COMP08076
Programming Native App Interaction

- Module coordinator: John Nixon
  - john.nixon@uws.ac.uk, room E259, x3617

- Lecture 6
  - Part 2 of module - games
 - Views
  - Motion events

View Class – from lecture 3

➤ This class represents the basic building block for user interface components. A View occupies a rectangular area on the screen and is responsible for drawing and event handling. View is the base class for *widgets*, which are used to create interactive UI components (buttons, text fields, etc.).

➤ http://developer.android.com/reference/android/view/View.html

➤ Extends object

➤ Has subclasses
   - ImageView
   - SurfaceView
   - TextView
   - Etc.

➤ ImageView has subclass ImageButton

➤ TextView has subclasses Button, EditText etc

➤ Button has subclasses CheckBox, RadioButton etc

Where we have used Views before

- TextView
  - A widget
- setContentView(R.layout.activity_main);
  - View from a layout file
- View from a widget (week 1)
  - Button button;
  - button = new Button(this);
  - setContentView(button);

- In interfaces
  - implements View.OnClickListener
  - public void onClick(View view)

ImageView
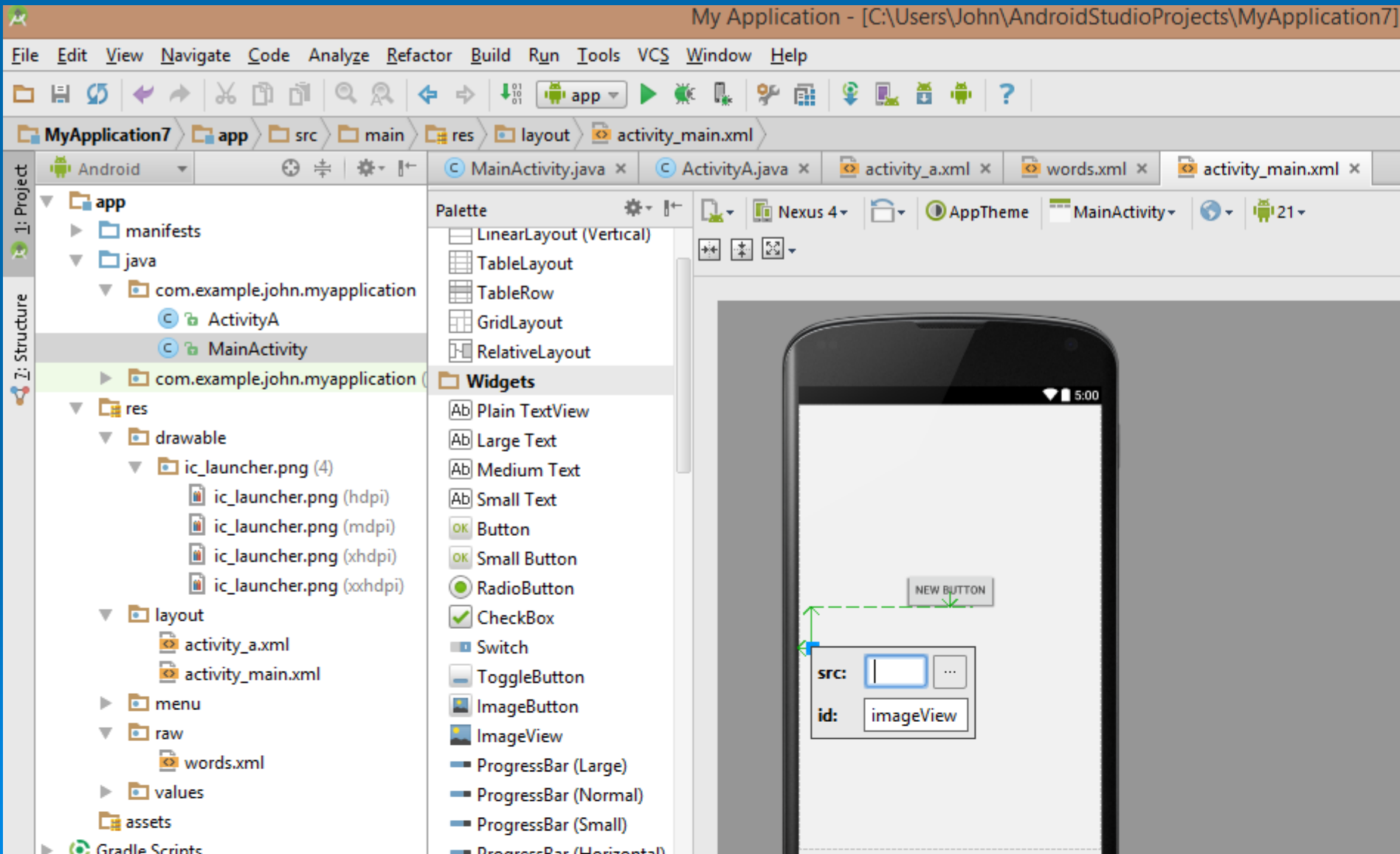
Resources res/

- ➢ Resources
  - res/ and subfolders
  - "Externalise" resources (images, strings) from code
  - Default and alternative (Layout, internationalisation…)
  - Must be placed in the right location. For example bitmap files must live in res/drawable .
  - Android automatically generates an R.java which contains fields whose names correspond to the resources found in res.
 - Are compressed, except for resources in res/raw

res/drawable

➤ A drawable resource is a general concept for a graphic that can be drawn to the screen

➤ We are interested in bitmap images
  • .png (preferred), .jpg (acceptable), .gif (discouraged)

➤ Must be in res/drawable folder

➤ Copy, select drawable folder in project and paste

➤ ImageView

➤ <ImageView
    android:layout_height="wrap_content"
    android:layout_width="wrap_content"
    android:src="@drawable/myimage" />

Constructor Methods

➤ Used to create an instance of a class (i.e. create an object)

➤ Intent intent;

➤ intent = **new** Intent(**this**,ActivityA.**class**);

➤ Button button = new Button(this);

➤ A constructor method has the same name as the name of the class

Creating a custom View
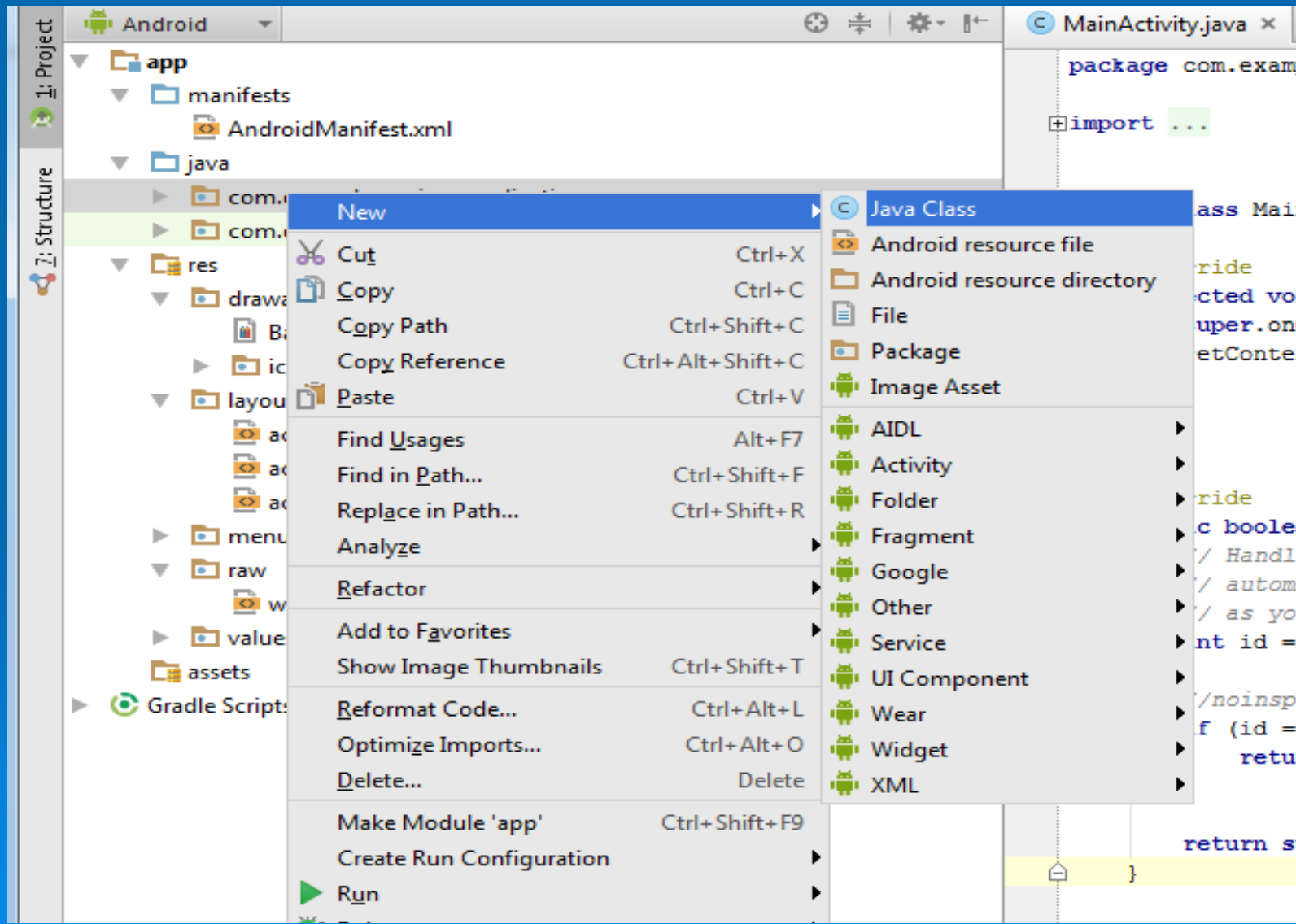
```java
public class GameView extends View {
    // declare variables needed for View
    private Paint redPaint;

    public GameView(Context context) {
        super(context);
        redPaint = new Paint();
        redPaint.setColor(Color.RED);
    }

    @Override
    protected void onDraw(Canvas canvas) {
        canvas.drawCircle(50, 50, 10, redPaint);
    }
}
```

Creating a GameView

➤ Declare

➤ **GameView GV;**

➤ In onCreate() put

➤ **GV = new GameView(this);**   // creates the custom view

➤ **setContentView(GV);**   //sets the content view to the
game view and that is then displayed

How a view works

- Constructor Function
  - "init" method for the view

- Canvas provided to draw on

- onDraw()
  - Called automatically when view constructed
  - Use draw methods
    - canvas.drawCircle(50, 50, 10, redPaint);
    - canvas.drawText("output here", 50, 50, null);
    - plus others
  - Call invalidate() to force further calls to onDraw()

Drawing bitmaps

➢ Declare

➢ **private Bitmap ball1;**

➢ attach the image in drawable in the GameView's constructor (after super(context)

➢ **ball1 = BitmapFactory.decodeResource(getResources(), R.drawable.ball);**

➢ and drawn in onDraw()

➢ **canvas.drawBitmap(ball1, 10, 10, null);**

Graphic

- Graphic 0,0 top left
- ball.getWidth();
- ball.getHeight();

Events/MotionEvents

➢ android.view.MotionEvent class for screen touch data

➢ MotionEvent contains info about active touch points on screen

➢ event.getAction()
  - ACtION_DOWN, ACTION_MOVE, ACTION_UP

➢ event.getX()

```java
public boolean onTouchEvent(MotionEvent event) {
    int eventaction = event.getAction();

    switch (eventaction ) {
        case MotionEvent.ACTION_DOWN:
            output = "down";
            break;
        case MotionEvent.ACTION_MOVE:
            output = "move";
            break;
        case MotionEvent.ACTION_UP:
            output = "up";
            break;
    }
    invalidate();
    return true;
}
```

"hittest"

> if(X > ballX && X < ballX + ball.getWidth() && Y > ballY && Y < ballY + ball.getHeight()) isHit = true;

Random numbers

- ballX = (int)(Math.random()*720);
- ballY = (int)(Math.random()*1280);
-
- where Math.random() gives a double between 0 and 1.