Lab Session 5 – Object Orientation in Android

.Objectives

The objectives for this week are:

- Understand adding a class structure to Android.
- Add classes to the Arkanoid game to allow the reuse of game items.
- Add game objects to the game to be able to use with a game loop and interaction via touch in later labs.

.Tasks

Task 1

- The use of classes is fundamental to Androids structure. During the course of this lab we will construct a number of classes to aid the construction of the Arkanoid game.
- Additionally the use of packages can add structure to a game project. We will organise the Arkanoid project so that the structure incorporates a number of packages to hold varying reusable content.
- Open the Arkanoid game that you have been developing over the past four weeks.
- Your game will contain a number of activities that are contained within the root package. Holding all of the code structure in the root can result in an unmanageable structure.
- It is best practice to structure packages to hold the code of our game. We can use the **Model – View – Controller** approach for Android games to hold the classes and activities for the game structure.
- In your root package (com.xxx.Arakanoid), add three further packages. The hierarchy would be:

    - com.xxx.Arkanoid
        - model
        - view
        - controller
- Move your activities into the controller package. They can be dragged and dropped (automatically refactored) or moved using the refactoring process.

Task 2

- One of Arkanoid's game mechanics is the use of a number of blocks that need to be destroyed. To allow the reuse of a "game block", we will produce a block abstract class that we can inherit from.
- Produce an abstract class called **Block** with a constructor that can be passed:
    - Width
    - Height
    - X Position
    - Y Position
    - Colour
- Android colours: http://developer.android.com/reference/android/graphics/Color.html
- Rectangles in android: http://developer.android.com/reference/android/graphics/Rect.html
- Add in getters and setters for the block.
- To assist you, a structure could be:

```
abstract class Block {
…add variables

    public Block (int widthIn, int heightIn, int xPosIn,
int yPosIn, int blockColor){
```

```
            …Instantiation

        }

        //Method to draw a rectangle on the screen
        public void drawRect(Paint p, Canvas c){
           ….need to set the position, colour and draw the
    rectangle.
           p.setColor(colour);
           c.drawRect(left, top, right, bottom, p);
        }

        //Set the width of the rectangle
        public void setWidth (int widthIn){
            …
        }

        //Set the height of the rectangle
        public void setHeight (int heightIn){
            …
        }

        //Get the width of the rectangle
        public int getWidth() {
            …
        }

        //Get the height of the rectangle
        public int getHeight (){
            …
        }
    }
```

- From the Block, produce a **BreakBlock** class that inherits the methods.
- These will be added to the **Model** Package.

Task 3

- To be able to display the block that has been created, we can use a **view.** This takes the place of the xml files that we have been using up until this point to display content.
- Add in the GameView class that is on Blackboard. Your lab tutor can discuss with you regarding the use of views.

- Review the code in the GameView. We will adapt this code to add a block and display it.
- To use the GameView, we need to adapt the GameActivity that we been previously using. This will begin to allow us to produce game content. Use the code below as a guide to change the MainActivity so that it will display a View.
- Here we are creating a new GameView and passing it to the activity to display.s

```
//The games gameview
private GameView gv;

@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
        //Pass the GameView to setContentView
    //setContentView(R.layout.activity_main);
     gv = new GameView(this);
    setContentView(gv);
}
```

- **<u>If you are unsure of this process, ask you lab tutor.</u>**

Task 4

- Add an interface to produce a BonusBlock. This will set and get a Bonus Boolean

Task 5

- Place a number of blocks across the screen that have differing colours.
- Use an **ArrayList** for this: http://developer.android.com/reference/java/util/ArrayList.html