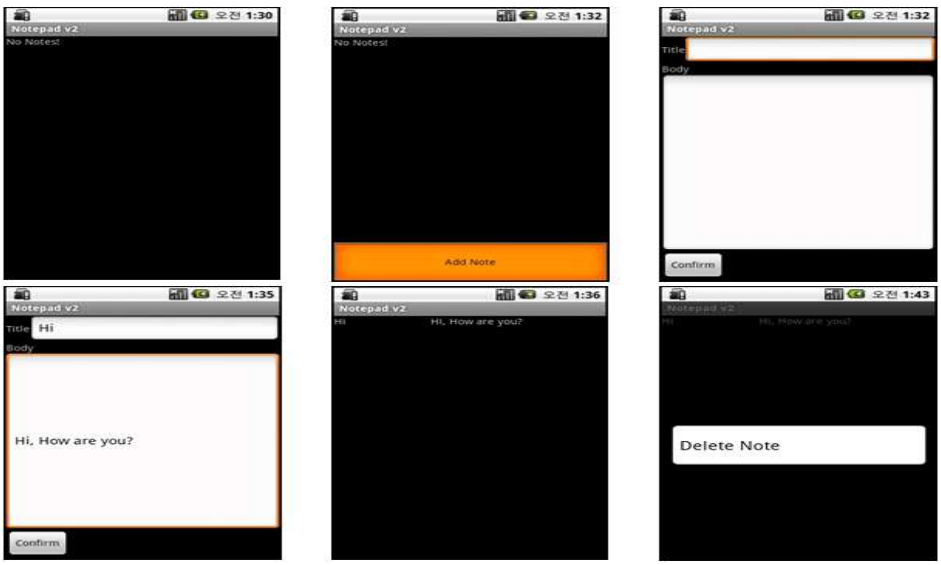


그림 12.5

12.2.4 Notepadv3

- Notepadv2의 오류

■ 취소 버튼을 클릭하면 오류 창 발생

-해결 방안

■ 노트를 생성하고 편집하는 모든 코드를 NoteEdit 액티비티로 이동

■ NoteEdit 액티비티가 NotesDbAdapter와 직접 인터페이스하며 편집 과정의 생명주기 동안에 애플리케이션의 상태를 유지

-프로젝트 구조

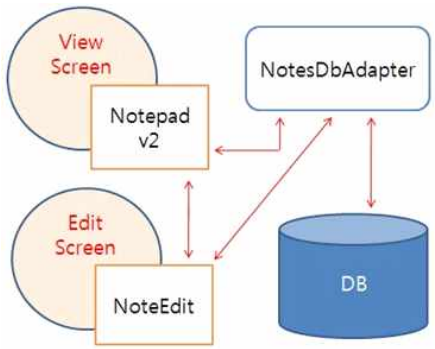


그림 12.6

제13장 웹뷰와 구글 지도

13.1 웹뷰

13.1.1 개요

■ 의미

■ WebView 클래스는 웹페이지를 전시하기 위한 뷰

■ 액티비티 내부에 있는 컨텐트를 웹으로 보여주는 기본 클래스

■ 웹페이지를 전시하기 위하여 안드로이드에 내장된 WebKit 렌더링 엔진을 사용하며 히스토리 항해, 확대 및 축소, 텍스트 검색 등을 위한 메소드를 포함

-액티비티가 인터넷을 접근하려면 권한 필요

■ 매니페스트 파일에 다음과 같은 권한을 추가<uses-permission android:name="android.permission.INTERNET" />

-웹뷰와 URL

■ 웹뷰가 URL을 받으면 일반적으로 안드로이드의 디폴트 브라우저로 하여금 URL에 명시된 웹페이지를 화면에 출력

■ 액티비티의 인터페이스인 웹뷰에 웹페이지가 출력되는 것이 아님.

■ 액티비티는 안드로이드의 디폴트 웹 브라우저에 의하여 중단되기 때문에 웹페이지를 제어 불가

■ 액티비티가 URL 요청을 처리하려면 WebViewClient 클래스가 제공하는 다음 메소드를 재정의public boolean shouldOverrideUrlLoading (WebView view, String url)

-유용한 메소드

■ public WebSettings getSettings ()
■ public void loadUrl (String url)
■ public void setWebViewClient (WebViewClient client)
■ public void loadDataWithBaseURL (String baseUrl, String data, String mimeType, String encoding, String historyUrl)
■ public boolean canGoBack ()
■ public boolean canGoForward ()
■ public void goBack ()
■ public void goForward ()
■ public void clearHistory ()

13.1.2 응용

-예제

■ 웹뷰 클래스를 이용하여 간단한 로컬 웹페이지, assets 폴더 아래에 있는 html 파일, 그리고 외부 URL을 검색

그림 13.1

-예제

■ URL 요청을 호스트 애플리케이션에서 처리

그림 13.2

13.2 맵뷰와 맵액티비티

13.2.1 구글 지도 API 키

-MD5 지문 획득

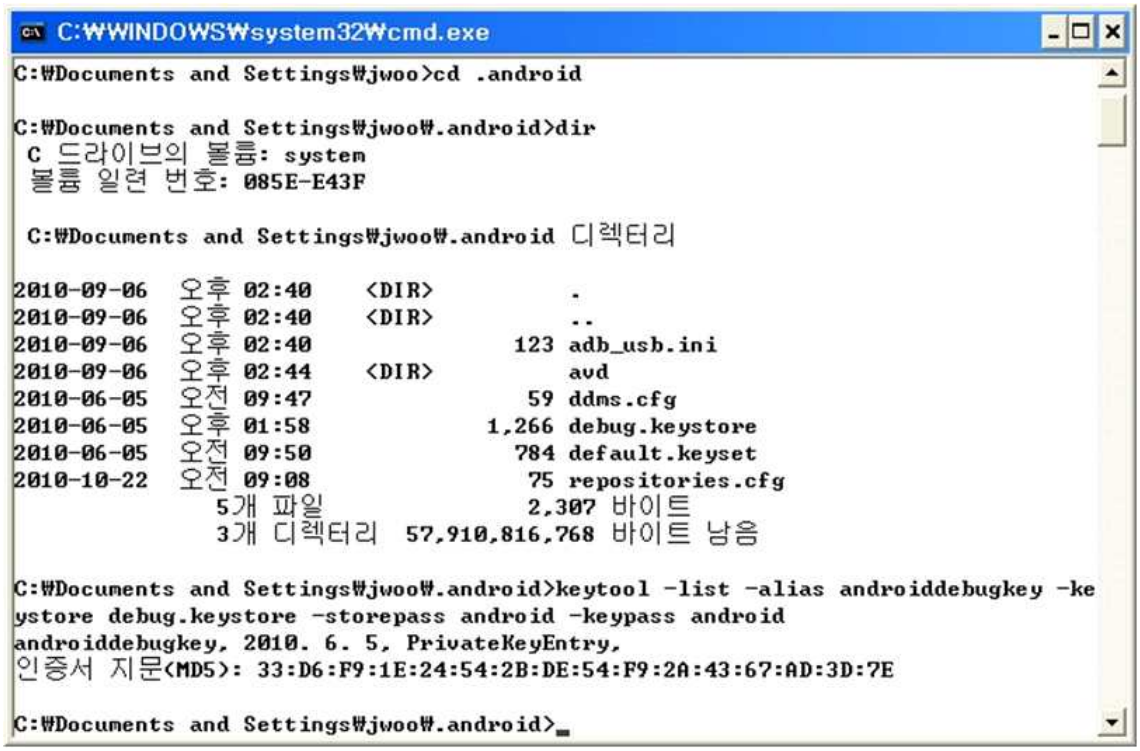


그림 13.3

-API 키 생성

■ 구글 이메일 계정을 준비

■ http://code.google.com/android/maps-api-signup.html에 MD5 지문을 등록

■ 약관에 동의한 후 MD5 지문을 입력하고 'Generate API Key' 버튼을 클릭 → 구글 메일 계정과 암호를 입력

그림 13.4

13.2.2 지도 그리기

-MapActivity와 MapView

■ com.google.android.maps 패키지에 포함

■ MapActivity는 Activity 클래스를 확장한 서브클래스 - 추상 메소드인 isRouteDisplayed()를 포함

■ MapView 클래스가 제공하는 유용한 메소드

- public void displayZoomControls(boolean takeFocus)
- public int getZoomLevel()
- public void setBuiltInZoomControls(boolean on)
- public void setSatellite(boolean on)
- public boolean isSatellite()
- public void setStreetView(boolean on)
- public boolean isStreetView()
- public MapController getController()

-MapController

■ 지도의 이동 혹은 축소 및 확대를 관리하기 위한 유틸리티 클래스

■ 유용한 메소드

- public void animateTo(GeoPoint point)
- public void setCenter(GeoPoint point)
- public int setZoom(int zoomLevel)
- public boolean zoomIn()
- public boolean zoomOut()

-프로젝트의 생성

■ 빌드 타겟 중 반드시 Google APIs에 해당하는 타겟 이름을 선택

■ 매니페스트 파일에 퍼미션과 라이브러리를 추가

```
~~~ 생략 ~~~
<uses-library android:name="com.google.android.maps"/>
</application>
<uses-sdk android:minSdkVersion="9" />
<uses-permission android:name="android.permission.INTERNET"/>
```

13.2.3 응용

-예제

그림 13.5

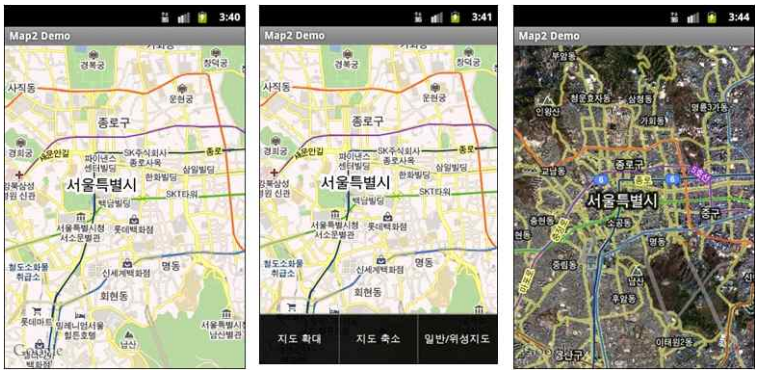


그림 13.6

13.3 위치 기반 서비스

13.3.1 개요

-LBS

■ Location Based Services

■ 모바일 디바이스에서 제공할 수 있는 가장 매력적인 기능중의 하나

■ 모바일 사용자의 현재 위치 파악 및 사용자의 움직임 추적

-안드로이드 LBS 애플리케이션

■ GPS와 안드로이드 네트워크 위치 공급자를 이용하여 사용자의 위치를 확보

■ LBS는 android.location 패키지와 외부 라이브러리인 구글 지도를 통하여 제공

■ LBS를 위하여 안드로이드에서 제공하는 주요 클래스는 LocationManager와 LocationProvider

-LocationManager

■ 시스템이 제공하는 위치 서비스에 대한 접근을 제공

■ 사용자의 현재 위치 감지, 움직임 추적, 또한 지정된 영역에 대한 근접 경보 설정 가능

■ 인스턴스 생성

```
LocationManager lm =
(LocationManager)getSystemService(Context.LOCATION_SERVICE);
```

■ 주요 메소드

- public void addProximityAlert (double latitude, double longitude, float radius,
- long expiration, PendingIntent intent)
- public String getBestProvider (Criteria criteria, boolean enabledOnly)
- public List<String> getProviders (boolean enabledOnly)
- public Location getLastKnownLocation (String provider)
- public void requestLocationUpdates (String provider, long minTime,
- float minDistance, LocationListener listener 혹은 PendingIntent intent)

-Location 클래스가 제공하는 메소드

■ public float distanceTo (Location dest)

■ public double getSomething1 ()

■ public boolean hasSomething2 ()

■ Something1: Altitude, Latitude, Longitude

■ Something1 대신에 Accuracy, Bearing, Extras, Speed, Time 등을 사용할 수 있지만 반환형이 다름

■Something2: Latitude와 Longitude를 제외한 Something1 대신에 사용할 수 있는 것

13.3.2 응용

-예제

■모바일 디바이스 사용자의 위치를 출력

그림 13.7

-모바일 디바이스 사용자의 움직임에 대한 위치 정보를 출력

그림 13.8

-예제

■모바일 디바이스 사용자를 위한 위치공급자를 모두 출력하고 가장 양호한 위치공급자를 사용하여 사용자의 위치 정보를 출력

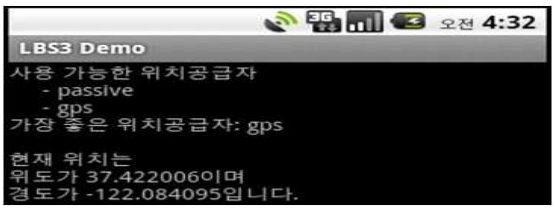


그림 13.9

13.4 지도 응용

13.4.1 지오코딩

-의미

■거리 주소나 위치 정보를 위도와 경도 좌표로 상호 변환하는 과정을 의미

■순방향 지오코딩: 주어진 거리 정보, 즉 거리 이름이나 주소 등을 사용하여 위도 및 경도 좌표를 구하는 것

■역방향 지오코딩은 위도 및 경도 좌표를 거리 정보로 변환

-Geocoder 생성자

■ public Geocoder (Context context, Locale locale)

■ public Geocoder (Context context)

-Geocoder 클래스가 제공하는 메소드

■ public List<Address> getFromLocation (double latitude, double longitude, int maxResults)

■ public List<Address> getFromLocationName (String locationName, int maxResults)

■ public List<Address> getFromLocationName (String locationName, int maxResults, double lowerLeftLatitude, double lowerLeftLongitude, double upperRightLatitude, double upperRightLongitude)

-Address 클래스가 제공하는 메소드

■ public String getAddressLine (int index)

■ public int getMaxAddressLineIndex ()

■ public String getCountryName ()

■ public String getFeatureName ()

■ public String getLocality ()

■ public String getThoroughfare ()

■ public double getLatitude ()

■ public double getLongitude ()

-예제: 순방향 지오코딩

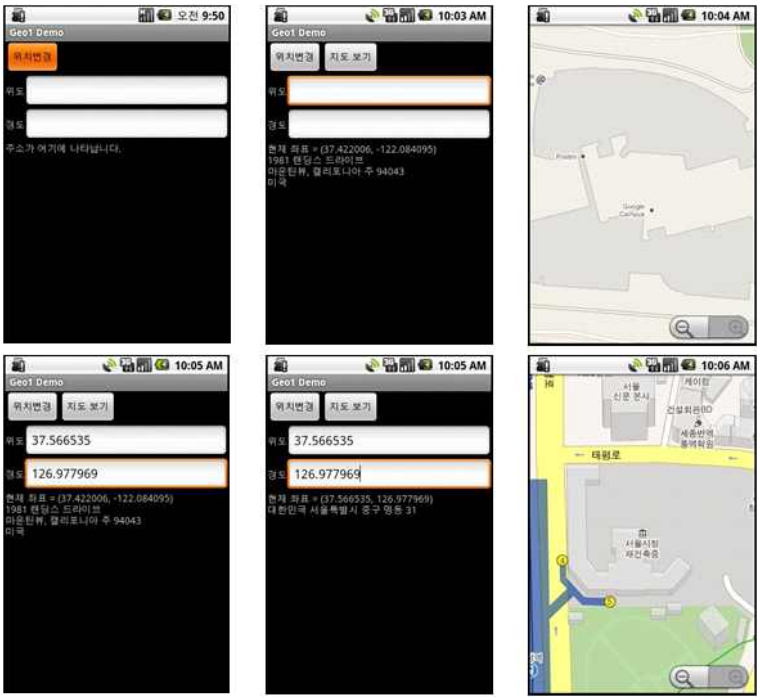


그림 13.10

13.4.2 오버레이

-의미

■ 안드로이드는 지도에 위치를 표시하기 위하여 오버레이를 제공
■ 오버레이는 맵뷰에 주석이나 마커를 추가해주는 com.google.android.maps 패키지에 포함된 클래스
■ 오버레이 클래스는 맵의 상단에 표시될 레이어를 나타내는 기본 클래스

-오버레이 추가

■ 오버레이의 서브클래스를 정의한 후 인스턴스를 생성
■ MapView.getOverlays() 메소드로부터 얻은 리스트에 생성한 인스턴스를 추가
■ 실제 지도에서 작업을 할 땐 오버레이의 서브클래스를 주로 이용
  • MyLocationOverlay: 현재의 위치와 방향을 지도위에 나타냄
  • ItemizedOverlay: 마커를 지도위에 나타냄

-지도 위에 마커를 표시

■ ItemizedOverlay의 서브클래스를 생성
■ ArrayList<OverlayItem>를 선언
■ addOverlay() 메소드를 정의
■ ItemizedOverlay의 서브클래스를 구현한 후 이를 이용하기 위한 액티비티 애플리케이션을 코딩
■ 액티비티 내에서 에 ItemizedOverlay의 서브클래스의 객체를 생성
■ 추가하고자 하는 오버레이 항목에 대한 객체를 생성한 후 ItemizedOverlay의 서브클래스의 addOverlay() 메소드를 사용하여 추가
■ 최종적으로 MapView.getOverlays() 메소드로부터 얻은 리스트에 생성한 인스턴스를 추가

-예제

■ 지도 위에 마커를 표시

■ 지도의 임의의 위치에 클릭하면 대응하는 지리적 좌표를 출력

그림 13.11

제14장 서비스 및 방송 수신자

14.1 노티피케이션

14.1.1 개요

-의미

■전화 수신 상태, 충전기의 상태 등과 같은 시스템의 변화에 대한 정보를 사용자에게 알려줘야 한다.
■토스트 기능은 잠시동안 메시지를 화면에 표시
■노티피케이션은 백그라운드 프로세스를 이용하며 사용자와 통신할 수 있는 방법 중의 하나로 자주 사용
■노티피케이션은 처음에 잠깐 보이다가 사용자가 확인할 때까지 아이콘을 상태바의 왼쪽에 둠
■노티피케이션은 소리, 진동 혹은 깜박임 등 사용 가능

-노티피케이션과 상태바

그림 14.1

14.1.2 사용 방법

-노티피케이션 관리자 획득

```
NotificationManager mNotificationManager = (NotificationManager) Context.getSystemService (Context.NOTIFICATION_SERVICE);
```

-노티피케이션 객체 생성

```
int icon = R.drawable.notification_icon;CharSequence tickerText = "Hello“;long when = System.currentTimeMillis();Notification notification = new Notification(icon, tickerText, when);
```

-노티피케이션의 확장 메시지와 인텐트 정의

```
Context context = getApplicationContext();CharSequence contentTitle = "My notification";CharSequence contentText = "Hello World!";Intent notificationIntent = new Intent(this, MyClass.class);PendingIntent contentIntent = PendingIntent.getActivity(this, 0, notificationIntent, 0);notification.setLatestEventInfo(context, contentTitle, contentText, contentIntent);
```

-노티피케이션을 노티피케이션 관리자에게 전달

```
private static final int HELLO_ID = 1;
mNotificationManager.notify(HELLO_ID, notification);
```

-노티피테이션 속성

표 14.1

속성	의미
sound	통지할 때 재생할 사운드
number	상태 바에 겹쳐서 출력될 숫자
ledARGB	표시할 LED 색상
ledOnMS	LED가 켜져 있는 시간 (단위: ms)
ledOffMS	LED가 꺼지는 시간 (단위: ms)
icon	상태 바에 표시할 아이콘의 id
contentView	확장된 상태 바 안에 표시하는 뷰
defaults	기본적인 설정으로 지정
flags	통지의 동작 방식을 지정
vibrate	진동 방식을 지정

-PendingIntent

■바로 실행되는 인텐트가 아니라 보류된 혹은 위임된 인텐트

■현재 실행되고 있는 애플리케이션 컴포넌트가 PendingIntent를 활성화하는 것이 아니라 다른 컴포넌트에 의하여 활성화

■일반 인텐트와 달리 애플리케이션에게 실행 권한을 위임

■만약 노티피케이션과 함께 사용된다면 현재 화면이 무엇이든 상관없이 상태바를 아래로 드래그할 때까지 인텐트의 활성화를 보류

■생성한 애플리케이션이 종료되더라도 다른 프로세스에 의하여 사용 가능

14.1.2 응용

-예제

■하나는 간단하게 Hi 메시지를 자신에게 보내는 경우이며, 다른 하나는 메시지를 보낼 뿐만 아니라 인텐트를 통하여 다른 액티비티 화면을 통하여 통지 내용을 표시

그림 14.2

14.2 방송 수신자

14.2.1 개요

-방송

■시스템에 변화가 발생할 때 신호를 보내 주는 방식을 의미

■애플리케이션이 방송에 반응하면 다양한 변화 및 제어가 가능

■전화가 오거나 문자 메시지를 받았을 경우 처리하는 것이 대표적인 보기

-방송 수신자

■안드로이드 애플리케이션 컴포넌트 중의 하나 → 매니페스트 파일에 등록 필요

■사용자와 직접 상호작용하지 않고 방송에 대한 수신만을 대기

■BroadcastReceiver라는 추상 클래스를 상속하고 onReceive() 추상 메소드를 구현해야 함.

■onReceive() 메소드는 방송수신자가 방송 메시지를 수신하면 안드로이드가 호출하는 콜백 메소드

-Context 클래스에 의하여 제공하는 방송 메소드

■public abstract void sendBroadcast (Intent intent)

■public abstract void sendBroadcast (Intent intent, String receiverPermission)

■public abstract void sendOrderedBroadcast (Intent intent, String receiverPermission)

14.2.2 응용

-예제

■단말기의 부팅이 완료되면 방송수신자가 감지하여 오늘의 날짜를 출력하는 액티비티를 실행

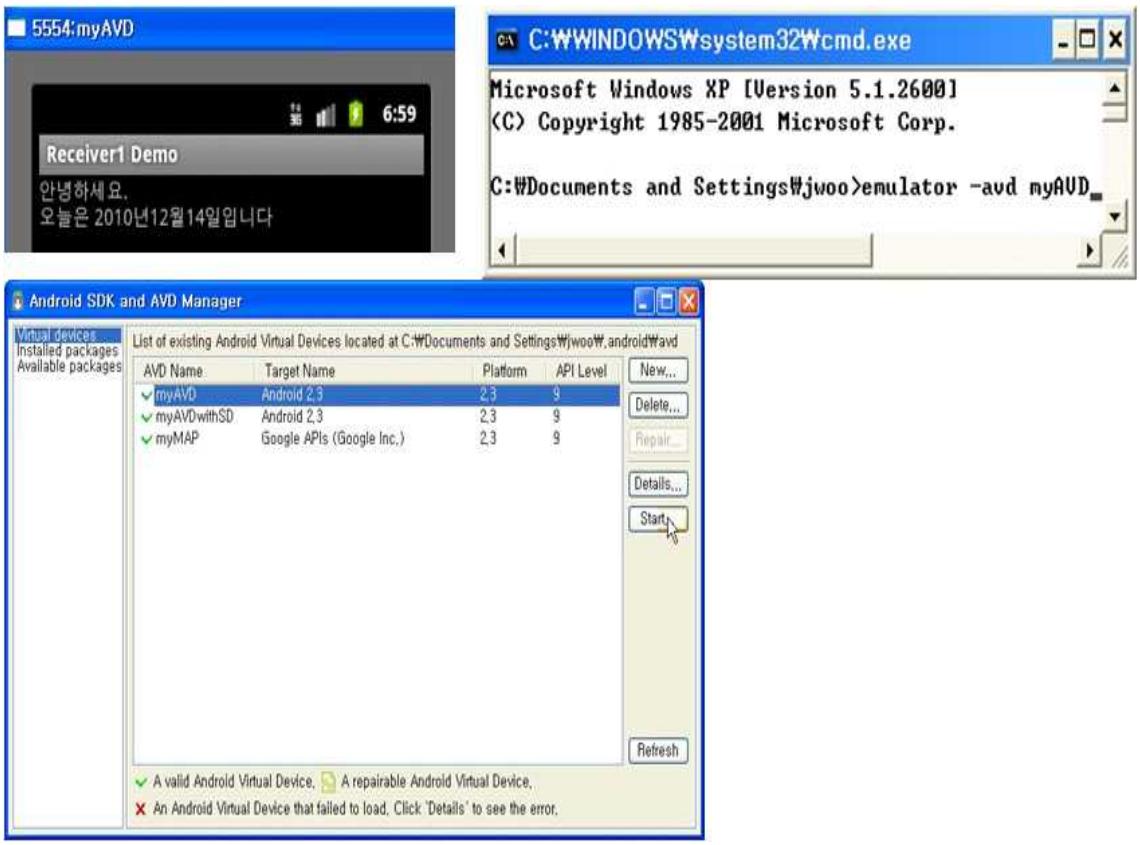


그림 14.3

-예제

■ 액티비티의 버튼을 클릭하면 1초 후에 방송을 보내고 방송수신자는 방송을 수신한 표시로 토스트 메시지를 출력

그림 14.4

-예제

■ SMS를 수신하는 상황에 대하여 방송수신자

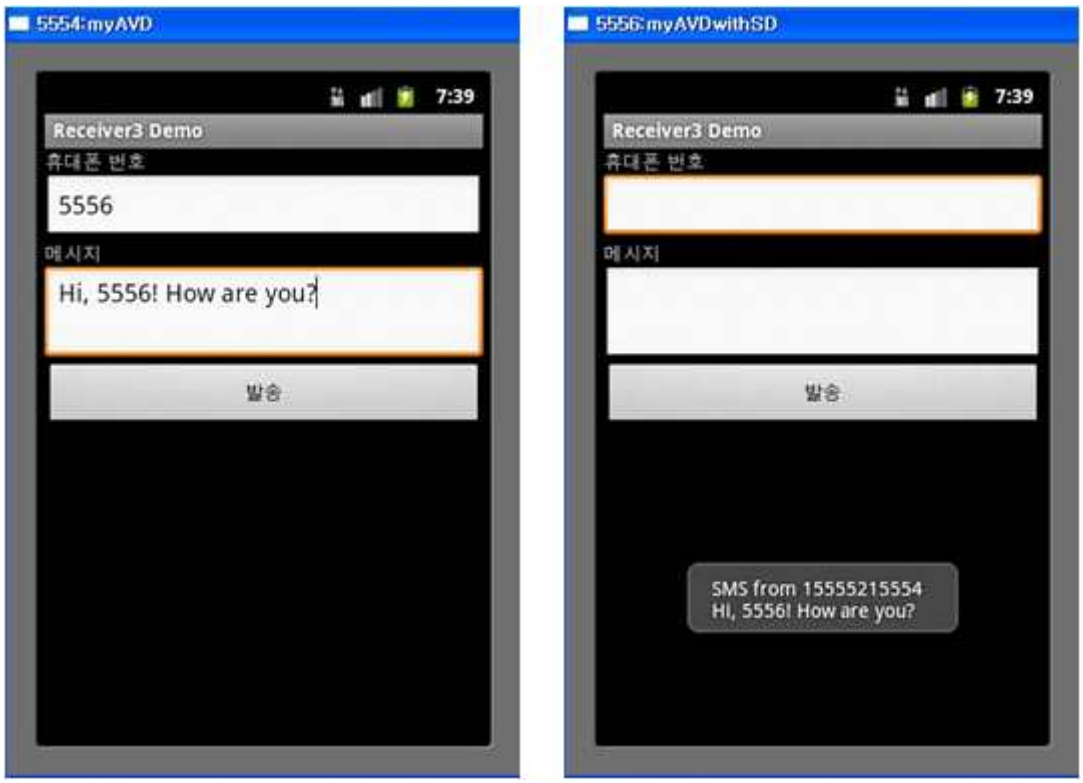


그림 14.5

14.3 알람

14.3.1 개요

-의미

■ 미리 지정해 놓은 시간에 이벤트를 발생시키는 시스템 서비스
■ 운영체제가 관리하기 때문에 애플리케이션이 종료되어도 발생 가능
■ 알람 서비스는 애플리케이션을 장래의 특정 시점에 실행되도록 스케쥴링 가능
■ 알람 시간이 되면 등록된 인텐트가 시스템에 의하여 방송되고 대상 애플리케이션이 자동적으로 시작
■ 핸들러와는 달리 알람은 시스템이 관리하기 때문에 알람이 설정되어 있으면 단말기가 자고 있는 동안에도 단말기를 깨워 애플리케이션을 동작

-알람 객체 생성

■ Context.getSystemService (ALARM_SERVICE)

-알람 등록 및 취소

■ public void set (int type, long triggerAtTime, PendingIntent operation)
■ public void setRepeating (int type, long triggerAtTime, long interval, PendingIntent operation)
■ public void cancel (PendingIntent operation)

14.3.2 응용

-예제

■ 알람을 사용하여 예약 시간이 되면 노티피케이션으로 통지

그림 14.6

14.4 서비스

14.4.1 개요

-의미

■ 데몬이라고도 불리며 액티비티와 연결되어 있지 않아도 계속 실행해야 하는 기능을 구현할 때 사용
■ 사용자와 상호작용하지 않으면서 긴 작업을 수행하거나 다른 애플리케이션이 사용할 기능을 제공
■ 애플리케이션 컴포넌트 중의 하나 → 매니페스트 파일에 추가 필요

-서비스의 종류

■ 백그라운드 데몬: 백그라운드 방식으로 실행되는 프로세스로써 별도의 사용자 명령이 없어도 실행을 계속. 화면에 보이지는 않더라도 계속해서 MP3를 재생하는 MP3 재생기 애플리케이션이 대표적인 보기

■ 원격호출 인터페이스: 클라이언트에게 특정한 기능을 제공하는 역할. 자신의 기능을 메소드 형식으로 제공하여 클라이언트가 메소드를 호출함으로써 서비스를 이용. 다른 종류의 운영체제에서 사용하는 COM이나 CORBA와 유사한 개념.

-서비스 생명 주기

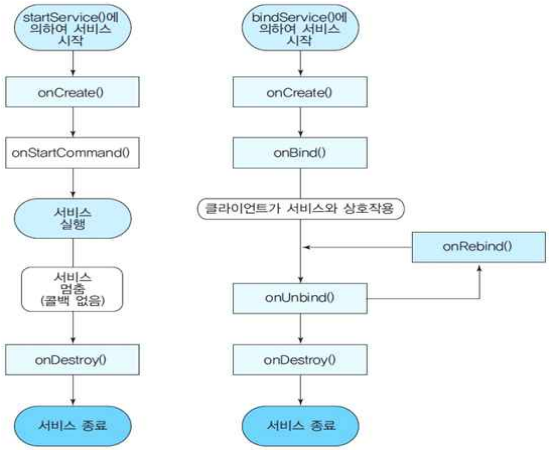


그림 14.7

-백그라운드 데몬

■ 두 가지 방식의 서비스는 전체적으로 별개로 사용되는 것이 아님

■ 백그라운드 데몬 서비스 바인드될 수 있음

■ 따라서 사용되지 않더라도 onCreate()와 onDestory() 메소드 외에 다음 메소드도 반드시 구현

- public int onStartCommand (Intent intent, int flags, int startId)
- public abstract IBinder onBind (Intent intent)

14.4.2 응용

-예제

■ 버튼을 클릭하면 시간을 주기적으로 알려주는 서비스

그림 14.8

[과목2] 안드로이드 예상문제

01. 안드로이드의 구성과 특징에 해당하지 않는 것은?
①자바 바이트 코드를 실행할 수 있는 런타임 라이브러리
②원시 코드를 생성과 개발을 쉽게 할 수 있는 이클립스 IDE
③데이타 베이스를 구동하는 SQLite와 같은 미들웨어
④C언어를 이용한 프로그램에 용이한 안드로이드 SDK

02. 안드로이드가 리눅스 커널이라고 할 수 없는 이유가 아닌 것은?
①GLIBC를 지원 안한다
②EAB와 Binder를 사용
③커널기능향상을 위한 컴포넌트 추가
④리눅스의 커널 코어는 유지

03. 안드로이드 지원 라이브러리의 설명으로 다른 것은?
①SQLite - 소형 관계형 DB 엔진
②SGL - 3D 그래픽 엔진
③Free type - 비트맵 및 벡터 방식의 폰트 렌더링제공
④OpenGL|ES - OpenES 스팩기반의 2D,3D 그래픽 라이브러리 제공

04. 다음 중 안드로이드 플랫폼의 특징이 아닌 것은?
①어플리케이션 프레임워크 - 일종의 마법사를 통하여 프로그래밍할 수 있는 프레임워크를 제공
②달빅 가상머신 - 모바일 디바이스를 위하여 최적화된 자바 가상머신 제공
③ SQLite - 정형화된 데이터를 저장하고 검색하기 위한 중대형 DB시스템
④풍부한 개발환경 - 디바이스 에뮬레이터, 디버깅도구, 메모리 및 성능 프로파일링, 이클립스 IDE를 위한 플러그인 제공

05. 안드로이드의 어플리케이션은 주로 무슨 언어로 작성하는가?
①Basic
②C/C++
③Pascal
④Java

06. 다음 컴포넌트 중 설명이 다른 것은?
①액티비티는 하나이상의 뷰를 사용하며 뷰는 액티비티와 사용자가 상호 작용하는 영역이다
②서비스는 액티비티와 같이 뷰를 가진 사용자 인터페이스를 포함한다
③브로드캐스트 리시버는 방송되는 공지를 수신하고 응답하는 어플리케이션 컴포넌트이다
④컨텐트 프로바이더는 어플리케이션이 보유한 파일이나 데이터베이스 등에 포함된 데이터를 다른 어플리케이션과 공유할 수 있도록 하는 메카니즘이다.

07. 이클립스(Eclipse)에 대한 것으로 적합한 것은?
①IDE(통합개발도구)
②컴파일러
③링커
④인터프리터

08. 중요 안드로이드 도구에 대한 설명으로 틀린 것은?
①에뮬레이터: 달빅 가상머신의 구현으로 하드웨어 의존적
②ADB(Android Debug Bridge): 안드로이드 에뮬레이터 혹은 안드로이드 디바이스에 접속할 수 있도록 하는 클라이언트/서버 애플리케이션을 의미
③DX(Dalvic Executable): 달빅 가상머신에 구동할 수 있는 안드로이드 어플리케이션으로 변환하는 도구
④DDMS(Dalvid Debug Monitor Service): 활성화된 스레드를 감시하거나 중단하는 도구이다. 활성화된 모든 에뮬레이터의 파일시스템을 탐색

09. 프로젝트 폴더에서 설명이 다른 것은?
①₩asset 폴더에서 애플리케이션 배포시 static 하게 패키지에 포함시킬 리소스 저장 파일
②₩asset 폴더에서 리소스들은 컴파일이 안된 raw형태로 패키징되며 AssetManager 클래스를 이용해 바이트 스트림 형식으로 읽어와 사용
③₩asset 폴더에 컴파일된 자바 바이너리 저장
④₩gen 폴더는 프로젝트를 처음 빌드하면 루트폴더 밑에 gen이라는 이름의 폴더가 생성되고 내부에는 R.java 파일이 자동으로 생성

10. APK 파일에 대한 설명으로 맞는 것은?
①실제 안드로이드 디바이스에서 실행할 수 있는 어플리케이션 파일
②디바이스 공간을 줄이기 위해 zip 압축 알고리즘 사용한 앱스토어에서 소프트웨어 제공
③달빅 가상 머신에서 실행가능한 컴파일된 dex파일도 포함
④컴파일이 필요한 리소스 ₩res₩raw폴더 밑의 자료

11.AVD(Android Virtual Devices)의 역할로 틀린 것은?
①안드로이드 SDK에 포함된 에뮬레이터
②안드로이드 SDK1.5이상부터Java와 XML과의 관계로 설명이 다 가능
③가상의 안드로이드 단말기
④각 SVD에 공통의 안드로이드 에뮬레이터 구동

12. Java와 XML과의 관계로 설명이 다른 것은?
①XML로 레이아웃을 선언하는 것은 UI구조가 보다 쉽게 눈에 들어오며 가독성을 높일 수 있다
②Java는 레이아웃과 뷰를 사용하여 화면을 구성하고 XML은 세부적인 행위와 로직을 구성한다
③XML은 이클립스 IDE와 같은 GUI 디자인 툴을 이용해 쉽게 자동화된 생성편집이 가능하다
④XML은 Data문서와 Display문서로 나눌 수 있다

13. XML과 연결되어 처리되는 자바 프로그램을 안드로이드에서는 무엇이라 하는가?
①프로세스(Process)
②액티비티(Activity)
③상태(State)
④쓰레드(Thread)

14. XML에 대해 달리 설명된 것은?
①대부분 "android:"으로 시작되고 뒤에 속성명을 기입한다
②뷰나 뷰그룹을 가로와 세로로 배열할 수 있다
③화면의 크기를 결정하는 요소로 폭과 높이 속성을 지정
④패딩은 여백과 비슷한 개념으로 여백을 선언하지 않으면 좌상단에 붙여서 배치된다

15. 안드로이드의 3가지 기본 유형에 속하지 않는 것은?
①팝업메뉴-스크린 화면 팝업창을 띄우는 메뉴
②옵션 메뉴-액티비티를 위한 메뉴 아이템의 기본적인 집합
③컨텍스트 메뉴-뷰를 길게 누름을 수행할 때 나타날 수 있는 스크린 상에 떠있게 되는 리스트 형태의 메뉴 아이템
④서브메뉴-옵션 메뉴 또는 컨텍스트 메뉴안의 아이템 만으로 부족하여 추가되는 스크린상에 떠 있는 메뉴아이템 목록

16. 레이아웃 인플레이트 사용방법으로 아닌 것은?
①setContentView 사용하기
②직접 버튼 변수에 레이아웃 인플레이트 매핑
③레이아웃 인플레이트이후 i.d. 사용하여 버튼 변수에 매핑
④메인 레이아웃의 작성

17. 안드로이드에서 다음 중 설명이 틀린 것은?
①픽셀은 가장 직관적으로 결과를 보고 배치를 조정하기 편리하다
②ldp는 밀도가 160dpi일 때 1픽셀에 해당한다
③픽셀은 상대적인 값이다
④인치, 밀리미터, 포인트같은 단위는 절대적인 단위이다

18. 안드로이드에서 스타일과 테마에 대한 것 중 설명이 다른 것은?
①스타일은 레이아웃 XML 파일에 있는 하나의 엘리먼트 단위로 적용가능한 하나 이상의 속성집합
②스타일로 예를 들어 특정 텍스트 크기와 그리고 칼라를 정의할 수 있고 특정유형의 뷰 엘리먼트에 적용이 가능하다
③테마는 어플리케이션내의 액티비티 단위로 적용할 수 있는 하나 이상의 속성 집합이다
④테마는 스타일과 달리 XML<style>엘리먼트로 선언되지 않는다.

19. Java1.2. 버전 이후 추가된 Collection Framework는 사용자의 상황에 맞게 객체를 저장하거나 쉽게 조작하는 사용자 인터페이스를 제공한다. 다음 중 설명이 다른 것은?

①자주 사용하는 Array, Maps, Lists 등을 미리 구현하여 제공
②멀티쓰레드 환경을 위한 동기화 처리
③Key를 이용한 검색
④병렬처리, 접근허용, 중복 허용

20. 배열리스트는 배열이 갖고 있는 문제에 대한 해결방안을 제시하는 방법으로 다음과 같다. 선언 방법에 대한 설명 중 다른 것은?
①ArrayList<int>list = new ArrayList<int>(0);
<>은 타입 파라메타라 하는 데 위와 같이 <int>로 선언한 경우는 정수값만 받을 수 있다
②배열은 경우는 a[1]=10;이라 하면 되나 ArrayList에서는 list.add(10);명령어로 입력한다
③만약 a[1]=10;과 같이 특정 인덱스에 값을 넣을 때는 list.add(10)으로 하면 된다
④만약 자료를 읽을 때 배열의 경우에는 x=a[1];인 반면 ArrayList에서는 list.get④으로 선언하여야 한다.

21. 다음 중 어댑터 뷰에 해당되는 것은?
①스피너,그리드 뷰
②커서, 스피너
③커서, 그리드
④스피너,그리드

22.다음 중 설명이 달리 된 것은?
①토스트는 사용자에게 간단한 메시지를 전하는 Vew이다
②안드로이드에서 팝업을 통해 유저에게 특정메시지는 알리는 것으로 토스트와 다이얼로그 두가지 방법이 있다
③토스트는 일정시간이 경과해도 남아있으며 기존 작업의 포커스를 빼았는다
④다이어로그는 사용자 입력이 있어야 없저지며 기존 작업으로부터 포커스를 빼앗는다

23. 다음 중 설명이 잘못된 것은?
①액티비티-스크린의 UI와 관련된 컴포넌트(클래스)
②서비스-백그라운드 작용을 수행하는 기능을 갖는 컴포넌트
③브로드캐스트 리시버-브로드캐스트 인덴트에 대한 수신기능을 수행하는 컴포넌트
④컨텐트 프로바이더-자료를 탐색하는 기능을 갖는 컴포넌트

24. 다음 중 설명이 잘 못된 것으로 잘 못된 것은?
①태스크가 오랫동안 사용되지 않으면 안드로이드는 루트 액티비티를 포함해 모든 액티비티를 삭제한다
②alwaysRetainState 속성: 태스크의 루트 액티비티에 이 속성을 설정하면 이 태스크는 오랫동안 생존한다
③ClearTaskOnlaunch: 태스크의 이 액티비티에 이 속성을 설정하면 액티비티가 태스크를 나갈 때 삭제된다
④finishOnTaskLaunch: clearTaskOnLaunch와 유사하나 이 속성은 액티비티에만 유사하다

25. 액티비티에 대한 전체 생명주기가 정의되는 데 다음 중 3개의 라이프타임에 해당하지 않는 것은?
①전체 라이프타임
②백그라운드 라이프타임
③비저블 라이프타임
④포그라운드 라이프타임

26. 안드로이드 메니페스트 파일은 레이아웃 파일과 유사하게 <manifest> 태그가 전체를 감싸는 구조를 하고 있다, 페키지 속성값이 아닌 것은?
①애플리케이션 베이스로 지정하는 패키지 이름
②<manifest> 태그의 안에 다룰 <application> 태그
③<manifest> 태그의 안에 다룰 <users-permission> 태그
④<android-manifest> 태그

27. <intent-filter>태그에 대한 적절한 설명이 아닌 것은?
①수행되었을 때의 데이터
②컴포넌트 구동에 대한 의도를 무시하기 위해
③적절한 지시(Action)
④지시(Action)를 구성하기 위한 구성요소의 모음(Category)

28. 퍼미션(permission)은 사전에 정의된 것외에 누구나 정의하고 적응할 수 있다. 적용시기에 해당하지 않는 것은?
①어플리케이션 컴포넌트를 시작할 때
②하드웨어나 시스템의 기능을 사용하고자 할 때
③intent로 브로드캐스트를 보내거나 받을 때
④콘텐트 프로바이더에서 제공하는 데이터에 접근을 봉쇄할 때

29. 안드로이드에서 스레드에 대한 다음 중 설명이 달리 된 것은?
①java.lang 패키지의 Thread 상속과 run() 메소드 오버라이드 방법을 사용
②java.lang 패키지의 Runnable 인터페이스 구현
③Runnable 인터페이스에는 한 개이상의 run()이란 추상 메소드 제공
④스레드를 상속 받거나 Runnable이라는 인터페이스를 구현하여 동일하게 run()이란 메소드 구현

30. 다음 중 안드로이드 폰 개발시스템의 특징이 아닌 것은?
①비디오/오디오 지원-SMS, MMS
②핸드셋/레이아웃-VGA, 2D 그래픽 라이브러러, OpenES에 기반한 3D 그래픽
③미디어지원-H.263,H.264 또는 MPEG-4...
④개발환경-기기 에뮬레이터

31. 안드로이드만에 향상된 기능을 위해 추가된 컴포넌트가 아닌 것은?
①바인더

②로우 메모리 킬러
③디바이스 드라이버
④파워 매니지먼트

32. 안드로이드가 제공하는 달빅과 JVM(자바가상머신)의 특징으로 서로 다른 특징이 아닌 것은?
①안드로이드 자바코드를 사용하여 자바코드(class)와 동일하다
②반드시 .class 와 .jar 확장자를 갖는 자바 코드와 구분하기 위하여 달빅 확장자로 .dex로 표기
③한 개의 단말기에 여러개의 가상머신(VM)을 표기
④자바 명령어들이 전부 8비트를 가지는 반면에 달빅은 레지스터 기준으로 32비트로 구성

33. 다음 중 설명이 달리된 것은?
①커널 - 안드로이드 커널은 리눅스 커널이 수정되거나 추가되지 않는 완전히 동일한 커널이다
②아키텍처 - 소프트웨어 구성 및 구성요소들의 상호 인터페이스, 소프트웨어의 설계 및 업그레이드를 위한 원칙 등 소프트웨어의 주요 특징을 결정하는 모든 설계 구조
③프레임워크 -애플리케이션 개발에 기본이 되는 골격코드로 라이브러리와 같이 재사용되는 공통부분은 코드로 구현되며 사용자의 요구 사항에 적합하게 확장될 부분은 최적화한 미완성 어플리케이션
④플랫폼 - 소프트웨어를 실행하기 위한 모든 구성과 구조를 포함한 환경을 의미한다. 윈도우즈와 리눅스는 각각 윈도우즈 프로그램과 리눅스 프로그램을 실행시킬 수 있는 플랫폼이다

34. 다음 중 안드로이드 어플리케이션이 달빅 가상 머신에서 수행되는 것으로 맞지 않은 설명은?
①하나의 어플리케이션은 기본적으로 하나의 프로세스 내에서 동작한다
②안드로이드는 항상 명시적으로 종료할 필요가 있다.
③모든 프로세스는 자신의 가상 머신을 가지기 때문에 다른 프로세스와는 격리되어 실행된다
④파일을 공유하기 위한 여러 방법이 있지만 파일을 가진 어플리케이션은 자신만 파일에 접근할 수 있다.

35. 다음 중 프로세스에 대한 설명으로 다른 것은?
①전경 프로세스: 현재 포커스를 가지고 있는 프로세스,즉 사용자와 상호작용하는 프로세스
②가시 프로세스: 포커스가 있고 화면에 보이는 프로세스
③서비스 프로세스: 화면에 보이는 즉 네트워크를 통해 파일을 다운로드하거나 음악을 재생하는 프로세스
④배경 프로세스: 화면에 보이지 않지만 액티비티를 포함하는 프로세스이다

36. 안드로이드폰 어플 개발에 꼭 필요한 툴체인이 아닌 것은?
①타겟 하드웨어
②자바개발도구(JDK)
③안드로이드 SDK
④이클립스(Eclipse)

37. 중요 안드로이드에 대한 설명으로 틀린 것은?

①MKSDCARD: 하드 디스크의 일부분을 안드로이드 에뮬레이터에서 가상의 SD카드로 생성할 수 있도록 하는 도구
②AAPT(Android Asset Packing Tool): 배포할 수 있는 안드로이드 패키지 파일을 생성하는 도구
③SQLite3: 중대규모의 관계형 DB 파일 즉 모바일 디바이스를 위한 데이터베이스 파일을 제거하는 도구
④AIDL(Android Interface Definition Language): 안드로이드 디바이스에서 2개의 프로세스가 IPC를 사용하여 대화할 수 있는 코드를 작성하는 언어

38. 프로젝트 폴더의 내용 중 설명이 다른 것은?
①₩gen 폴더의 R.java 파일은 프로젝트에서 사용되는 이미지,레이아웃, 스트링등의 리소스를 가르키는 인덱스 파일
②R.java는 ₩src 디렉토리의 자바코드 내부에서 리소스 디렉토리의 XML 파일과 연관시키는 클래스
③₩gen은 애플리케이션이 필요한 외부 jar파일을 저장하는 폴더
④₩res 는 애플리케이션의 레이아웃,이미지,문자열등 컴파일된 자바코드와 함께 패키징되는 리소스 저장폴더

39. APK 파일 내부의 구성 요소가 아닌 것은?
①dex 파일
②META-INF
③AndroidManifest.xml
④strings.xml

40. 인스톨 패키지(APK)에 대한 설명으로 해당되지 않는 것은?
①APK파일은 실제 안드로이드 디바이스에서 실행되지 않는 가상의 어플리케이션 파일
②디바이스 공간을 줄이기위해 zip압축 알고리즘을 사용한 압축 파일형태 안드로이드 마켓에서 소프트웨어를 제공하는 방안
③달빅 가상 머신에서 실행 가능한 컴파일된 바이너리 파일은 dex 파일 포함
④컴파일된 리소스 및 컴파일이 필요없는 리소스 포함

41. 안드로이드에서 제공하는 두가지 XML문서의 형태는?
①레이아웃 클래스와 위젯 클래스
②뷰그룹 클래스와 레이아웃 클래스
③뷰클레스와 위젯 클래스
④기본클래스와 파생클래스

42. 안드로이드에서 XML을 사용하여 화면을 디자인하고 XML과 자바를 연결하기 위해 정한 규약이 아닌 것은?
①속성은 선택사항으로 "android"로 시작하고 뒤에 속성명을 기입힌다
②XML 엘리먼트(요소)는 자바와 연관된 <LinearLayout>,<TextView>,<Button>등 정

해진 이름을 사용해야 한다.
③XML에서 요소(엘리먼트) 이름은 클래스와 대응하여 속성의 이름은 메소드의 이름과 연관이 있다
④현재 XML에서 자신의 뷰나 뷰그룹보다 아래쪽에 정의된 뷰도 참조할 수 있다.

43. 텍스트 관련 뷰 종류에 대한 설명으로 틀린 것은?
①TextView:표준 읽기 전용 텍스트 레이블
②EditText:편집이 가능한 텍스트 입력 박스
③ListView:항목을 리스트에 나타내는 데 사용되는 뷰들의 그룹을 생성하고 관리하는 뷰그룹
④Spinner:EditView와 그와 연관된 ListView를 표시하는 복합컨트롤러

44. 레이아웃 배치에 대한 설명으로 다른 것은?
①상위레이아웃 기준 배치는 상대배치이다
②별도의 뷰 기준으로 배치하는 것은 상대배치이다
③별도의 뷰를 기준으로 배치하는 것은 절대배치이다
④ 테이블 레이아웃배치는 같은 뷰들을 테이블 형태로 배치하는 XML의 속성이다

45. 옵션 메뉴에 대한 설명으로 다른 것은?
①액티비티를 위한 아이템의 기본적인 집합으로 디바이스 메뉴키를 누르면 나타난다
②아이텐 선언시에 사용자가 텍스트 이외에 눈으로 쉽게 구분할 수 있도록 할 수 있다
③아이콘 메뉴와 확장 메뉴가 속한다
④컨텍스트 메뉴와 서브메뉴가 이에 속한다

46. 다음 설명중 틀린 것은?
①리소스 디렉토리에 놓인 파일들은 컴파일된 R클래스를 통해 애플리케이션에서 접근 가능하다
②모든 안드로이드 애플리케이션은 리소스를 위한 디렉토리와 에셋을 위한 디렉토리를 가진다
③에셋 디렉토리 안에 있는 것은 원시 파일 포맷으로 유진된다
④리소스 디렉토리 안에 있는 것은 문자 폰트처럼 바이트 스트림으로 파일을 읽기 위한 메니저를 사용

47. 안드로이드에서 뷰와 내용물간의 간격을 조정하는 것으로 안쪽 여백에 해당하는 것은?
①padding
②visibility
③clickable
④focusable

48. 다음 중 이벤트 리스너와 핸들러에 대한 설명으로 다른 것은?
①이벤트 리스너는 버튼에서 어떤 동작이 일어날지 기다린 후에 사용자가 버튼을 누르면 정해진 동작을 실행한다
②이베트가 일어나면 안드로이드 프레임워크에 의해 콜백함수가 호출된다
③이벤트 핸들러는 화면의 UI엘리먼트에 관련되며 주로 카메라,키보드 그리고 방향키를

누르는 것과 같이 같은 이벤트에 대해 사전 정의된 핸들러 메쏘드가 실행된다
④주로 액티비티나 리스트 액티비티 그리고 다이어로그와 관련해 작업을 실행하며 대표적인 예가 Key Event이다

49. 자바에서 자바 컬렉션 프레임워크란 이름으로 제공하지 않는 것은?
①List
②Set
③Array
④Map

50. Map에 대한 설명으로 맞는 것은?
①키는 중복을 허용한다
②각 키는 1개이상의 값에 매핑을 할 수 있다
③키는 중복을 허용하며 각 키는 한 개의 갑에 매핑을 할 수 있다
④키는 중복을 허용하지 않으며 각 키는 한 개의 값에 매핑할 수 있다

51. 다른 비주얼 툴 킷의 Combo box와 같이 드롭다운선택의 기능을 하는 것으로 리스트뷰와 같이 화면을 모두 차지하지 않으면서 특정의 data set에서 입력이 필요할 때 사용하는 것은?
①커서
②스피너
③그리드
④배열

52. 액티비티에 대한 설명으로 잘못된 것은?
①활동을 뜻하는 데 주로 직접 사용자의 명령에 대한 인터페이스 작업 수행
②스크린에 우리가 제작한 화면을 프리젠테이션
③기본적으로 하나의 윈도우가 부여
④윈도우는 버튼, 스크롤, 메뉴와 같은 비주얼 콘텐츠를 위해 다양한 종류의 뷰클래스들을 제공

53. 컴포넌트 활성화와 종료에 대한 설명으로 잘못된 것은?
①인텐트(indent)란 어떤 작업을 요청하는 의사, 의향, 의도를 담은 메시지
②컨텐트 resolver는 컨텐트 해결사로 리눅스에서는 URL에 따른 해석을 요청하는 작업
③브로트캐스트는 방송이란 의미로 안드로이드 시스템 내부의 일부 어플리케이션을 대상으로 방송한다
④안드로이드에서는 “생명주기”란 방식으로 액티비티와 서비스를 종료하기 위한 메카니즘을 제공

54. 일반 PC나 서버와 달리 안드로이드 어플리케이션 컴포넌트들은 각각 나름대로의 생명주기를 갖는다, 이유는 스마트폰 속성상 임베디드 리눅스에 없던 안드로이드만의 특수기능으로 커널에 추가된 것이 아닌 것은?
①알람

②로우 메모리 킬러
③파워매니지먼트
④브로드캐스팅

55. 다음 중 디바이스 변경관리 중 디바이스 설정값이 변경되는 경우가 아닌 것은?
①화면 방향의 변화
②쿼티 자판의 숨기기와 나타내기
③언어 및 디바이스 설정 파일 변경
④전원의 온오프

56. 위치 정보, SMS 수신, 전화걸기, 주소록 읽기, 인터넷 사용 등을 애플리케이션에서 사용하기 위해서 그에 해당하는 퍼미션은?
①<users-permission>
②<application-permission>
③<manifest-permission>
④<instumentation-permission>

57. 다음 태그 들에 대한 설명이 아닌 것은?
①<activity-alias> : targetActivity으로 <activity> 타겟으로 지정하고 그 타겟과 다른 속성으로 <activity> 호출하는 데 activity-alias는 targetActivity의 속성을 따르지만 activity-alias안에 같은 속성이 선언되면 activity-alias 속성을 따른다
②<receiver tag>: 애플리케이션 내 브로드캐스트리시버 컴포넌트가 브로드캐스트 메시지를 수신할 수 있도록 안드로이드 시스템안에 알리게 된다
③<service-tag>: 만약 어플리케이션안에 서비스가 존재한다면 필요한 것으로 액티비티가 백그라운드라면 서비스는 포그라운드 오퍼레이션에 해당한다
④<users-library> 태그: 어플리케이션에 링크되어야 하는 외부 라이브러리를 존중한다

58. 스레드에 대한 설명으로 다른 것은?
①프로세스에 비해 생성이 늦다
②다른 스레드와의 통신에 대한 작업로드가 작다
③메인 스레드는 프로세스의 메인함수로서 실행되는 스레드를 말한다
④일반 스레드는 메인스레드에서 파생되어 자식으로 언급되는 스레드나 언급되는 스레드나 다른 어플리케이션의 스레드이다

59. 다음 중 설명이 달리 된 것은?
①한번 사용한 스레드는 재사용이 불가능하다
②각 스레드는 CPU의 점유순서를 결정하는 우선순위를 정하는 정수를 가지고 있다
③포그라운드의 스레드들이 비활성화된다 하더라도 백그라운드에서 독립적인 데몬형태로 작동되는 스레드는 종료되지 않을 수 있다
④일반적으로 자바언어는 다중상속이 되므로 Thread의 상속이 Runnable인터페이스보다 자주 사용된다.

60. 안드로이드 플랫폼의 레이어가 아닌 것은?
①커널-리눅스 커널 2.6 코어
②하드웨어 추상-HAL에 속하는 부분
③라이브러리
④안드로이드 마켓

61. 안드로이드에서 HAL과 중간 미들웨어를 만든 이유가 아닌 것은?
①커널 드라이버는 GPL의 라이센스에 적용을 받는다
②안드로이드의 모든 라이브러리들이 표준화되어 있지않다
③안드로이드만의 독특한 기능(바인더와 파워매니저) 부여
④사용방법으로 호출방식은 파스칼에서 호출하는 방법과 같다

62. 안드로이드 SDK란 무엇인가?
①유닉스 기반의 운영체제 사용자들을 위한 그래픽 사용자 인터페이스와 일련의 컴퓨터 데스크탑 어플리케이션
②안드로이드 애플리케이션을 개발할 수 있도록 윈도우/리눅스/MAC OS 개발환경
③윈도우 사용시 cigwin을 사용하여 리눅스와 같은 환경을 만드는 것
④안드로이드를 개발하기 위한 C/C++과 이클립스

63. 안드로이드 어플리케이션 프레임의 특징이 아닌 것은?
①임베디드 리눅스 기반의 디바이스에 적합한 주로 자바 라이브러리
②PacketVideo의 OpenCore기반의 미디어 라이브러리
③2D와 3D 그래픽 레이어를 단일하게 관리하기 위한 서피스(Surface) 매니저
③OpenGL ES API 기반으로 구현된 3D 라이브러리

64. 안드로이드 어플리케이션이 시스템이 필요로 할 때 인스턴스화되고 실행될 수 있는 4가지 유형의 컴포넌트로 맞지 않는 하나는?
①액티비티
②컨텐트 프로바이더
③브로드캐스트 리시버
④유니캐스트 리시버

65. 아이폰을 개발하는 데 사용하는 프로그램 언어는?
①Basic
②Java
③Object-C
④Pascal

66. 임베디드 시스템 개발 환경에 대해 잘 못 설명한 것은?
①호스트 시스템은 모바일 디바이스를 개발하기 위한 자원 즉 컴퓨팅 환경을 제공
②일반적으로 호스트 운영체제로는 임베디드 리눅스를 사용

③백엔드는 호스트와 타겟 사이의 시리얼 통신, 이더넷, JTAG 및 케이블
④타겟 시스템은 개발된 임베디드 시스템 소프트웨어를 테스트하고 구동할 수 있는 시스템

67. 안드로이드 명령어로 Adb 명령어의 기능이 아닌 것은?
①애플리케이션 패키지(APK)설치 및 삭제
②파일들을 단말기에 복사작업
③단말기 내부에서 안드로이드 명령어 수행
④응용 어플 프로그램의 실행

68. 프로젝트 폴더 ₩res에 다음의 서로 다른 리소스가 저장되는 데 해당되지 않는 것은?
①res₩raw: 컴파일된 리소스의 저장
②res₩anim: animation을 지정하는 xml형식의 문자 저장
③res₩drawable:png, jpeg 등의 이미지 파일
④res₩layout: UI 레이아웃 관련 xml형식의 문서파일

69. ADT 플러그인의 역할은?
①이클립스와 JDK의 연결
②이클립스와 SDK의 연결
③JDK와 SDK의 연결
④에뮬레이터와 JDK의 연결

70. 다음 중 잘못 설명이 된 것은?
①제작한 안드로이드 어플리케이션에 서명을 넣어야 설치 및 판매가 이루어진다
②안드로이는 애플리케이션의 신뢰와 제작자의 신원확인 목적으로 서명을 한다
③인증은 한국공인인증기관이나 베리사인과 같은 국제 공인인증기관의 키와 자체인증키가 있어야 한다
④인증서 종료는 특별한 기한을 명기하지 않는 경우 10년이다

71. 다음 중 뷰그룹과 뷰에 대한 설명으로 맞지 않는 것은?
①뷰그룹은 레이아웃 클래스, 뷰는 위젯 클래스이다
②뷰그룹에는 아키텍처로 리니어,태블러 그리고 상대 레이아웃의 서브클래스를 제공
③뷰클래스는 텍스트 필드, 버튼 및 체크박스 UI객체들을 제공하는 서브 클래스를 사용한다
④뷰그룹은 위젯 클래스, 뷰는 레이아웃 클래스이다

72. 객체 지향 프로그램에서 클래스의 개념으로 맞지 않는 것은?
①클래스는 전부를 그 클래스 특성으로 상속받는 서브클래스를 가질 수 없다
②클래스는 각 서브클래스에 대해 슈퍼 클래스가 된다
③서브클래스는 자신만의 메소드와 변수를 정의할 수 있다
④클래스와 서브클래스간의 구조를 “클래스 계층”이라고 한다

73. 다음 중 설명이 달리 된 것은?
①액티비티는 자바 문법의 클래스 객체이다
②위젯은 사용자와의 상호작용을 위한 인터페이스를 제공하는 뷰객체이다.
③레이아웃은 안드로이드 단말기 화면안에 자신으로 소속된 뷰들을 배열하고 배치하는 틀을 제공
④스크롤뷰는 한정된 물리화면을 한번에 표시하기 어려울 때 사용

74. 다음 중 설명이 달리 된 것은?
①이미지 버튼은 TextView를 상속받은 클래스이다
②버튼은 TextView를 상속받은 클래이스이다
③TextView는 문자열을 표시하는 뷰이다
④ImageView는 이미지를 표시해주는 역학을 하는 뷰이다

75. switch~~case 문에서 case로 선언된 item id와 switch인자로 전달된 item 인스턴스의 id를 비교하여 어떤 item인지 확인하여 그에 맞는 case별로 처리를 한다. switch~~case문의 특징이 아닌 것은?
①default는 항상 존재해야 한다
②case문의 item id로 서용되는 정수의 값은 항상 달라야 한다
③case문내 break문 또는 return문이 없을 때 그 아래 case 문을 수행한다
④swith~~case 문은 if~~else if 문으로도 표현이 가능하다.

76. 문자열과 스타일 텍스트의 특징이 아닌 것은?
①문자열의 경우 res/values/string.xml에 모여있음
②일반 문자열은 언어 사용에 있어 불편함
③HTML 스타일은 문자열을 웹에서 사용하는 HTML형태로 관리 가능
④문자열 포맷팅:적절한 위치에 변수값을 넣어 문자열 변형이 가능

77. text는 텍스트 뷰의 가장 중요한 속성으로 출력할 문자열을 지정한다. 여기에 제공되는 것이 아닌 것은?
①textColor
②textSize
③textStyle
④textLocation

78. 다음 중 콜백 함수에 대한 설명으로 틀린 것은?
①콜백함수는 프로그래머가 그 함수의 실행을 명령하는 것으로 이벤트와 메시지를 처리한다
②전화를 건 후 전화번호를 남겨두면 상대방이 다시 거는 것과 유사하다
③콜백함수를 쓰면 이벤트 발생을 감지하기 위한 처리와 그 이벤트 발생시 실행할 각각의 처리를 나누어 코딩할 수 있다
④대부분 On으로 시작되는 함수들로 운영체제가 호출할 어플리케이션의 함수를 지정해 특정한 사건 또는 메시지가 발생하였을 때 호출되도록 지정할 수 있다.

79. 자바에서 Set과 Map에 대한 설명으로 다른 것은?

①SET은 중복을 허용하지 않으며 저장순서가 유 지되지 않는 특징 보유
②Map은 키의 값을 하나의 쌍으로 묶어 컬렉션을 구현
③HashSet은 HashMap보다 느리지만 빠른 집합
④HashMap은 가장 느리며 null값을 허용 안함

80. 다음 중 BaseAdapter에 해당되는 것은?
①ListAdapter와 SpinnerAdapter의 인터페이스에서 공동으로 사용되는 Adapter인터 페이스를 구현하는 역할을 수행하는 어댑터
②데이타 베이스와 어플리케이션을 연결하는 Adapter
③화면과 어플리케이션을 연결하는 어댑터
④여러 컬럼을 구분하여 리스트하기에 편하게 사용할 수 있게 만드는 어댑터

81. 안드로이드에서 제공하는 Tab의 구성요소에 대한 설명이 잘못된 것은?
①탭호스트:Tabwidget과 FrameLayout을 포함하는 전체 컨테이너
②탭 Widget: Tab버튼 모음을 나타내며 각각의 tab 버튼은 text+icon(옵션)으로 이루어진다
③FrameLayout:선택된 Tab에 대한 실제 내용을 표현하기 위한 레이아웃
①스피너: 다른 비쥬얼 툴킷의 콤보박스와 같은 드롭다운 선택의 기능을 구현한다

82. 어플리케이션 컴포넌트에 해당하지 않는 것은?
①어댑터
②서비스
③브로드캐스트 리시버
④컨텐츠 프로바이더

83. 태스크와 엑티비티에 대한 다음 설명 중 틀린 것은?
①태스크는 사용자가 실질적으로 하나의 어플리케이션처럼 느끼는 액티비티 들의 집합
②태스크는 스택형태로 구성되며 최하단에는 루트(root)엑티비티가 존재한다
③상단 액티비티는 화면에 보이지 않는 백그라운드 액티비티이다
④안드로이드에서는 태스크 액티비디를 액티비티가 싸인 스택이라하며 액티비티 스택이라 한다

84. 액티비티는 생명주기는 사용자의 포커스(현재 어플리케이션의 사용유무)에 따라 같은 세가지 상태에 해당되지 않는 것은?
①활성 중 혹은 실행 중인 상태
②대기 상태
③멈춤 상태
④정지 상태

85. 액티비티의 생명주기에 따라 액티비티의 OnDestroy()와 OnCreate()라는 과정을 통해 활성화 시간이 소요되고 바테리 소모등의 문제가 발생할 수 있는 상황이 아닌 것은?
①대용량 파일의 리로드 문제
②네트워크의 연결 종료와 재연결 시도

③전원의 온오프
④Bundle 객체의 용량 한계

86. 안드로이드에서는 사전에 애플리케이션에서 필요로하는 퍼미션들을 정의했는데 이에 해당되는 조건이 아닌 것은?
①액티비티나 서비스와 같이 어플리케이션을 시작할 때
②콘텐트 프로바이더에서 제공하는 데이터에 접근할 때
③시스템 소프트웨어를 사용하고자 할 때
④intent로 브로드캐스트를 보내거나 받을 때

87. 안드로이드 보안 모델에 대한 다음 설명 중 다른 것은?
①기본적으로 리눅스 기반의 프로세스들간의 격리
②윈도우즈 파일 시스템
③애플리케이션별 고유한 사용자 ID의 부여
④애플리케이션에 대한 서명 구조에 기반

88. 다음 중 설명이 달리 된 것은?
①스레드는 프로세스내의 개별 실행 유닛이다
②다른 스레드와의 통신에 대한 작업 로드가 작다
③프로세스에 대해 생성이 느리다
④일반 스레드는 메인 스레드에서 파생되어 자식으로 언급되는 스레드나 다른 어플리케이션의 스레드를 말한다

89. 다음 중 설명이 달리 된 것은?
①쓰레드가 많으면 많을수록 이벤트 핸들러처럼 스레드를 전문적으로 처리하기 위하여 핸들러를 도입하는 것이 바람직하다
②핸들러는 스레드와의 상호 작용을 위한 메소드로 핸들러 인스턴스는 역시 하나의 스레드이며 메시지큐는 일대일 대응이 안된다
③핸들러 사용목적은 특정 시점에 예정 작업을 위하여 필요한 실행명령이나 실행 Runnable객체의 관리를 행한다
④핸들러 자신이 아닌 다른 스레드에 작업을 요청한다

90. 리눅스 커널의 특징으로 맞지 않는 것은?
①오픈 소스
②안전성 검증
③공유 라이브러리 지원
④실시간 운영체제

91. 이클립스(Eclipse)에 대한 설명으로 틀린 것은?
①IDE(통합개발환경)
②Java개발시 유리하다

③유료이다
④C/C++개발도 가능하다

92. 다음 중 리소스 매니저의 역할은?
①어플리케이션의 데이터를 공유
②문자열, 그래픽, 레이아웃 파일과 같은 비코드 리소스의 접근
③모든 어플리케이션의 상태에 경고나 알림 내용 표시
④어플리케이션의 생명주기와 어플리케이션의 향해 역사를 관리

93. 안드로이드에 대해 다음 중 틀린 것은?
①어플리케이션을 구성하는 컴포넌트는 매니패스트(Manifest)파일에 정의
②모든 컴포넌트는 메인(main)시작점에서 출발한다
③안드로이드는 인덴트(intent)를 통하여 어플리케이션을 구동시킨다
④액티비티는 사용자와 상호작용하는 하나의 화면으로 어플리케이션은 적어도 하나의 액티비티를 포함한다

94. 안드로이드의 강점이 아닌 것은?
①플랫폼의 개방
②플랫폼의 무료
③플랫폼의 완전성
④애플의 지원

95. 다음 설명중 틀린 것은?
①JDK(자바개발도구)는 선마이크로시스템스에서 무료로 배포
②안드로이드 SDK는 해당 웹사이트서 압축파일을 다운 받아 설치한다
③이클립스는 선마이크로시스템스에서 개발된 오픈 소스로 제공한 자바기반의 확장 가능한 개발도구 제작에 필요
④이클립스를 안드로이드 개발 목적으로 사용하려면 이클립스에 ADT(Android Development Toolkit) 플러그인을 설치

96. 안드로이드 프로젝트를 컴파일 후 프로젝트폴더₩bin₩ 폴더의 내용을 달리 설명한 것은?
①bin₩classes 폴더: 컴파일된 자바 코드
②bin₩classes.dex: 실행가능한 컴파일된 자바코드
③bin₩AppName.apk: res 폴더 밑의 리소스 들이 압축이 해제된 파일
④bin₩AppName-debug.apk 또는 bin₩AppName-unsigned.apk: install 가능한 안드로이드 애플리케이션

97. 해당 애플리케이션의 컴포넌트(activity, service, Content provider) 정보 및 보안 그리고 버전업 등과 관련된 xml형태의 문서는?
①AndroidManifest.xml
②build.xml

③hello.xml
④strings.xml

98. ADT 플러그인에 대해 기술된 것 틀린 것은?
①SDK설치 후 ADT 플러그인을 설치한다
②JDK 설치 후 ADT 플러그인을 설치한다
③JDK와 SDK설치 후 ADT 플러그인을 설치한다
④이클립스 설치전 ADT 플러그인을 설치한다

99. 서명작업과 키에 대한 설명으로 다른 것은?
①이클립스를 띄우고 프로젝트명에 오른쪽 마우스를 클릭하고 "Android Tools"->Export Signed Application Package"를 선택하여 서명한다
②이클립스에서 File->Export->Android->Export Android Application를 선택하여 서명한다
③공인키는 아무에게나 오픈할 수 없는 키이다
④개인키는 자신만이 보관하는 키이다

100. 스마트폰 상의 어플리케이션을 제작할 때 고려사항이 아닌 것은?
①짧은 응답시간
②안전성 확보
③복잡한 네트워크 구조
④효율적인 전원 사용

101. 다음 설명이 다르게 된 것은?
①클래스로부터 객체를 만드는 과정을 클래스의 인스턴스라 한다
②인스턴스와 객체는 같은 의미이며 엄격히 구별할 수 없다
③클래스는 속성을 나타내는 변수와 기능을 나타내는 메소드로 구분된다
④액티비티는 사용자가 사용하는 화면과 관련된 인터페이스를 담당하는 C 프로그램이다.

102. 스마트 폰이 데스크 탑 환경 PC에 비해 가지는 단점이 아닌 것은?
①바테리에 따른 한정된 운영시간
②느린 응답성
③불안정한 네트워크 환경
④낮은 하드웨어 환경

103. 라디오 버튼의 특성으로 잘 못된 설명은?
①라디오버튼은 보통 라디오그룹에 의해 여러개가 한 묶음으로 사용된다
②같은 라디오 그룹내부의 라디오버튼은 한번에 하나만 선택 가능하다
③라디오버튼은 체크박스와 같이 한번 체크되면 사용자가 직접 체크를 푸는 방법이 있다
④체크 한 상태와 체크하지 않은 두 상태를 가진다

104. 레이아웃 인플레이터(Layoutinflator) 개발단계로 잘못된 것은?

①자바 프로그램 내부에서 레이아웃 인플레이터를 만든다
②원하는 뷰를 읽어 뷰그룹을 만든다
③메인 레이아웃을 만든다
④화면에 디스플레이 작업을 행한다.

105. 다음 중 설명이 맞지 않는 것은?
①뷰그룹-직접적으로 보이진 않으며 다른 뷰를 담는 컨테이너 역할을 한다
②위젯 - 직접적으로 보이지 않으며 사용자 인터페이스를 구성한다
③뷰그룹-여러개의 뷰를 유기적으로 모아놓은 뷰의 집합으로 이 부류 클래스를 레이아웃이라 한다
④View는 자바 클래스의 일종으로 당연히 루트인 object로부터 파생된다.

106. 다음 중 안드로이드에서 글꼴의 모양을 지정하는 것은?
①textColor
②typeface
③textSize
④textStyle

107. 자바에서 this 레퍼런스에 대한 설명으로 다른 것은?
①자바에서 인스턴스를 참조하기 위해서는 보통 직접 선언된 클래스에 대해 참조하도록 한 후에 사용한다
②this는 인스턴스 자체를 참조할 수 있게 하는 기능이다
③인스턴스를 전역변수처럼 사용한다
④자기 자신만을 가르키는 레퍼런스로 메소드 내부의 모든 인스턴트 필드와 메소드 앞에 암묵적으로 사용한다

108. 다음에 대한 설명 중 다른 것은?
①Set은 중복을 허용하며 저장 순서가 유지되지 않음
②MAP은 키와 값을 하나의 쌍으로 묶어 저장하는 컬렉션을 구현하는 데 사용
③List는 요소들의 중복을 허용하면서 요소들을 순차적으로 유지시킨다
④안드로이드에서는 Set을 거의 사용하지 않으며 키로 인한 저장 검색을 원하면 Map을 쓰게 되며 대부분의 경우는 List를 사용하게 된다

109. 다음 설명 중 틀린 것은?
①안드로이드는 직접 물리적인 화면에 Row단위로 디스플레이하며 가상화면과 실제화면의 매핑을 하지 않는다
②베이스 어댑터와 같이 추상 계열의 클래스에서는 getView()라는 메쏘드를 이용하여 가상화면의 뷰단위에 대비되는 물리적인 화면에 디스플레이한다.
③ArrayAdapter와 같은 어댑터들은 스스로 가상화면(논리적인 화면)을 구성하여 모든 리스트를 물리적인 화면에 맞게 조절하여 디스플레이한다
④가상 화면의 Row는 주어진 position으로 식별되고 Row단위로 하나씩 읽어 두 번째 인자인 convertView의 형태로 제공된다

110. TabActivity를 이용한 동적구성은 TabHost에 탭 위젯을 등록한다. 탭은 탭자체를 추적하기 위해 담아야 하는 내용으로 다른 것은?
①탭제목
②탭지시자
③탭내용
④탭태그

111. 뒤에서 활동(백그라운드)하는 클래스는?
①액티비티
②서비스
③브로드캐스트 리시버
④컨텐트 프로바이더

112. 다음 중 스택과 엑티비티의 설명이 달리 된 것은?
①액티비티가 다른 액티비티를 시작하면 그 새로운 액티비티가 스택에 푸시(push)되고 그 액티비티가 하단 Bottom 액티비티가 된다.
②같은 액티비티라 해도 여러개의 인스턴스가 가능하며 각각의 인스턴스들은 다른 설정환경이나 메모리 번지 수등은 각각 다르다
③태스크내 모든 액티비티들은 하나의 집합으로 백그라운드 또는 포그라운드로 이동이 가능하다
④스택내 액티비티가 재정렬되지 않으며 순서는 그대로 유지되며 단지 PUSH나 POP으로 필요한 액티비티를 넣을 수 있으며 액티비티를 제거할 수 있다.

113. 다음 중 액티비티가 Visible, Foreground에 해당되는 것은?
①OnResumed, OnPaused
②OnCreate, OnRestart
③OnRestart, OnStop
④OnCreate, OnDestroy

114. 안드로이드 프로젝트안에는 꼭 하나의 매니페스트 파일이 존재하여야 한다. 이 매니페스트 파일안에속하는 항목이 아닌 것은?
①애플리케이션 컴포넌트 들의 구성 목록
②외부 자바 라이브러리 항목
③프로세스와 테스트의 메모리 사용방법
④user-permission을 사용해 사용하고자 하는 하드웨어나 네트워크 사용 유무

115. 안드로이드에서 서비스를 태스트하기 위한 테스트 어플리케이션을 만들 수 있는 프레이워크를 제공하여 시스템 리소스와 상호작용에 대한 모니터링을 제공하는 것은?
①<instrumentation> 태그
②<users-feature> 태그
③<supports-screen>태그

④<users-permission> 태그

116. 퍼미션은 안드로이드 매니페스트 파일의 <permission>요소를 사용해 정의되며 보호수준 <protection level>이라는 속성을 가지게 된다. 퍼미션의 기본적인 보호 수준이 아닌 것은?
①보호(normal): 자바 프로그램안에서 권한들을 시행할 때 사용하는 기본값이다
②위험(dangerous):기기에 해를 입힐 수 있는 위험한 활동이 될 수 있는 보안 수준을 의미한다
③활동(activity): 응용 프로그램의 연동을 위해 사용되는 보안 수준을 의미한다
④서명(signature): 자신의 것과 동일한 인증서로 서명된 응용프로그램 들이 애플리케이션의 구성요소에 자유로이 접근할 수 있다

117. 스레드에 대해 설명이 달리 된 것은?
①프로세스내의 여러 스레드는 실행에 필요한 카운터, 레지스터, 스택을 스레드별로 따로 갖는다
②프로세스내의 스레드들은 서로의 메모리 공간을 공유하지 못한다
③프로세스간의 전환속도보다 스레드간의 전환속도가 빠르다
④일반 스레드는 메인 스레드에서 파생되어 자식으로 언급되는 스레드나 다른 어플리케이션의 스레드를 말한다

118. 다음 중 설명이 달리 된 것은?
①자식 스레드는 작업 요청을 위해 sendMessage()또는 postMessage() 메소드를 이용하여 메시지나 Runnable객체를 메시지 큐에 보내 작업을 요청하고 핸들러는 handleMessage()를 통해 메시지의 내용을 읽어 요청한 작업을 수행한다
②메시지 작업을 요청하기 위해서 위에서 전달하는 메시지에 의해 사전 정의된 상수값 또는 Runnalbe객체이다
③전달된 메시지는 메시지 큐를 통해서 쌓게되고 핸들러는 순차적으로 가져다 사용하게 된다
④메시지를 전달하기 위해서 핸들러의 obtainMessage()를 호출해서 메시지 풀내 메시지를 획득하고 sendMessage()를 통해 전달한다