CS260 Intro to Java & Android 05.Android UI(Part I)

Winter 2015

User Interface

- UIs in Android are built using View and ViewGroup objects
- A View is the base class for subclasses called “widgets”
- widget is a fully implemented UI object
- widget examples include
  - text field
  - button
  - textbox

View Class

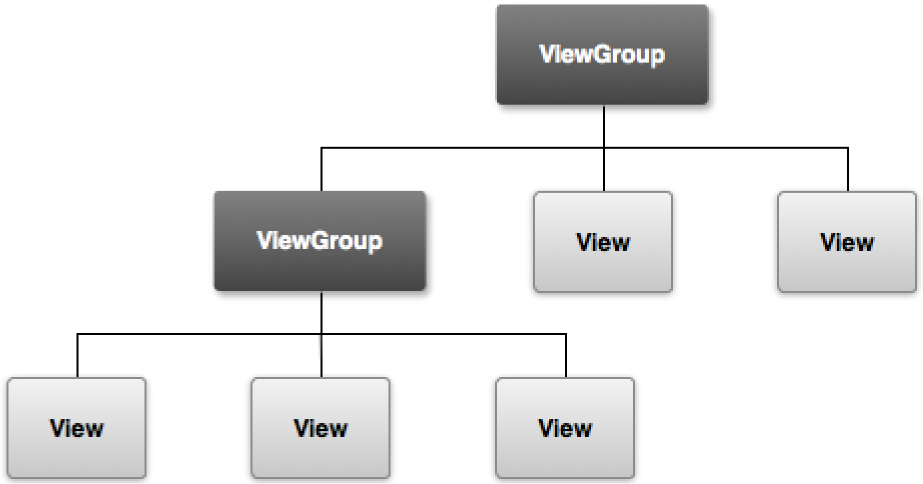
- A View class is the basic building block for UI components
- A View
  - is an object that draws something on the screen
  - occupies a rectangular area on the screen
  - has measurement information
  - has layout information
  - has drawing information
  - handles events such as scrolling & key interactions

ViewGroup Class

- A ViewGroup
  - extends a View
  - can contain other View (and ViewGroup) ojects (called children)
  - is the base class for layouts and view containers

View Hierarchy

- An Activity's UI is defined using View and ViewGroup objects
- The hierarchy tree can be complex or simple
- Design before implementing your UI



Using Views

- Views in a window are arranged in a single tree
- Views can be added
  - from code
  - from a view in an XML layout file
- Common operations on a tree of views
  - set properties (e.g. set the text of a TextView)
  - set the focus of a particular view
  - set up listeners for when something happens to a view object
  - set the visibility of a view object

setContentView

- The setContentView () method attaches the view hierarchy tree to the screen for rendering
- The root node requests that each child node draw itself
- Each ViewGroup requests that each child node draw itself

More View Hierarchy Facts

- children can make certain requests (e.g. size, location, …), but the parent has the final say

- Views are instantiated from the root node down the tree

- If elements overlap, the last element drawn is displayed

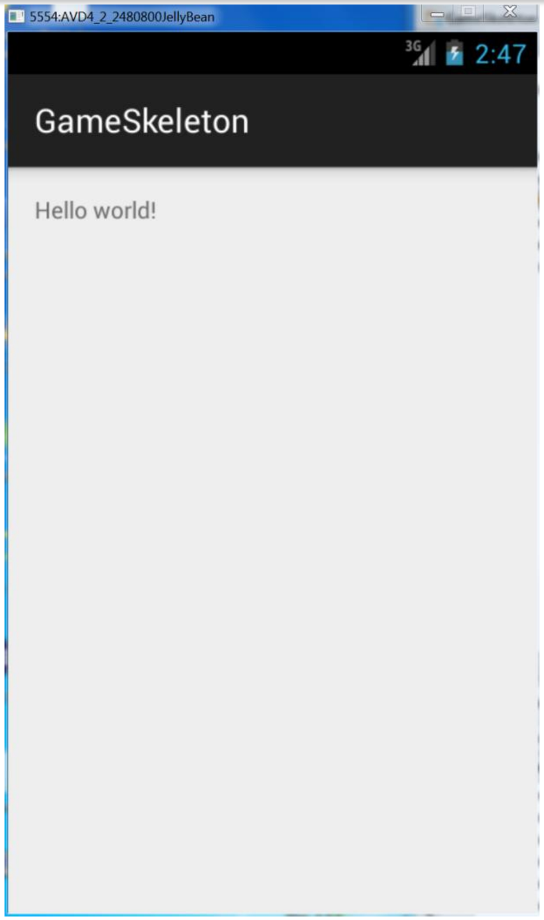
Android User Interfaces

- We are going to create the UI for a generic game
- The game will have:
  1. An App name GameSkeleton
  2. New Game (button)
  3. Continue (button)
  4. Rules (button)
  5. About (button)
  6. Exit (button)

Game Project

- Using AndroidStudio, create a game project called GameSkeleton
- Build the project
- Run the application in the AVD4.2.2 emulator

GameSkeleton Project Executed

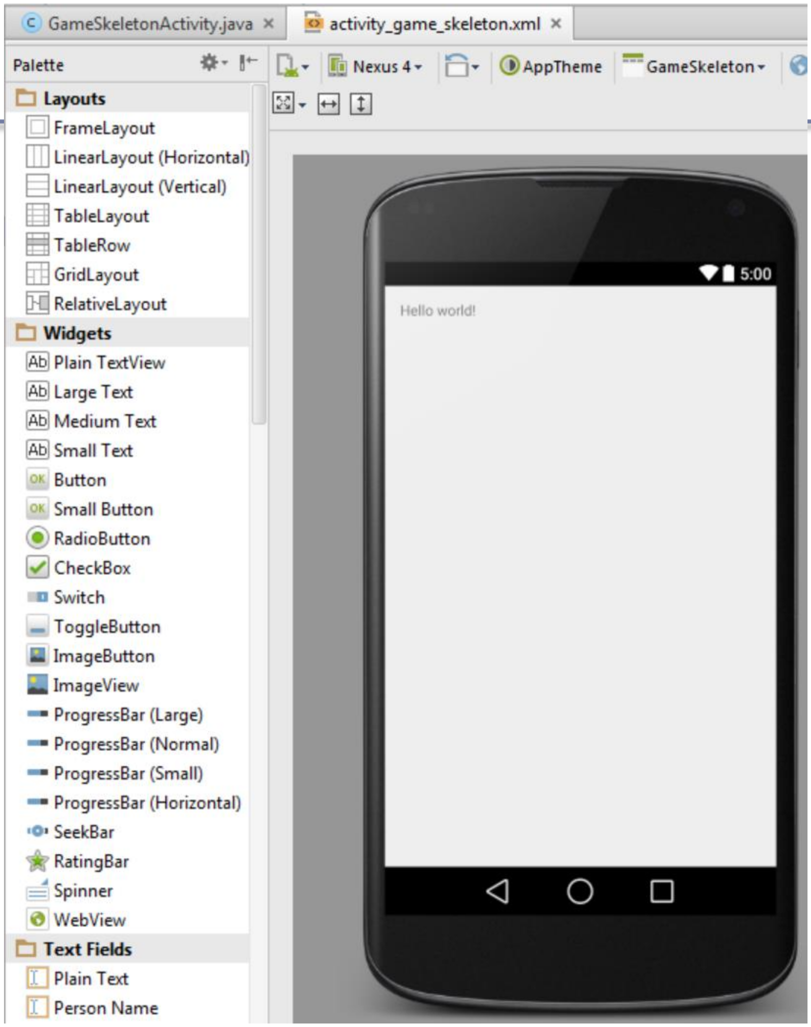


Hello world!

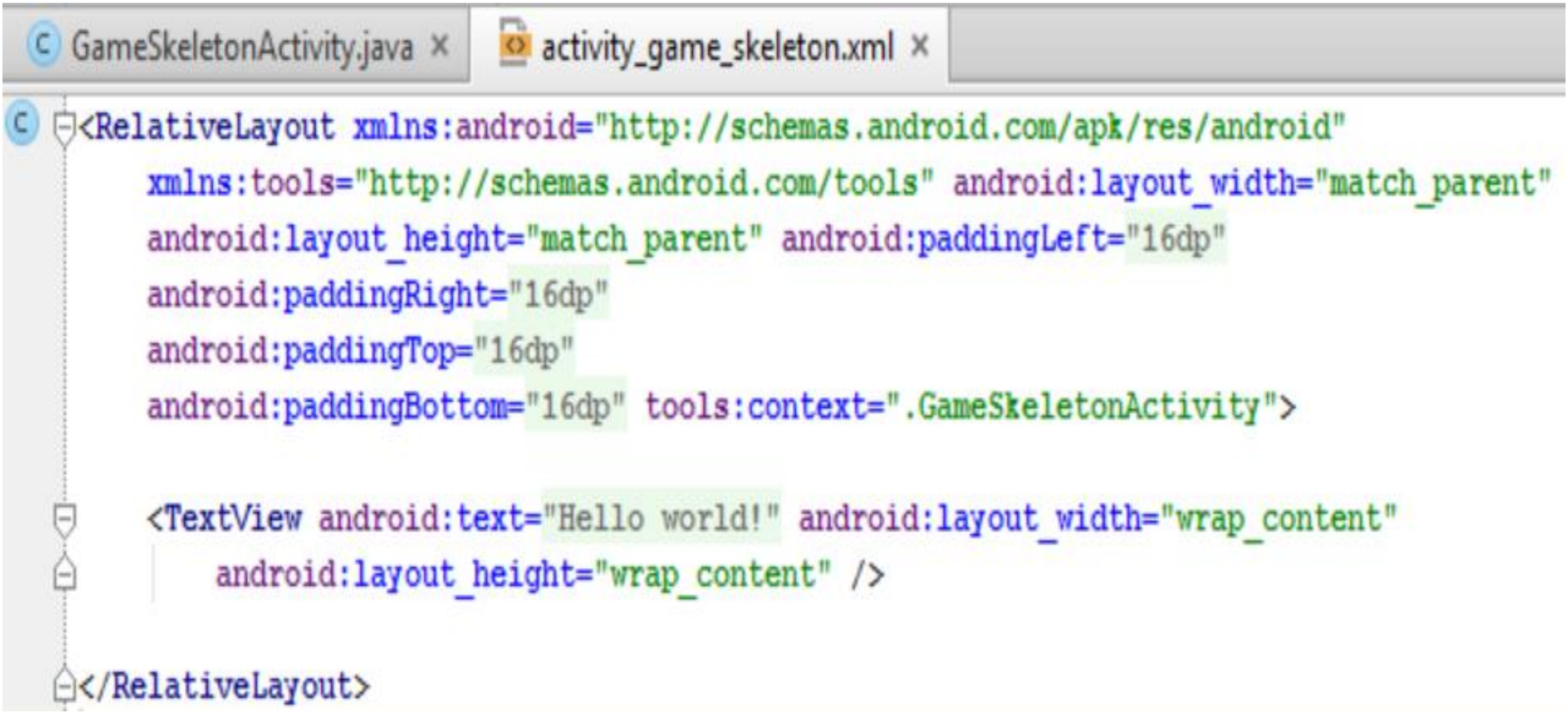
UI Design

- UIs can be designed in one of two ways
  - procedurally - meaning “ in code”
  - declaratively - meaning using some descriptive language (e.g. html, xml, …) and no code
- Our initial game will use a declarative approach

GameSkeletonActivity.xml Graphical Layout



GameSkeletonActivity.xml xml code



```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools" android:layout_width="match_parent"
    android:layout_height="match_parent" android:paddingLeft="16dp"
    android:paddingRight="16dp"
    android:paddingTop="16dp"
    android:paddingBottom="16dp" tools:context=".GameSkeletonActivity">

    <TextView android:text="Hello world!" android:layout_width="wrap_content"
        android:layout_height="wrap_content" />

</RelativeLayout>
```

Android's Use of XML

- XML is used when writing Android applications
- Android resource compiler (aapt) compiles xml code into a compressed binary format
- Compressed binary format stored on device, not xml code
- xml code (as compressed binary format) is instantiated (inflated) when necessary

Layout

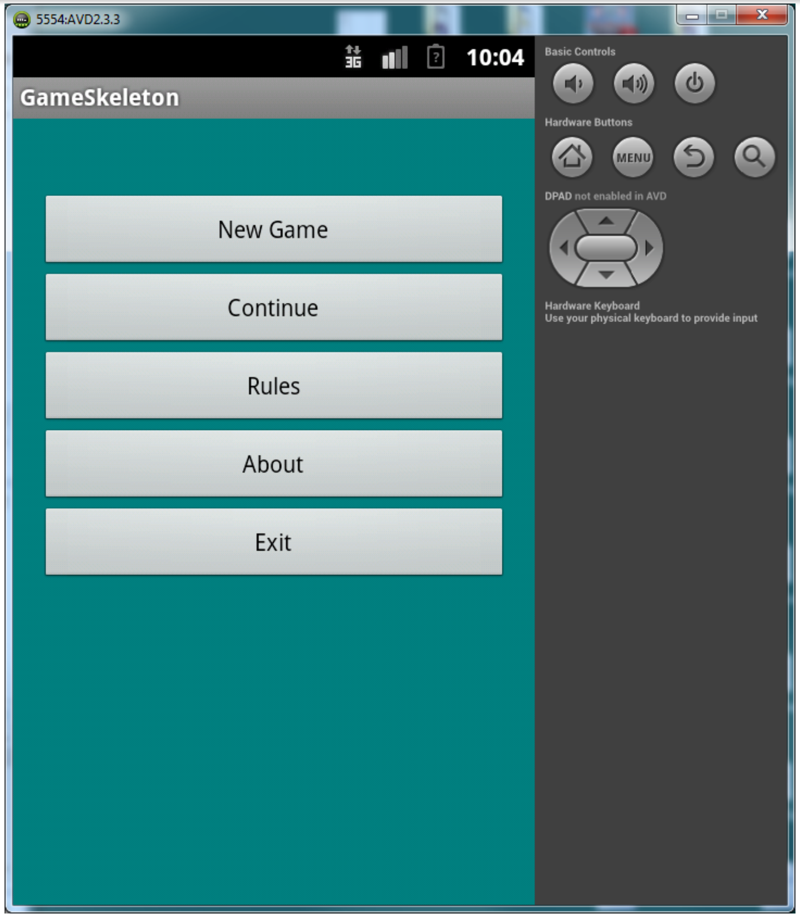
- What is a layout?
  - container for one or more child objects
  - behavior to position child objects on the screen
- Common layouts
  - FrameLayout
  - LinearLayout
  - RelativeLayout
  - TableLayout

Attributes

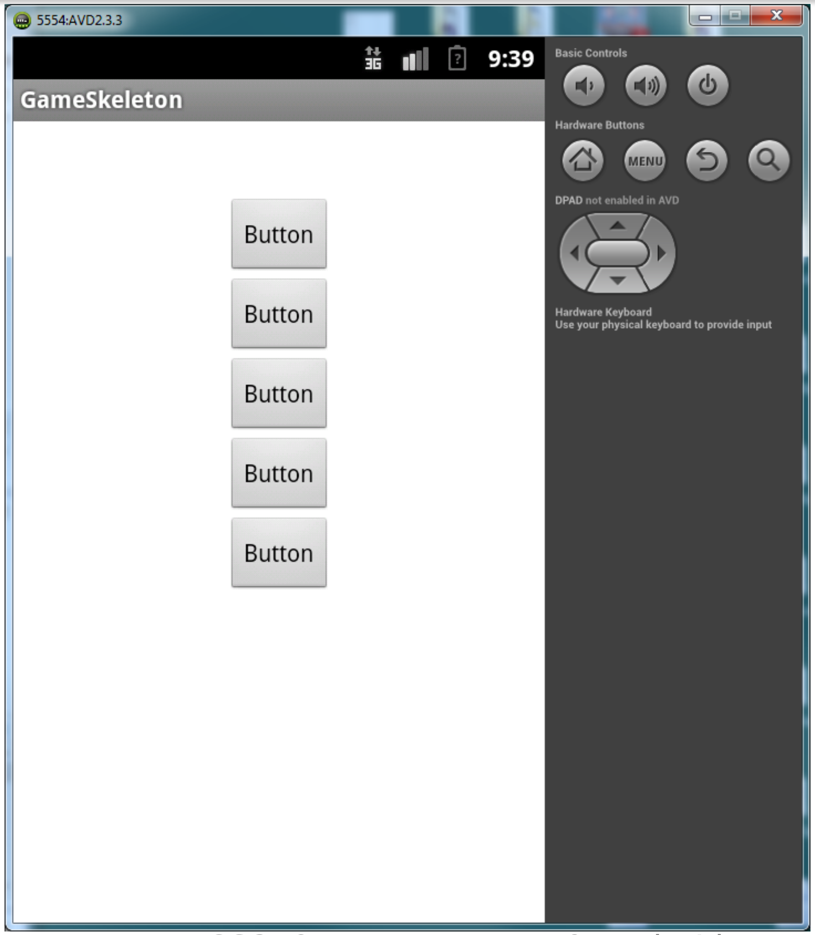
- Each View and ViewGroup object has a variety of XML attributes
  - Example: TextView has an attribute called textSize

- We will examine attributes in more detail after the following example

Create the following UI



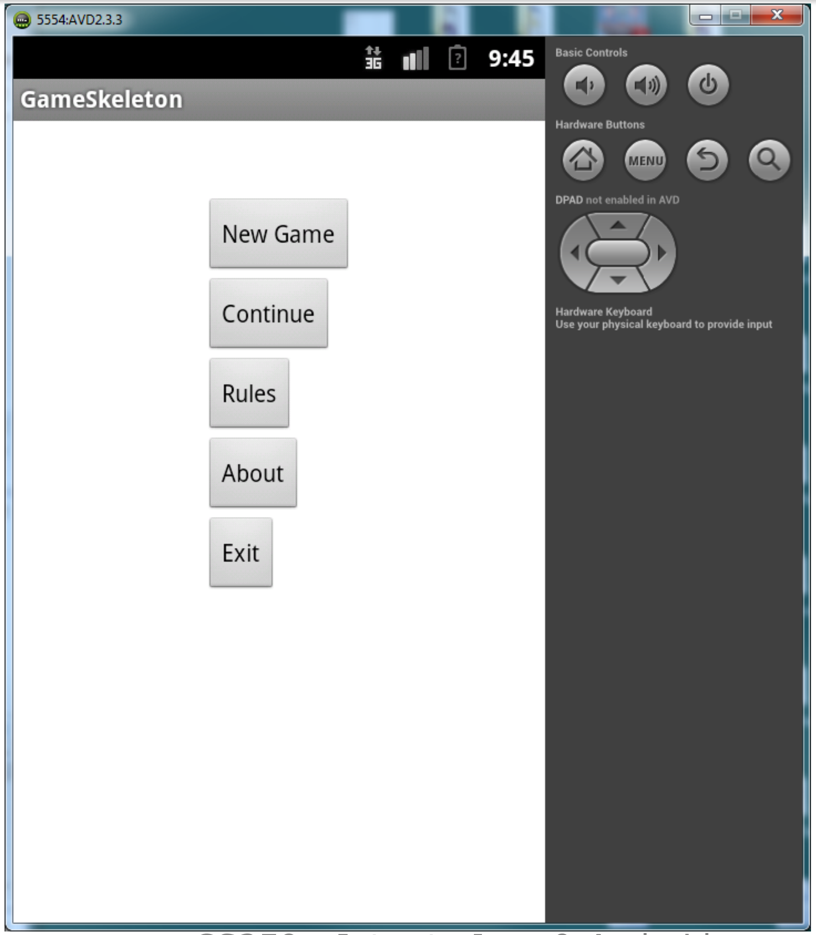
Step #1
Add 5 Buttons



UI Design Specifics

1. Button names are btnNewGame, btnContinue, btnRules, btnAbout, and btnExit
2. String name & values are:
   - sNewGame is New Game
   - sContinue is Continue
   - sRules is Rules
   - sAbout is About
   - sExit is Exit

Step #2
Change Button Text



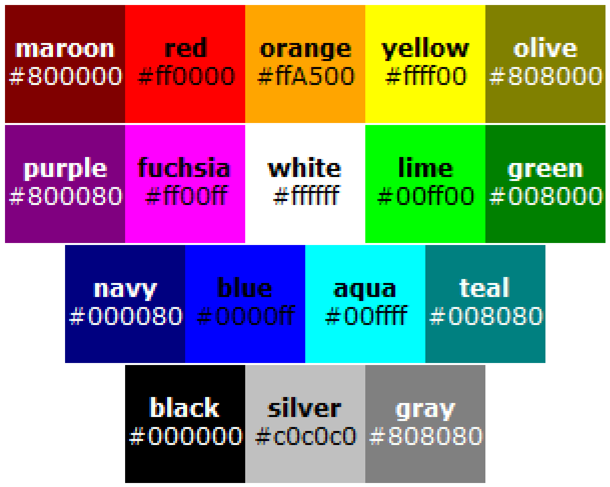
More XML

- What if we want to change the background color?

1. Create an xml color definition resource in the values folder called **colors.xml** as follows:

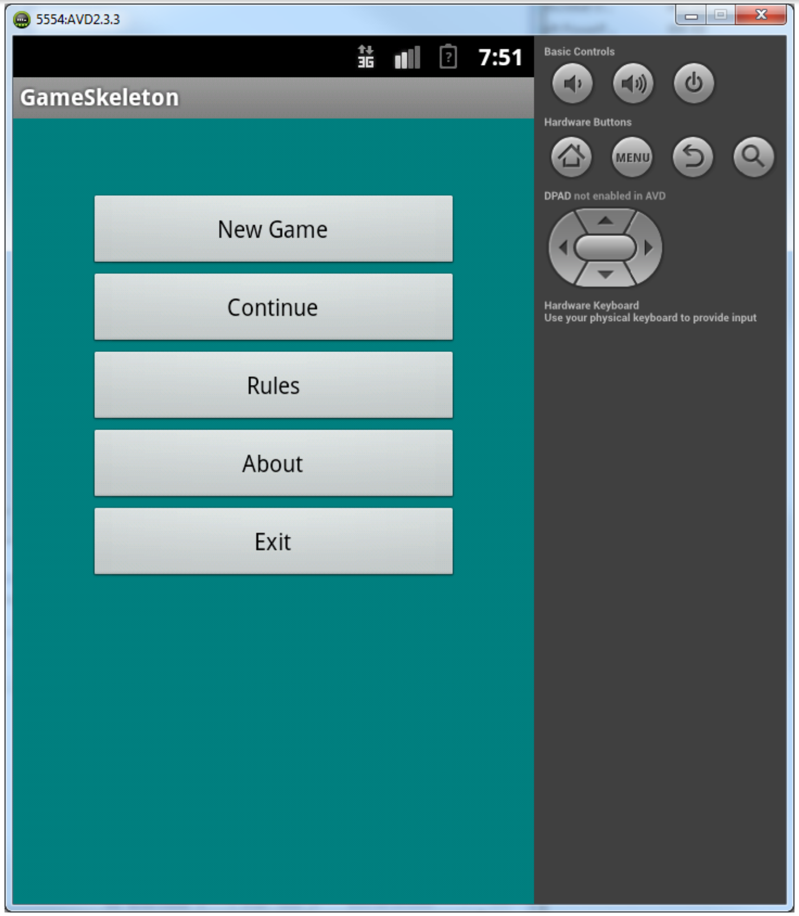
```
<?xml version="1.0" encoding="UTF-8"?>
<resources>
</resources>
```

2. Add the following colors:

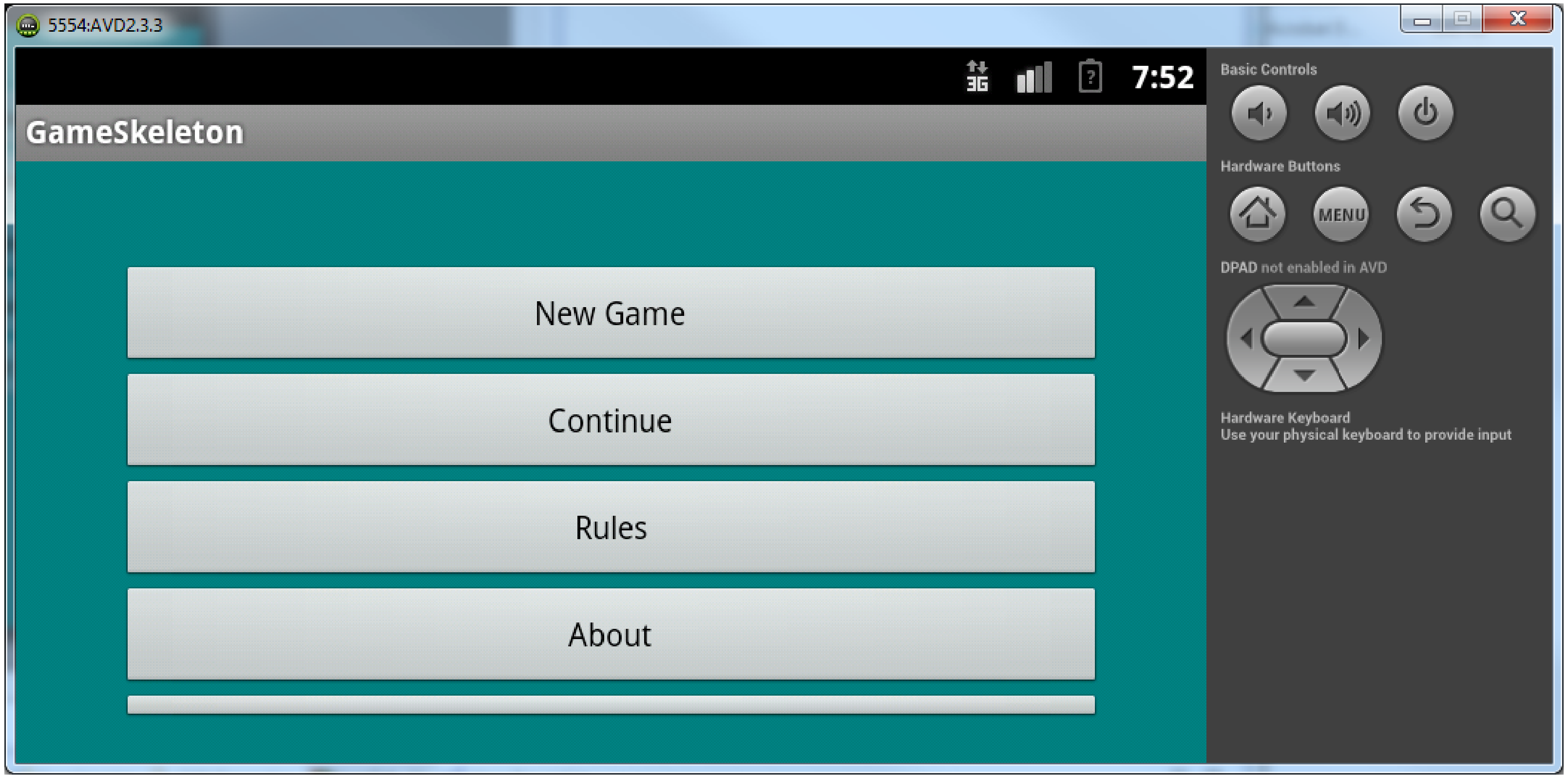


maroon #800000	red #ff0000	orange #ffA500	yellow #ffff00	olive #808000
purple #800080	fuchsia #ff00ff	white #ffffff	lime #00ff00	green #008000
navy #000080	blue #0000ff	aqua #00ffff	teal #008080	
black #000000	silver #c0c0c0	gray #808080		

Step #3
Change the Buttons/Background



Switch to Landscape left-ctl + F11

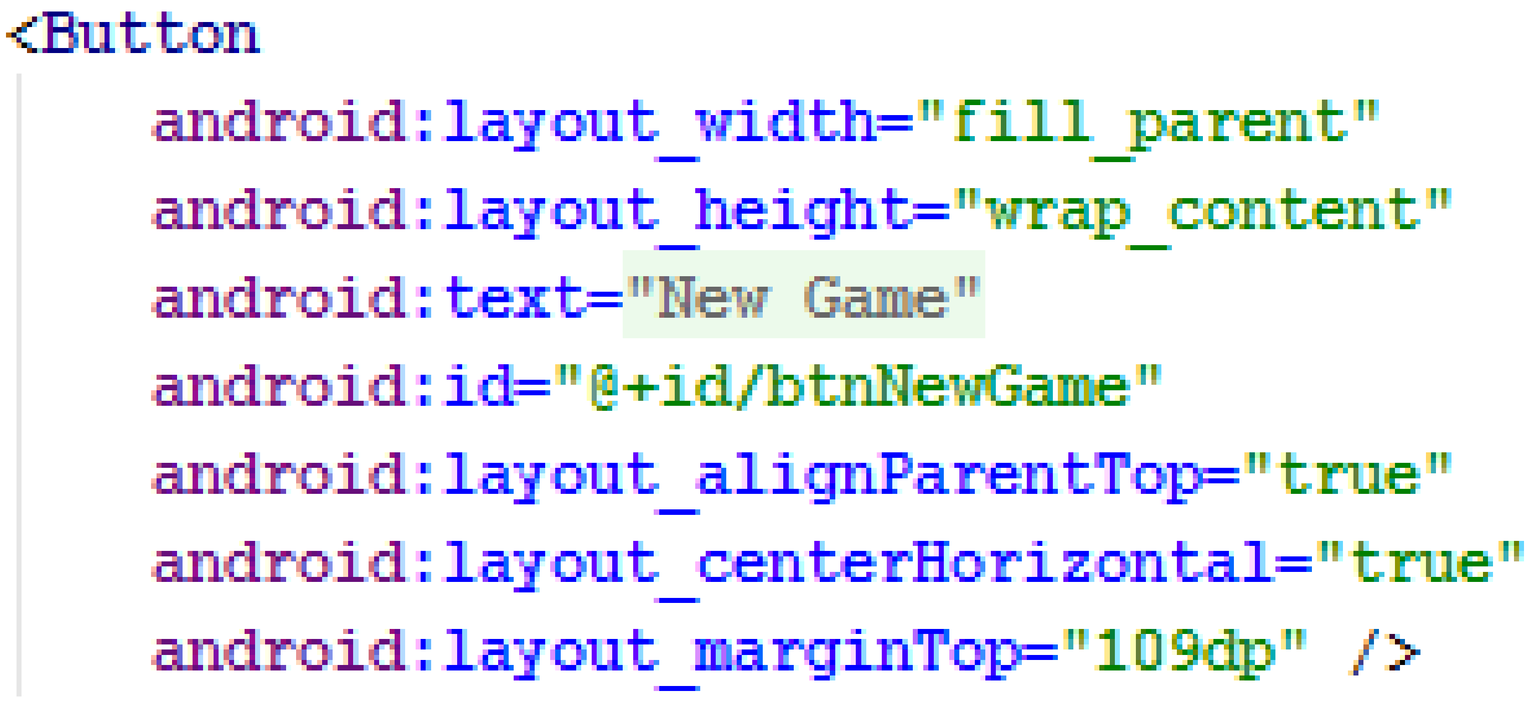


More Attributes

Open activity_game_skeleton.xml and answer the following questions:

1. How many objects exist?
2. How many Views exist?
3. How many ViewGroups exist?
4. What is a Button?
5. How many attributes for the Button btnNewGame are displayed in the xml code?

Button Attributes



```
<Button
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:text="New Game"
    android:id="@+id/btnNewGame"
    android:layout_alignParentTop="true"
    android:layout_centerHorizontal="true"
    android:layout_marginTop="109dp" />
```

Button Attributes

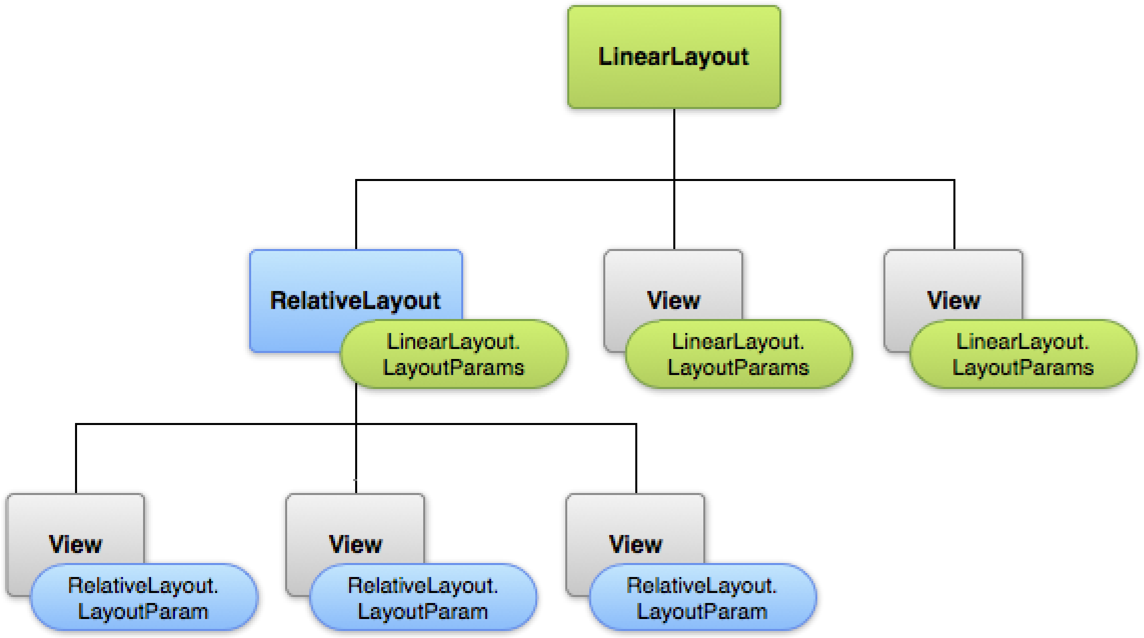
```
android:id="@+id/btnNewGame"
```

@ indicates XML parser should parse & expand the rest of the string and identify it as an ID resource

+ adds resource name to R.java file

More with Layouts

- XML layout attributes named layout_something define layout parameters for each View in a ViewGroup


LinearLayout
RelativeLayout
LinearLayout. LayoutParams
View
LinearLayout. LayoutParams
View
LinearLayout. LayoutParams
View
RelativeLayout. LayoutParam
View
RelativeLayout. LayoutParam
View
RelativeLayout. LayoutParam