Android Applications
Application Fundamentals

- Android apps are generally written in Java
- Bundled into .apk packages by aapt tool
- .apk packages can then be loaded onto devices
- All code in package comprises an application
Running an Application

• Apps usually runs in their own Linux processes
• Android manages process creation & shutdown
  – Starts process when any of the application's code needs to be executed
  – Shuts down when process is no longer needed and system resources are required by other applications
• Each process runs its own virtual machine (VM)
• Each application is usually assigned a unique Linux user ID
Application Components

• Apps can have multiple entry points
• Each app comprised of components that the system can instantiate and run as needed
• Component types include:
  – Activities
  – Services
  – Broadcast receivers
  – Content providers
Activity

• Primary class for interacting with user
  – Usually implements a focused task
  – Involving one screenful of data

• Example:
  – Calculator
Service

• Runs in the background for indefinite periods of time
• Does not have a visual user interface
• Example
  – Music playing application
Broadcast Receivers

- Run in the background listening for events
  - called intents
- Does not have a visual user interface
- Example
  - Alarm manager
Content Providers

- Store & retrieve data across application boundaries
- Uses database-style interface
- Example
  - Contacts
Simple Application

- MapLocation
  - User enters an address
  - App display a corresponding map
App Organization

• Define resources
Resources

- Layout
- Strings
- Drawables
- Menus
- Other values
• UI Layout specified in XML
• Stored in res/layout/filename.xml
• Accessed from R.layout class
• With Eclipse can also do layout visually
<?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="wrap_content"
        android:layout_height="wrap_content"
        android:text="Enter Location"/>

    <EditText android:id="@+id/location"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content" />

    <Button android:id="@+id/mapButton"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Show Map" />
</LinearLayout>
Strings

• Types
  – String
  – String Array
  – Plurals

• Can include style and formatting
• Stored in res/values/filename.xml
  – Each string specified as
    • @string/string_name

• Accessed as R.string.string_name
• At compilation time, aapt generates the R class
  – App code uses the R class to get previously specified resources
public final class R {
    public static final class attr {
    }

    public static final class id {
        public static final int location=0x7f040000;
        public static final int mapButton=0x7f040001;
    }

    public static final class layout {
        public static final int main=0x7f030000;
    }
}
public class MapLocation extends Activity {
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);

        final EditText addressfield = (EditText) findViewById(R.id.location);
        final Button button = (Button) findViewById(R.id.mapButton);

        button.setOnClickListener(new Button.OnClickListener() {
            public void onClick(View v) {
                try {
                    String address = addressfield.getText().toString();
                    address = address.replace(' ', '+');
                    Intent geoIntent = new Intent(android.content.Intent.ACTION_VIEW,
                                                  Uri.parse("geo:0,0?q=\" + address));
                    startActivity(geoIntent);
                } catch (Exception e) {};
            }
        });
    }
}
AndroidManifest.xml

• Provides application information to system
  – Application Name
  – Components & component types
  – Permissions
  – API level
  – Libraries the application is linked against
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="edu.umd.cs.cmsc498g.SimpleActivity">
    <application>
        <activity android:name=".MapLocation" android:label="Map A Location">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>
</manifest>
• Components can communicate by sending and receiving Intent events
• From AndroidManifest.xml (intent filter)
  – <action android:name="android.intent.action.MAIN" />
  – <category
    android:name="android.intent.category.LAUNCHER" />
• Specifies that the MapLocation Activity is the entry point for the application, should appear in the application launcher
Tasks

• A chain of related Activities is called a task
• The task’s Activity objects are stored on a stack
  – The currently running Activity is at the top
• At runtime,
  – Newly started activities are pushed onto stack
  – Hitting the BACK button pops current activity off the stack
• Tasks give the illusion that multiple, unrelated Activities are part of the same application
Component Lifecycles

- Android can pause or terminate individual components. For example when:
  - Task stack changes
  - Memory gets low
  - User stops interacting with the application
  - New application is launched

- At these times, Android notifies applications by calling their lifecycle methods
  - Each component type has its own lifecycle
  - Will discuss more in later classes