CS353 Linux Kernel

Chentao Wu吴晨涛
Associate Professor
Dept. of CSE, SJTU
wuct@cs.sjtu.edu.cn

Google™

Android Introduction

Platform Overview

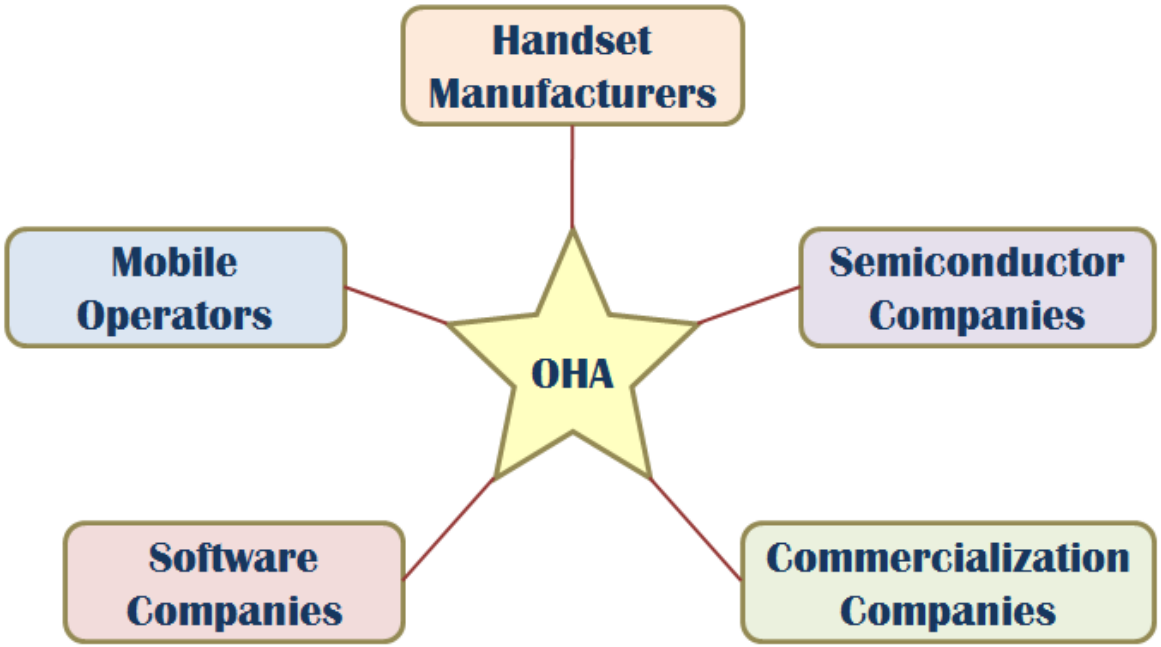
上海交通大學

What is Android?

- Android is a software stack for mobile devices that includes an operating system, middleware and key applications.

OHA (Open Handset Alliance)

- A business alliance consisting of 47 companies to develop open standards for mobile devices



Phones

HTC G1,
Droid,
Tattoo

Motorola Droid (X)

Suno S880

Samsung Galaxy

Sony Ericsson

Tablets

Velocity Micro Cruz

Gome FlyTouch

Acer beTouch

Dawa D7

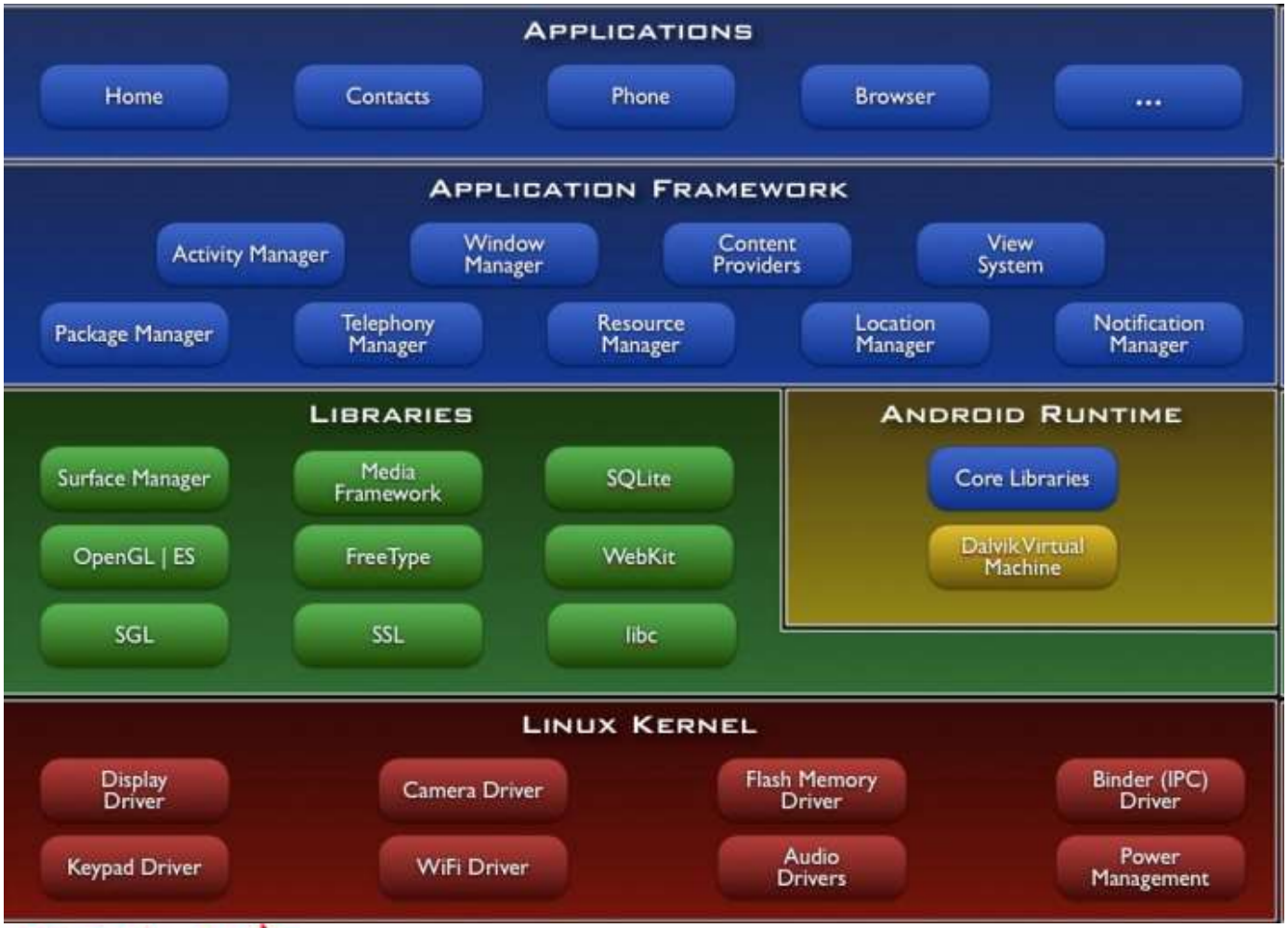
Toshiba Android SmartBook

Cisco Android Tablet

MarketShare

	Feb'10	May'10	Apr'11
RIM	42.1%	41.7%	29%
Apple	25.4%	24.4%	25%
Google	9%	13%	33%
Microsoft	15.1%	13.2%	7.7%
Palm	5.4%	4.8%	2.9%

Architecture



Android S/W Stack - Application

- Android provides a set of core applications:
  - ✓ Email Client
  - ✓ SMS Program
  - ✓ Calendar
  - ✓ Maps
  - ✓ Browser
  - ✓ Contacts
  - ✓ Etc

- All applications are written using the Java language.

Android S/W Stack – App Framework

- Enabling and simplifying the reuse of components
  - ✓ Developers have full access to the same framework APIs used by the core applications.
  - ✓ Users are allowed to replace components.

Android S/W Stack – App Framework (Cont)

- Features

Feature	Role
View System	Used to build an application, including lists, grids, text boxes, buttons, and embedded web browser
Content Provider	Enabling applications to access data from other applications or to share their own data
Resource Manager	Providing access to non-code resources (localized strings, graphics, and layout files)
Notification Manager	Enabling all applications to display customer alerts in the status bar
Activity Manager	Managing the lifecycle of applications and providing a common navigation backstack

上海交通大學

Android S/W Stack - Libraries

- Including a set of C/C++ libraries used by components of the Android system
- Exposed to developers through the Android application framework

Android S/W Stack - Runtime

- Core Libraries
  - ✓ Providing most of the functionality available in the core libraries of the Java language
  - ✓ APIs
    - Data Structures
    - Utilities
    - File Access
    - Network Access
    - Graphics
    - Etc

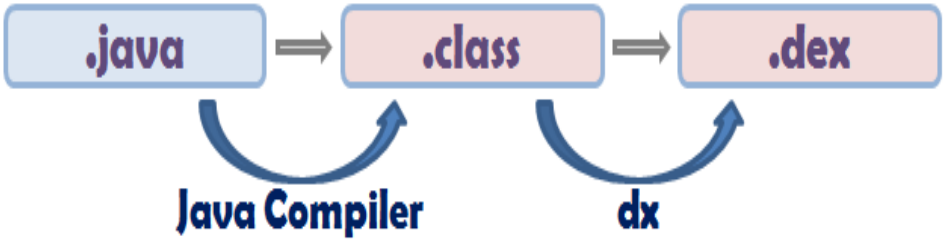
上海交通大學

Android S/W Stack – Runtime (Cont)

- Dalvik Virtual Machine
  - ✓ Providing environment on which every Android application runs
    - Each Android application runs in its own process, with its own instance of the Dalvik VM.
    - Dalvik has been written such that a device can run multiple VMs efficiently.
  - ✓ Register-based virtual machine

Android S/W Stack – Runtime (Cont)

- Dalvik Virtual Machine (Cont)
  - ✓ Executing the Dalvik Executable (.dex) format
    - .dex format is optimized for minimal memory footprint.
    - Compilation



  - ✓ Relying on the Linux Kernel for:
    - Threading
    - Low-level memory management

Android S/W Stack – Linux Kernel

- Relying on Linux Kernel 2.6 for core system services
  - ✓ Memory and Process Management
  - ✓ Network Stack
  - ✓ Driver Model
  - ✓ Security
- Providing an abstraction layer between the H/W and the rest of the S/W stack

Google™

Android Introduction

Application Fundamentals

上海交通大學

Goal

- Understand applications and their components
- Concepts:
  - activity,
  - service,
  - broadcast receiver,
  - content provider,
  - intent,
  - AndroidManifest

上海交通大學

Applications

- Written in Java (it's possible to write native code – will not cover that here)
- Good separation (and corresponding security) from other applications:
  - Each application runs in its own process
  - Each process has its own separate VM
  - Each application is assigned a unique Linux user ID – by default files of that application are only visible to that application (can be explicitly exported)

上海交通大學

Application Components

- Activities – visual user interface focused on a single thing a user can do
- Services – no visual interface – they run in the background
- Broadcast Receivers – receive and react to broadcast announcements
- Content Providers – allow data exchange between applications

上海交通大學

Activities

- Basic component of most applications
- Most applications have several activities that start each other as needed
- Each is implemented as a subclass of the base Activity class

上海交通大學

Activities – The View

- Each activity has a default window to draw in (although it may prompt for dialogs or notifications)
- The content of the window is a view or a group of views (derived from View or ViewGroup)
- Example of views: buttons, text fields, scroll bars, menu items, check boxes, etc.
- View(Group) made visible via Activity.setContentView() method.

上海交通大學

Services

- Does not have a visual interface
- Runs in the background indefinitely
- Examples
  - Network Downloads
  - Playing Music
  - TCP/UDP Server
- You can bind to a an existing service and control its operation

上海交通大學

Broadcast Receivers

- Receive and react to broadcast announcements
- Extend the class BroadcastReceiver
- Examples of broadcasts:
  - Low battery, power connected, shutdown, timezone changed, etc.
  - Other applications can initiate broadcasts

上海交通大學

Content Providers

- Makes some of the application data available to other applications
- It's the only way to transfer data between applications in Android (no shared files, shared memory, pipes, etc.)
- Extends the class ContentProvider;
- Other applications use a ContentResolver object to access the data provided via a ContentProvider

上海交通大學

Intents

- An intent is an Intent object with a message content.
- Activities, services and broadcast receivers are started by intents. ContentProviders are started by ContentResolvers:
  - An activity is started by Context.startActivity(Intent intent) or Activity.startActivityForResult(Intent intent, int RequestCode)
  - A service is started by Context.startService(Intent service)
  - An application can initiate a broadcast by using an Intent in any of Context.sendBroadcast(Intent intent), Context.sendOrderedBroadcast(), and Context.sendStickyBroadcast()

上海交通大學

Shutting down components

- Activities
  - Can terminate itself via finish();
  - Can terminate other activities it started via finishActivity();
- Services
  - Can terminate via stopSelf(); or Context.stopService();
- Content Providers
  - Are only active when responding to ContentResolvers
- Broadcast Receivers
  - Are only active when responding to broadcasts

上海交通大學

Android Manifest

- Its main purpose in life is to declare the components to the system:

```
<?xml version="1.0" encoding="utf-8"?>
<manifest . . . >
    <application . . . >
        <activity
android:name="com.example.project.FreneticActivity"
                android:icon="@drawable/small_pic.png"
                android:label="@string/freneticLabel"
                . . . >
        </activity>
        . . .
    </application>
</manifest>
```

Intent Filters

- Declare Intents handled by the current application (in the AndroidManifest):

```
<?xml version="1.0" encoding="utf-8"?>
 <manifest . . . >
    <application . . . >
       <activity android:name="com.example.project.FreneticActivity"
                 android:icon="@drawable/small_pic.png"
                 android:label="@string/freneticLabel"
                 . . . >
          <intent-filter . . . >
             <action android:name="android.intent.action.MAIN" />
             <category android:name="android.intent.category.LAUNCHER" />
          </intent-filter>
          <intent-filter . . . >
             <action android:name="com.example.project.BOUNCE" />
             <data android:mimeType="image/jpeg" />
             <category android:name="android.intent.category.DEFAULT" />
          </intent-filter>
       </activity>
       . . .
    </application>
 </manifest>
```

Shows in the Launcher and is the main activity to start

Handles JPEG images in some way

Google™

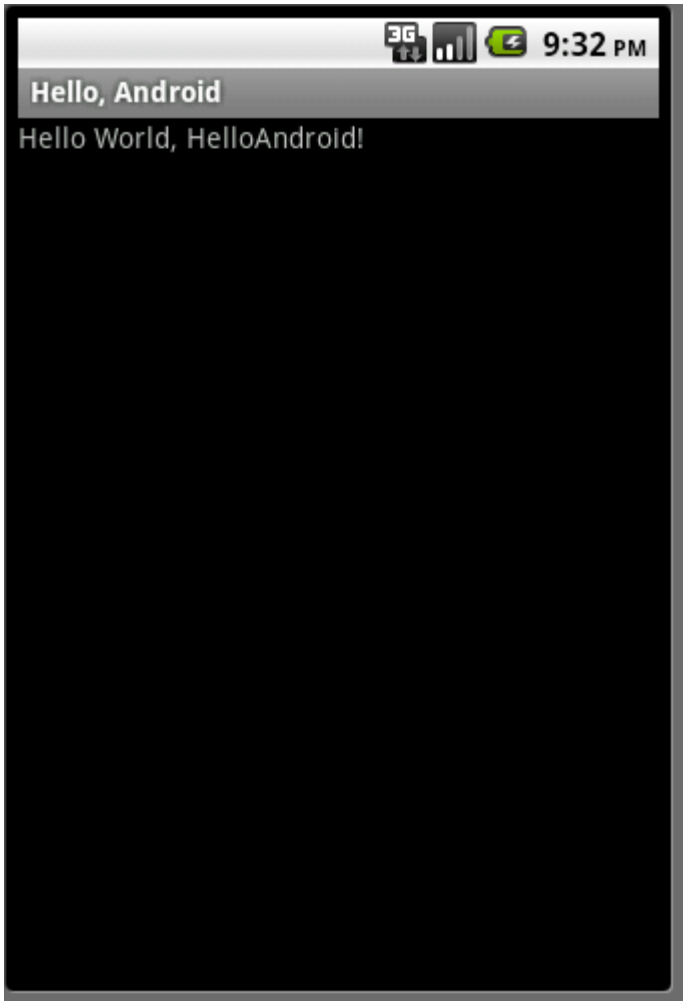
Android Introduction

Hello World

上海交通大學

Goal

- Create a very simple application
- Run it on a real device
- Run it on the emulator
- Examine its structure



Google Tutorial

- We will follow the tutorial at:

http://developer.android.com/resources/tutorials/hello-world.html

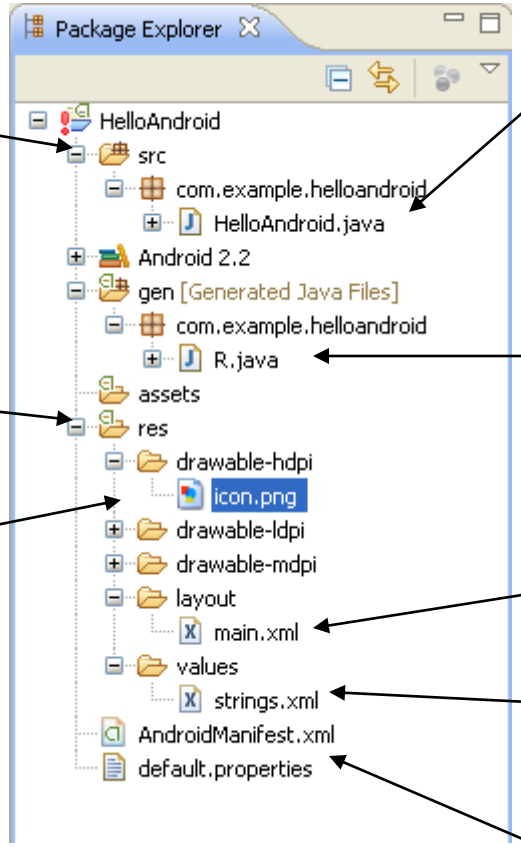
- Start Eclipse (Start -> All Programs -> Eclipse)
- Create an Android Virtual Device (AVD)
- Create a New Android Project

上海交通大學

Package Content

All source code here

Java code for our activity



All non-code resources

Generated Java code
Helps link resources to Java code

Images

Layout of the activity

Strings used in the program

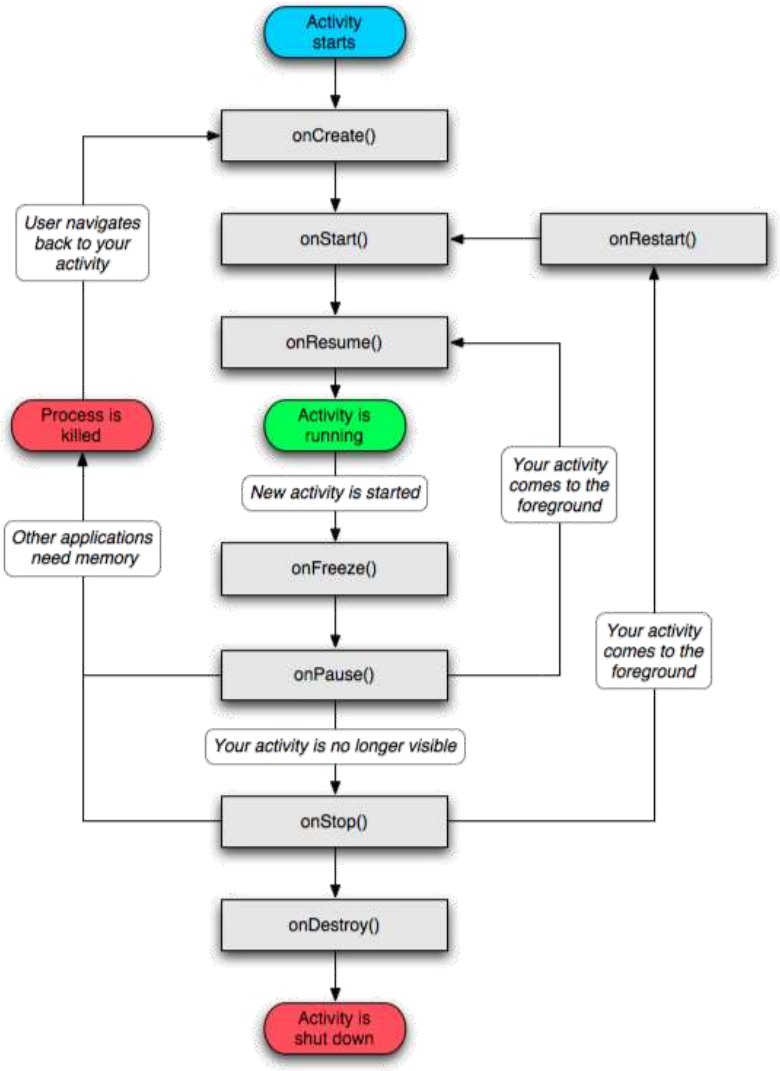
Android Manifest

Android Manifest

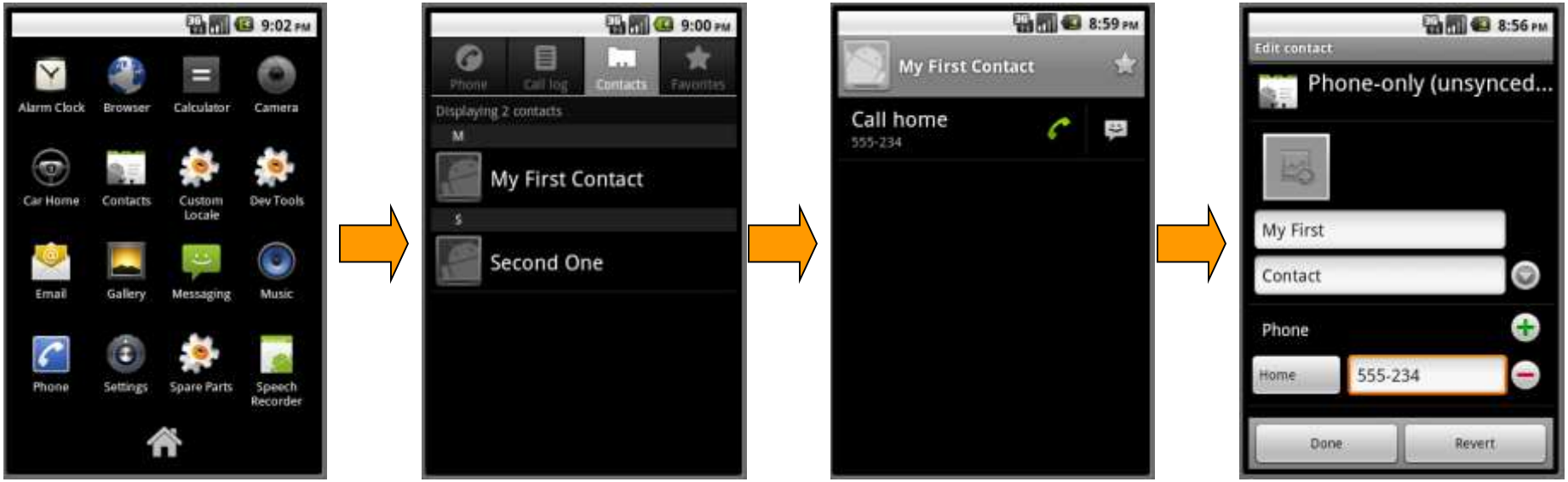
```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
      package="com.example.helloandroid"
      android:versionCode="1"
      android:versionName="1.0">
    <application android:icon="@drawable/icon" android:label="@string/app_name">
        <activity android:name=".HelloAndroid"
                  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>


</manifest>
```

Activity

- An Android activity is focused on a single thing a user can do.
- Most applications have multiple activities



Activities start each other



Revised HelloAndroid.java

```
package com.example.helloandroid;

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

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

Inherit from the Activity Class

Set the view "by hand" – from the program

上海交通大學

Run it!

/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"
    >
    <TextView
       android:layout_width="fill_parent"
       android:layout_height="wrap_content"
       android:text="@string/hello"
     />
</LinearLayout>
```

Further redirection to /res/values/strings.xml

/res/values/strings.xml

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

HelloAndroid.java

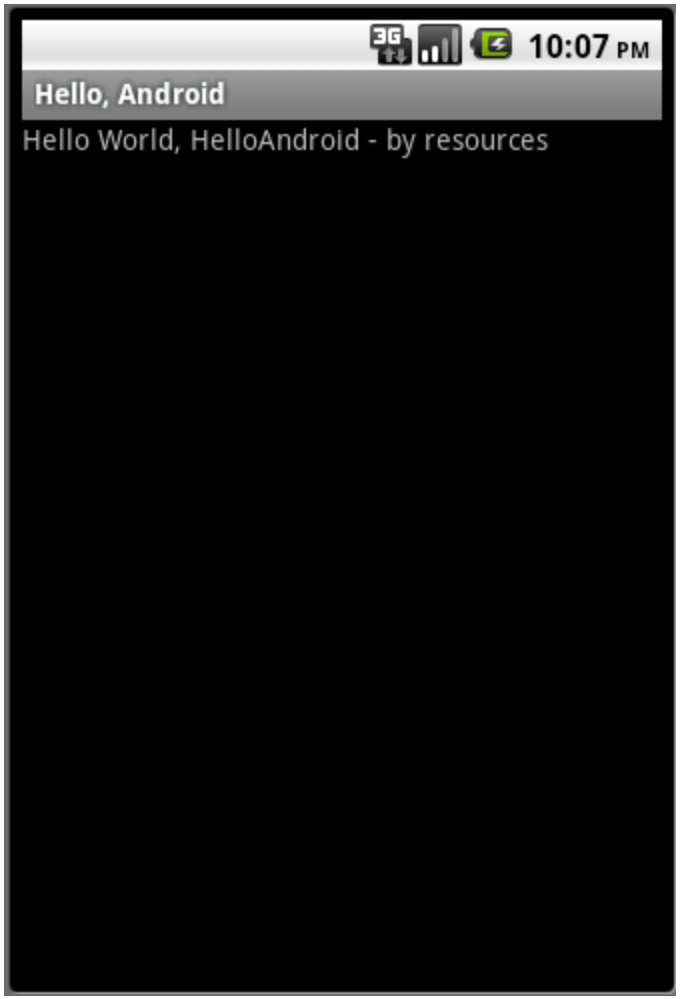
```
package com.example.helloandroid;

import android.app.Activity;
import android.os.Bundle;
public class HelloAndroid extends Activity {

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

Set the layout of the view as described in the main.xml layout

/gen/R.java

```
package com.example.helloandroid;

 public final class R {
    public static final class attr {
    }
    public static final class drawable {
        public static final int icon=0x7f020000;
    }
    public static final class id {
        public static final int textview=0x7f050000;
    }
    public static final class layout {
        public static final int main=0x7f030000;
    }
    public static final class string {
        public static final int app_name=0x7f040001;
        public static final int hello=0x7f040000;
    }
 }
```

Run it!


10:07 PM
Hello, Android
Hello World, HelloAndroid - by resources

Introduce a bug

```
package com.example.helloandroid;

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

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

上海交通大學

Run it!


10:11 PM
Alarm Clock
Browser
Calculator
Camera
Sorry!
The application Hello, Android (process com.example.helloandroid) has stopped unexpectedly. Please try again.
Force close
Music
Phone
Settings
Spare Parts

Google™

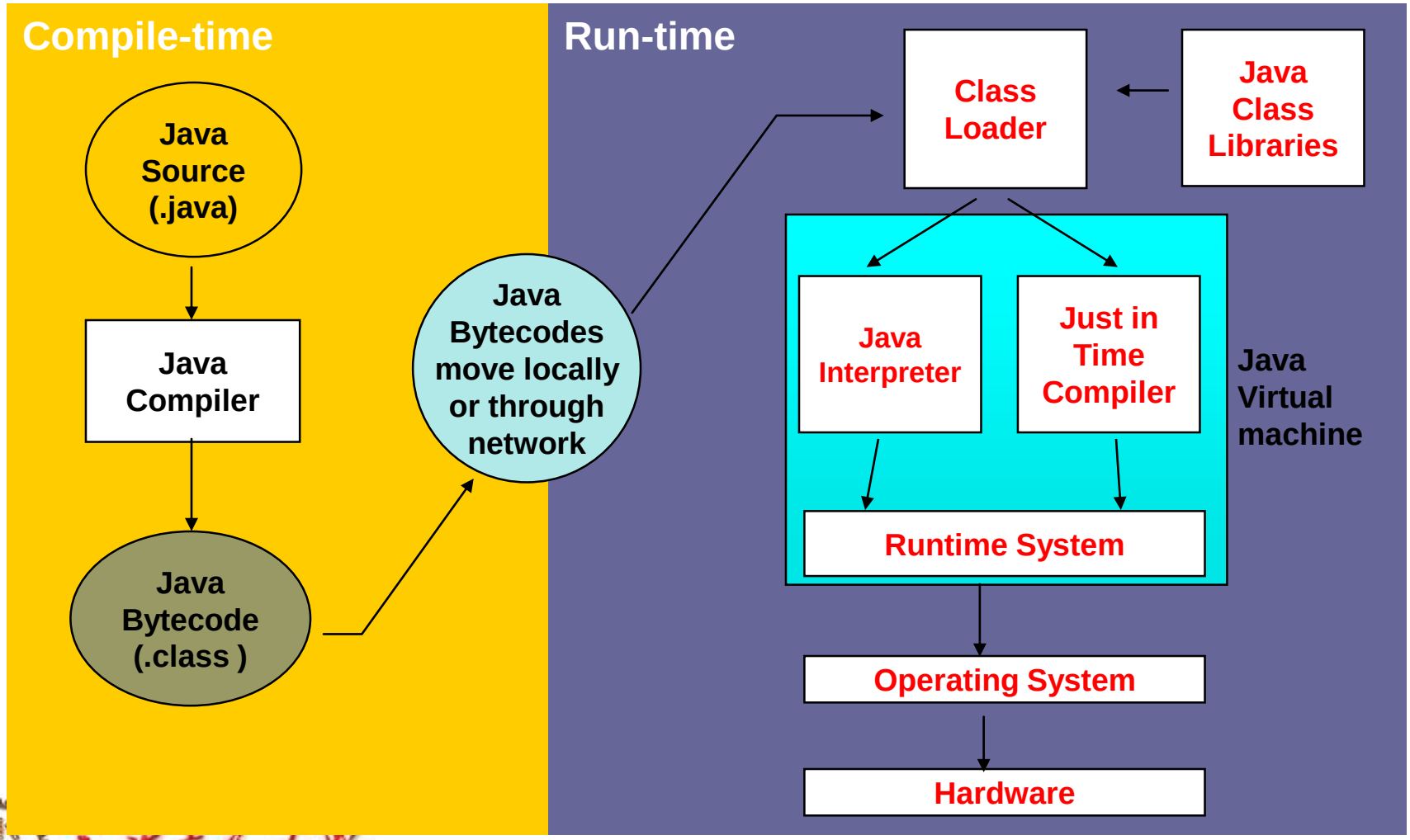
Java Review

上海交通大學

Java

- Java is an *object-oriented* language, with a syntax similar to C
  - Structured around *objects* and *methods*
  - A method is an action or something you do with the object
- Avoid those overly complicated features of C++:
  - Operator overloading, pointer, templates, friend class, etc.

How it works…!


Compile-time
Run-time
Java Source (.java)
Java Compiler
Java Bytecode (.class)
Java Bytecodes move locally or through network
Class Loader
Java Class Libraries
Java Interpreter
Just in Time Compiler
Java Virtual machine
Runtime System
Operating System
Hardware

Getting and using java

- JDK freely download from http://www.oracle.com
- All text editors support java
  - Vi/vim, emacs, notepad, wordpad
  - Just save to .java file
- Eclipse IDE
  - Eclipse
  - http://www.eclipse.org
  - Android Development Tools (ADT) is a plugin for Eclipse

上海交通大學

Compile and run an application

- Write java class `HolaWorld` containing a `main()` method and save in file "*`HolaWorld.java`*"
  - The file name *MUST* be the same as class name
- Compile with: `javac HolaWorld.java`
- Creates compiled .class file: *`HolaWorld.class`*
- Run the program: *`java HolaWorld`*
  - Notice: use the class name directly, no .class!

上海交通大學

Hola World!

File name: *HolaWorld.java*

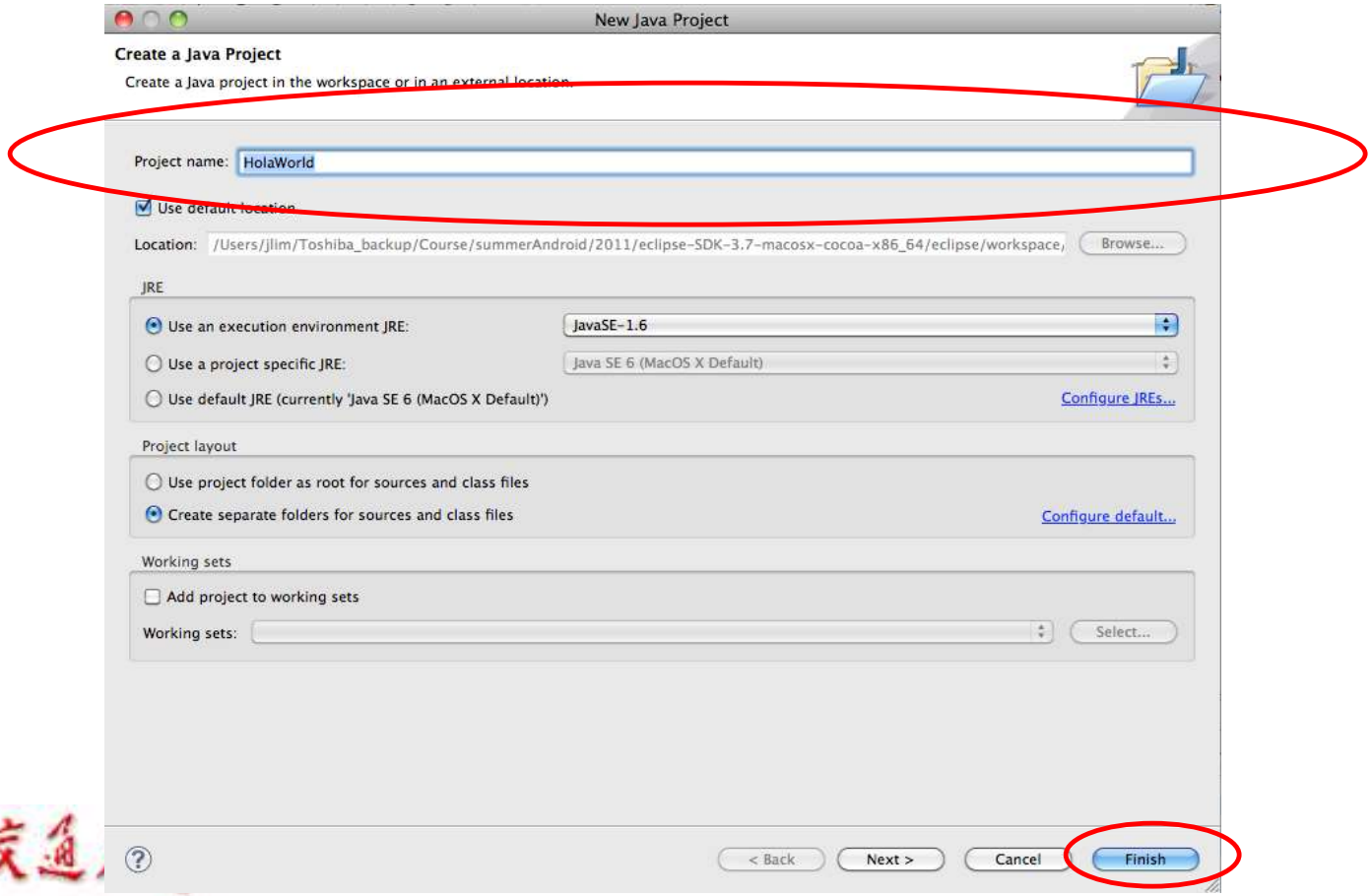
```
/* Our first Java program – HolaWorld.java */

public class HolaWorld {
      //main()
      public static void main ( String[] args )
{
            System.out.println( ”Hola world!"
);
      }
}
```

Command line arguments

Standard output, print with new line

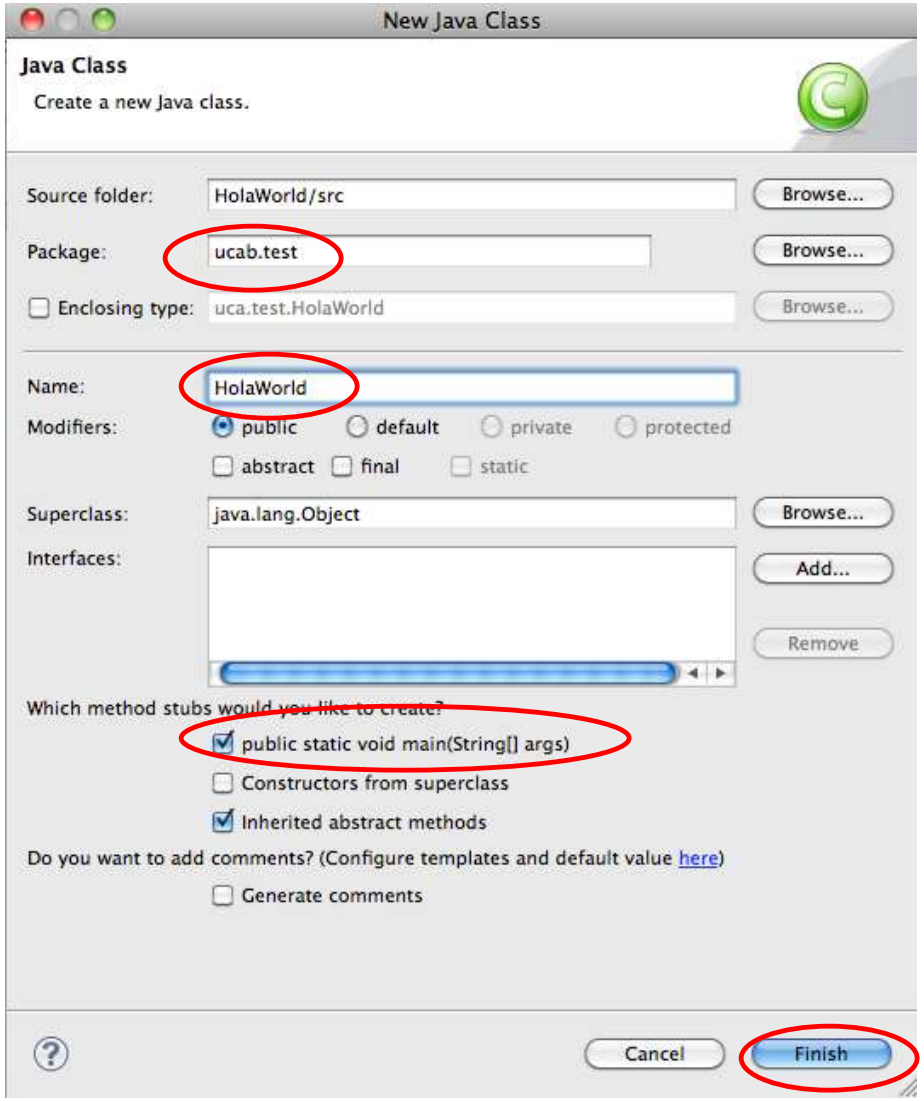
HolaWorld in Eclipse - create a new project

- File > New > Java Project
- Project Name : HolaWorld



HolaWorld in Eclipse – add a new class

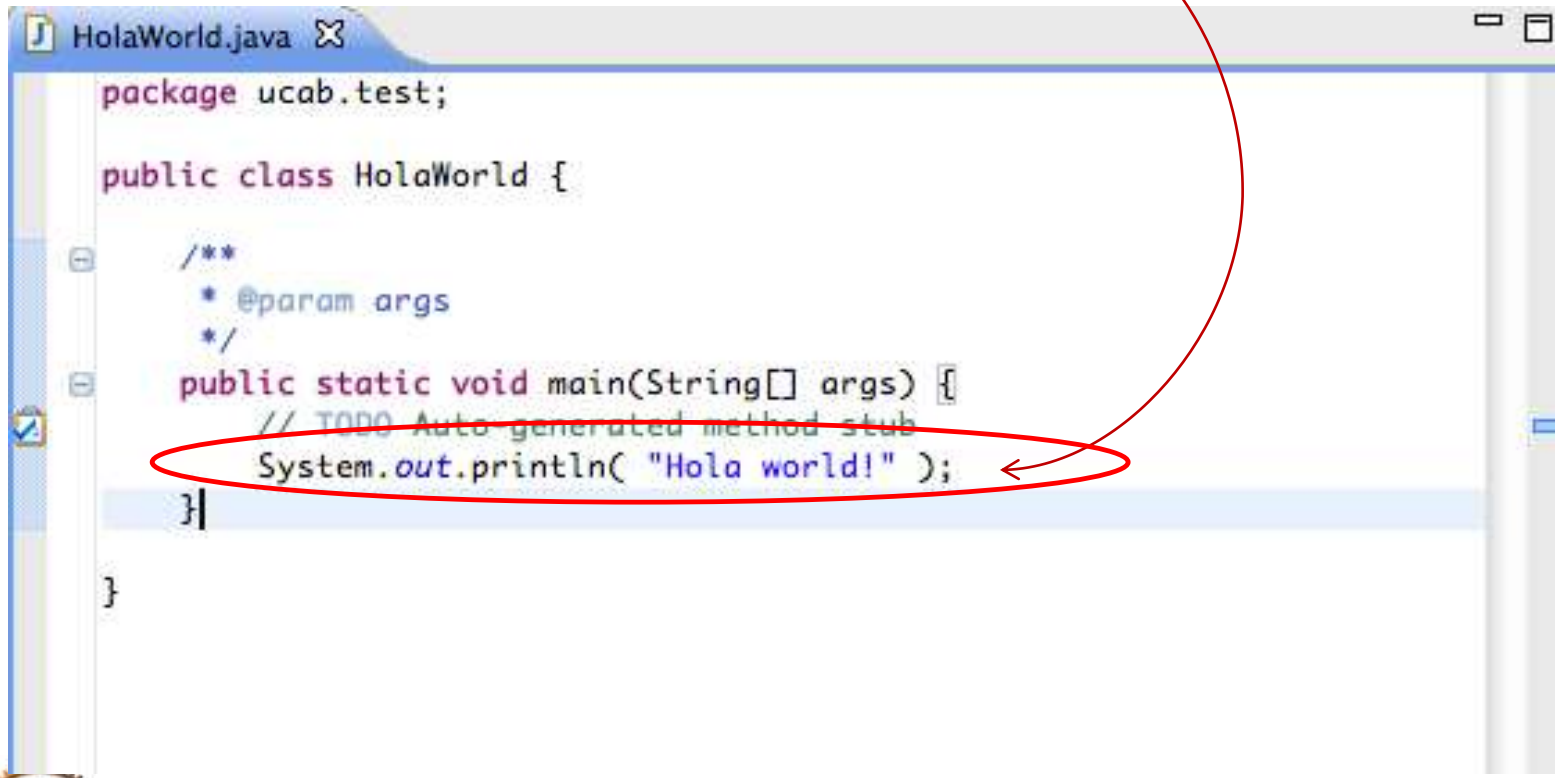
- File > New > Class
- source folder : HolaWorld/src
- Package : ucab.test
- Name : HolaWorld
- check "public static void main (String[] args)



HolaWorld in Eclipse – write your code

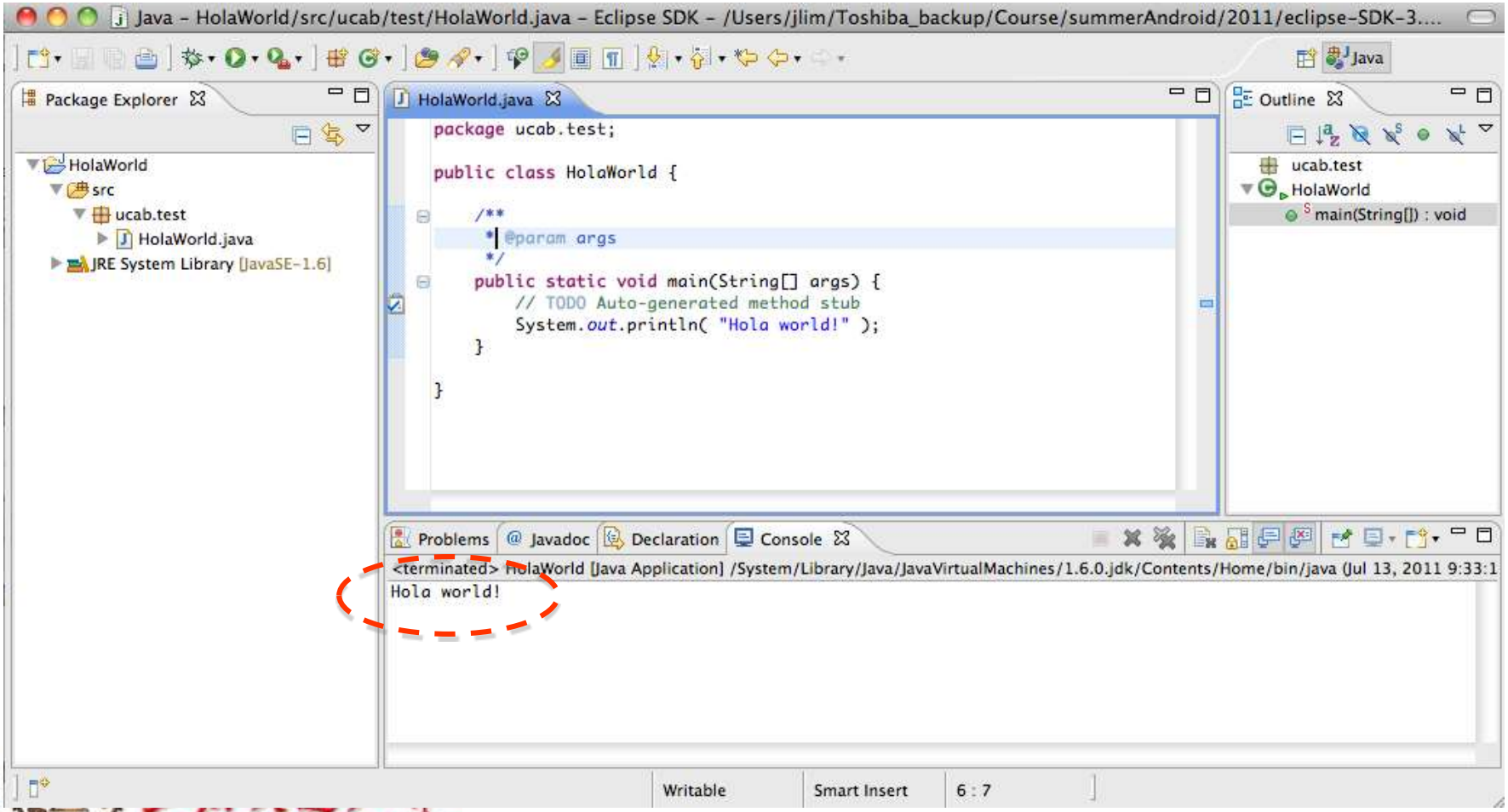
- Add your code

  System.out.println("Hola world!");



HolaWorld in Eclipse – run your program

- Run > Run As > Java Application



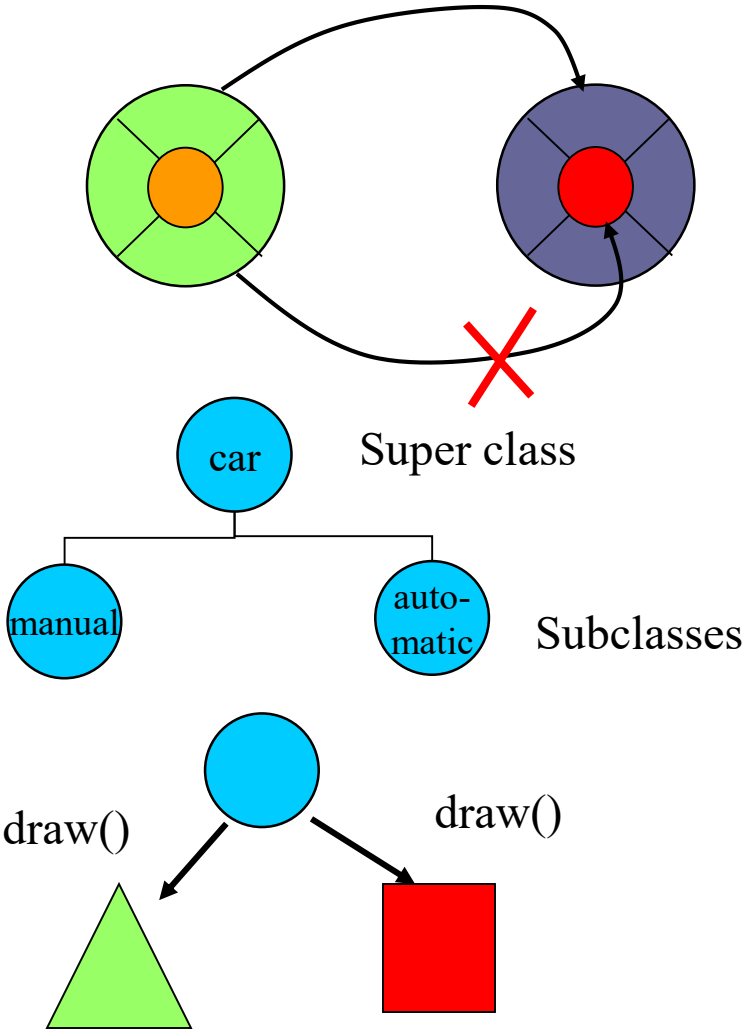
Object-Oriented

- Java supports OOP
  - Polymorphism
  - Inheritance
  - Encapsulation
- Java programs contain nothing but definitions and instantiations of classes
  - Everything is encapsulated in a class!

上海交通大學

The three principles of OOP

- Encapsulation
  - Objects hide their functions (**methods**) and data (**instance variables**)
- Inheritance
  - Each **subclass** inherits all variables of its **superclass**
- Polymorphism
  - Interface same despite different data types



About class

- Fundamental unit of Java program
- All java programs are classes
- A class is a template or blueprint for objects
- A class describes a set of objects that have identical characteristics (data elements) and behaviors (methods).
  - Existing classes provided by JRE
  - User defined classes
- Each class defines a set of fields (variables), methods or other classes

What is an object?

- An object is an instance of a class
- An object has state, behavior and identity
  - Internal variable: store state
  - Method: produce behavior
  - Unique address in memory: identity

上海交通大學

What does it mean to create an object?

- An object is a chunk of memory:
  - holds field values
  - holds an associated object type
- All objects of the same type share code
  - they all have same object type, but can have different field values.

Class Person: definition

```
class Person {
      String name;
      int height; //in inches
      int weight; //in pounds
      public void printInfo(){
        System.out.println(name+" with height="+height+",
weight="+weight);
      }
}
```

Variables

Method

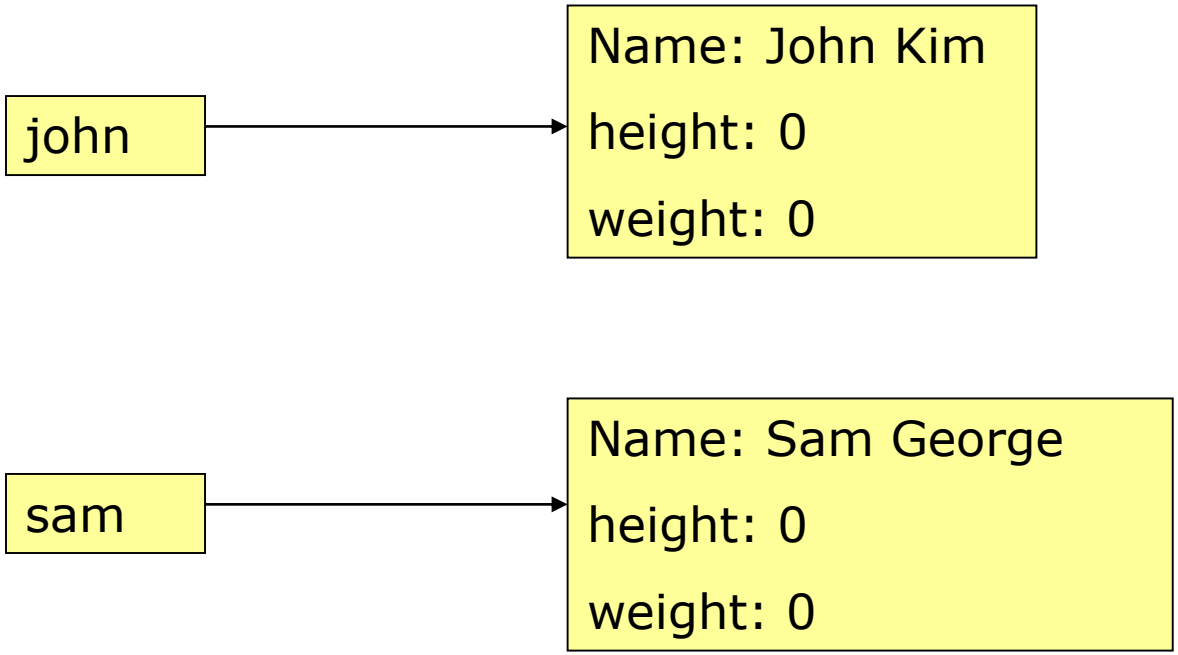
```
class ClassName{
        /* class body goes here */
}
```

class: keyword

Class Person: usage

```
Person john;          //declaration
john = new Person();//create an object of Person
john.name= "John Kim";//access its field
Person sam = new Person();
sam.name="Sam George";
john.printInfo();      // call method
sam.printInfo();
```

Class Person: reference



References

Objects allocated in memory

Reference

```
Person john;
```
//only created the reference, not an object. It points to nothing now (null).

```
john = new Person();
```
//create the object (allocate storage in memory), and john is initialized.

```
john.name="John";
```
//access the object through the reference

Primitive types

Primitive type	Size	Minimum	Maximum	Wrapper type
boolean	1-bit	—	—	**Boolean**
char	16-bit	Unicode 0	Unicode 2^{16}- 1	**Character**
byte	8-bit	-128	+127	**Byte**
short	16-bit	-2^{15}	$+2^{15}-1$	**Short**
int	32-bit	-2^{31}	$+2^{31}-1$	**Integer**
long	64-bit	-2^{63}	$+2^{63}-1$	**Long**
float	32-bit	IEEE754	IEEE754	**Float**
double	64-bit	IEEE754	IEEE754	**Double**

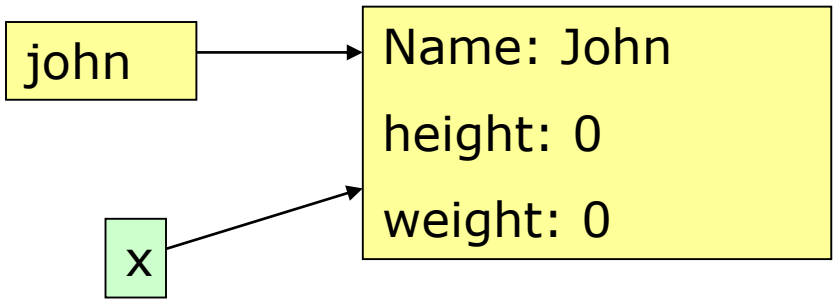
Reference vs. primitive

- Java handle objects and arrays always by reference.
  - classes and arrays are known as reference types.
  - Class and array are composite type, don't have standard size
- Java always handle values of the primitive types directly
  - Primitive types have standard size, can be stored in a fixed amount of memory
- Because of how the primitive types and objects are handles, they behave different in two areas: copy value and compare for equality

上海交通大學

Copy

- Primitive types get copied directly by =
  - int x= 10; int y=x;
- Objects and arrays just copy the reference, still only one copy of the object existing.




```
Person john = new Person();
john.name=”John";
Person x=john;
x.name="Sam";
System.out.println(john.name);   // print Sam!
```

Scoping: in a class

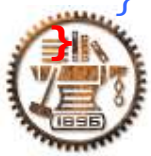
```
public class VisibilityDemo {
    private int classVar1;
    private int classVar2;
    public int method1(int x) {
        int local = 0;
        for (int i = 0; i < x; i++){
            local += i;
        }
        return local;
    }

    public void method2 ( int x) {
        classVar1 = x + 10;
        classVar2 = method1(classVar2);
    }
```

The red identifiers denote class varialbes and methods. They have visibility anywhere inside the outermost pair of red curly brackets

The blue identifiers are local to a single block (identified by blue brackets). They are not visible to anything outside of their block, but are visible inside blocks nested inside of the blue bracketed block.

The gray identifiers are found inside the for-loop. The gray variable *i* is visible only inside the loop.

Parameters are denoted by green. They are visible everywhere inside the method in which they appear, but only in that method



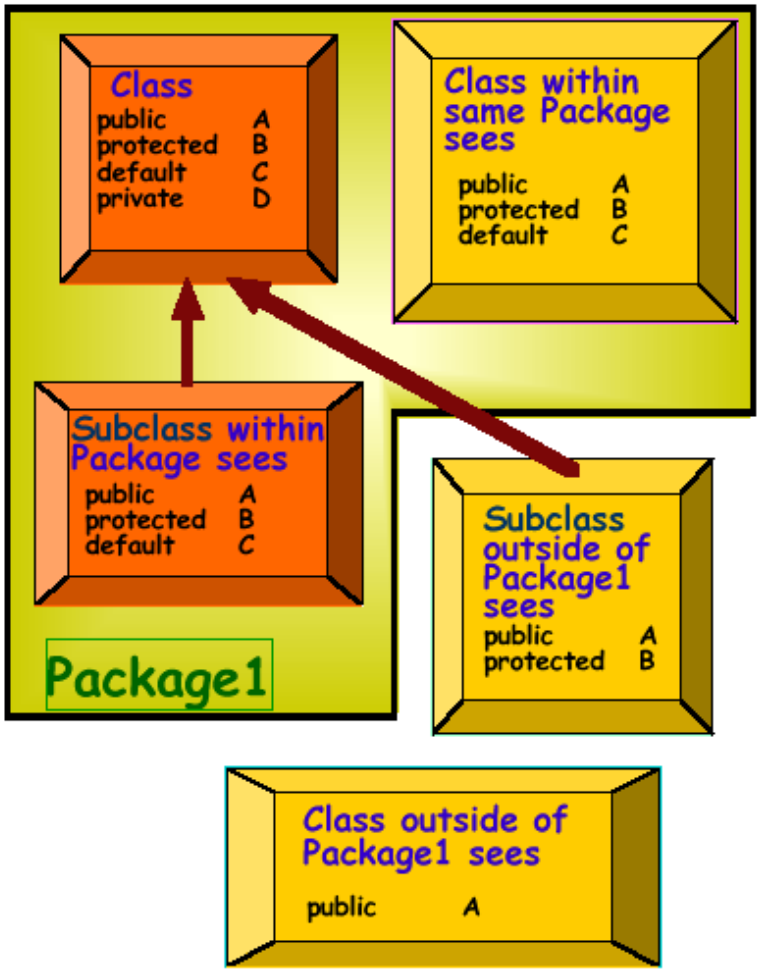
上海交通大學

Access control

- Access to packages
  - Java offers no control mechanisms for packages.
  - If you can find and read the package you can access it
- Access to classes
  - All top level classes in package P are accessible anywhere in P
  - All public top-level classes in P are accessible anywhere
- Access to class members (in class C in package P)
  - Public: accessible anywhere C is accessible
  - Protected: accessible in P and to any of C's subclasses
  - Private: only accessible within class C
  - Package: only accessible in P (the default)

上海交通大學

Scoping: visibility between classes



A summary of Java scoping visibility

The static keyword

- Java methods and variables can be declared static
- These exist **independent of any object**
- This means that a Class's
  - static methods can be called even if no objects of that class have been created and
  - static data is "shared" by all instances (i.e., one rvalue per class instead of one per instance

```
class StaticTest {static int i = 47;}
StaticTest st1 = new StaticTest();
StaticTest st2 = new StaticTest();
// st1.i == st2.I == 47
StaticTest.i++;    // or st1.I++ or st2.I++
// st1.i == st2.I == 48
```

上海交通大学

Google™

XML Review

上海交通大學

XML

- eXtensible Markup Language
- Simple text (Unicode) underneath
- Tags (like in HTML) are used to provide information about the data
- Similar to HTML, but:
  - HTML is used to describe how to display the data
  - XML is used to describe what is the data
- Often used to store and transfer data

HTML Example

```
<html>
   <head><title>Here goes the
   title</title></head.
<body>
   <h1>This is a header</h1>
    Here goes the text of the page
</body>
</html>
```

- Tags mean something specific to the browser
- They are used for display

XML Example

```
<?xml version="1.0"/>
<person>
   <name>
      <first>Jose</first>
      <last>Barrios</last>
   </name>
   <email>jb@ucab.edu</email>
   <phone 555-456-1234 />
</person>
```

- Tags mean whatever the user wants them to mean
- They are used to describe the data

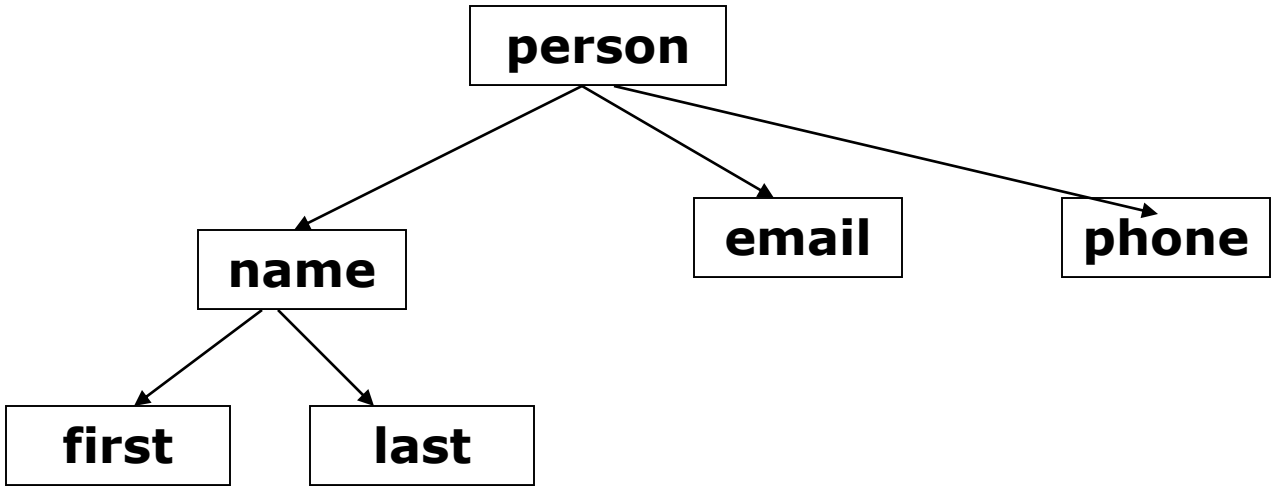
上海交通大學

XML Rules

- Tags are enclosed in angle brackets.
- Tags come in pairs with start-tags and end-tags.
- Tags must be properly nested.
  - **<name><email>…</name></email>** is not allowed.
  - **<name><email>…</email><name>** is.
- Tags that do not have end-tags must be terminated by a '/'.
- Document has a single root element

上海交通大學

XML Documents are Trees



Android Manifest

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.helloandroid"
    android:versionCode="1"
    android:versionName="1.0">
  <application android:icon="@drawable/icon" android:label="@string/app_name">
    <activity android:name=".HelloAndroid"
              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>


</manifest>
```

Attributes

Questions?