UWS UNIVERSITY OF THE WEST of SCOTLAND

13954 - COMP 11062: Mobile Networks And Smartphone Applications

Lab 5: Puzzle App

UNIVERSITY OF THE WEST of SCOTLAND

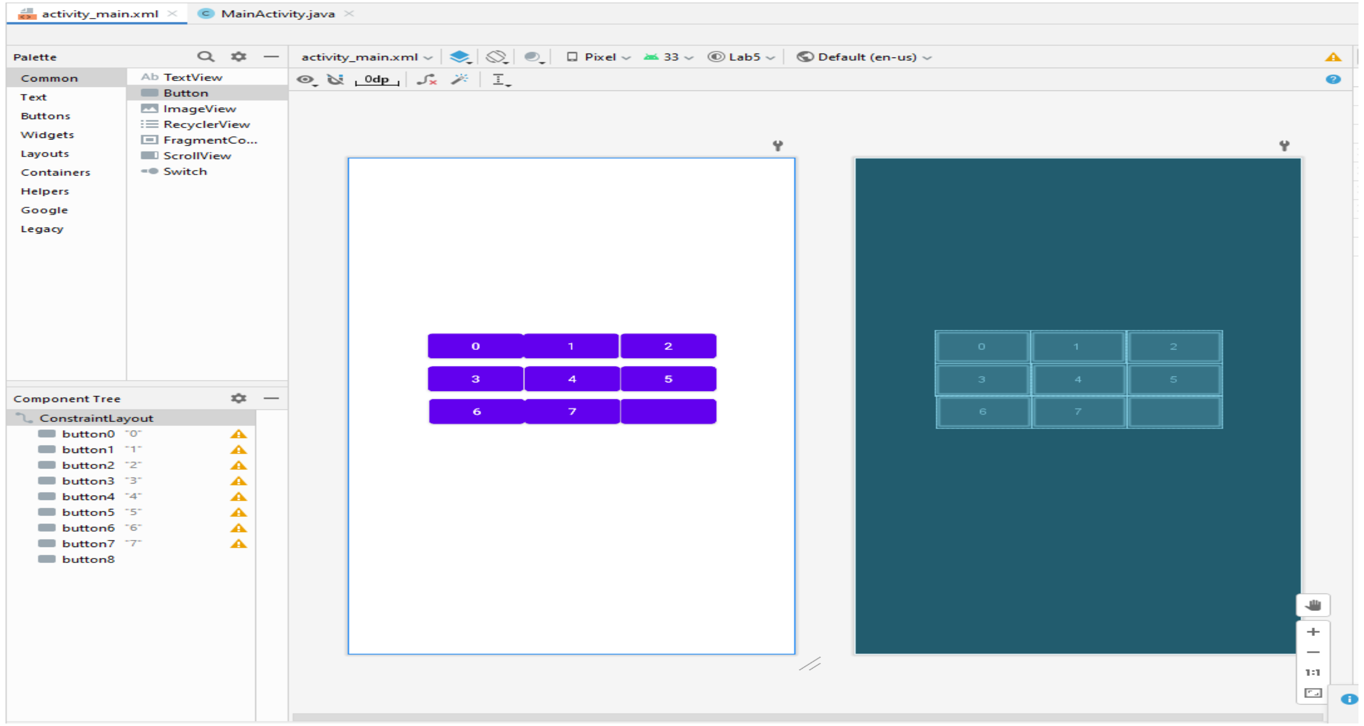
Learning Objectives

- Create a square solver with 3x3 Grids using buttons.
- Create a square solver with 3x3 Grids using Image Buttons.

Task 1

- Create the UI (User Interface) for a 3x3 grid of buttons
- Note: For every lab it is recommended to start a new Android Studio Project. Refer to Lab1 & Lab2 for instructions on how to start a new project.

Step 1: In the Design view of your activity_main.xml file, drag and drop nine buttons forming a 3x3 grid like the picture below:



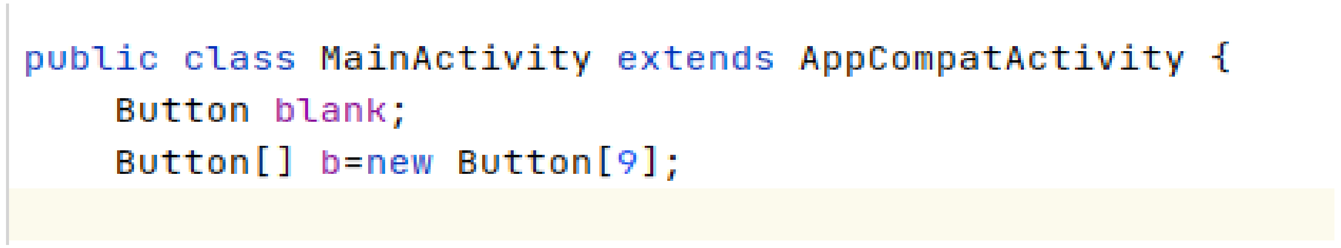
Step 2: Change the appropriate attribute of each button so that their ids range from button0 to button8, and their texts from 0 to 7. Leave the text of the last button blank. After ids change, if the GUI asks for confirmation to propagate/refactor the changes, say yes.

Task 2

- Write java code, for the User Interface composed of the nine buttons, to turn them into a sliding puzzle. You should only be able to press a button if it is on the left, right, above or below the blank button. When a numbered button is pressed, it should swap positions with the blank button.

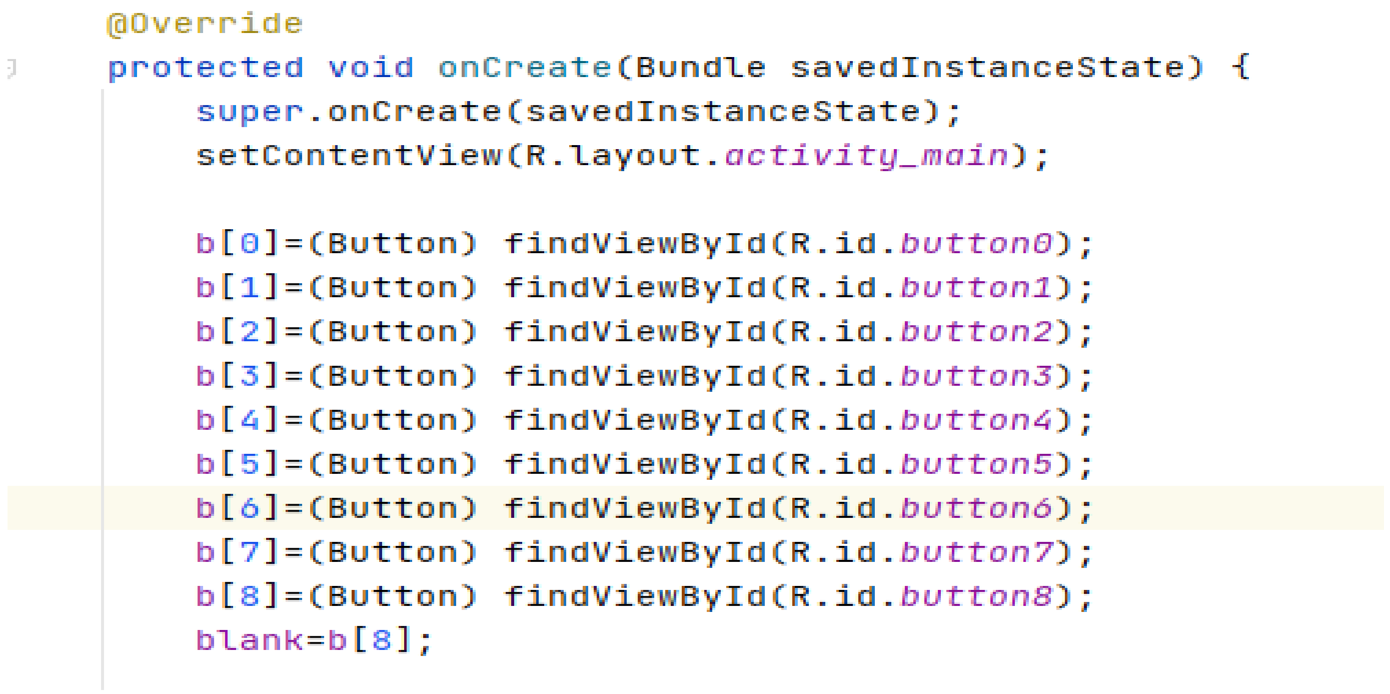
UWS UNIVERSITY OF THE WEST of SCOTLAND

Step 1: Create an object named blank of type Button to keep track of which button is currently the blank one. Create an array of buttons named b, of size 9 and type Button.



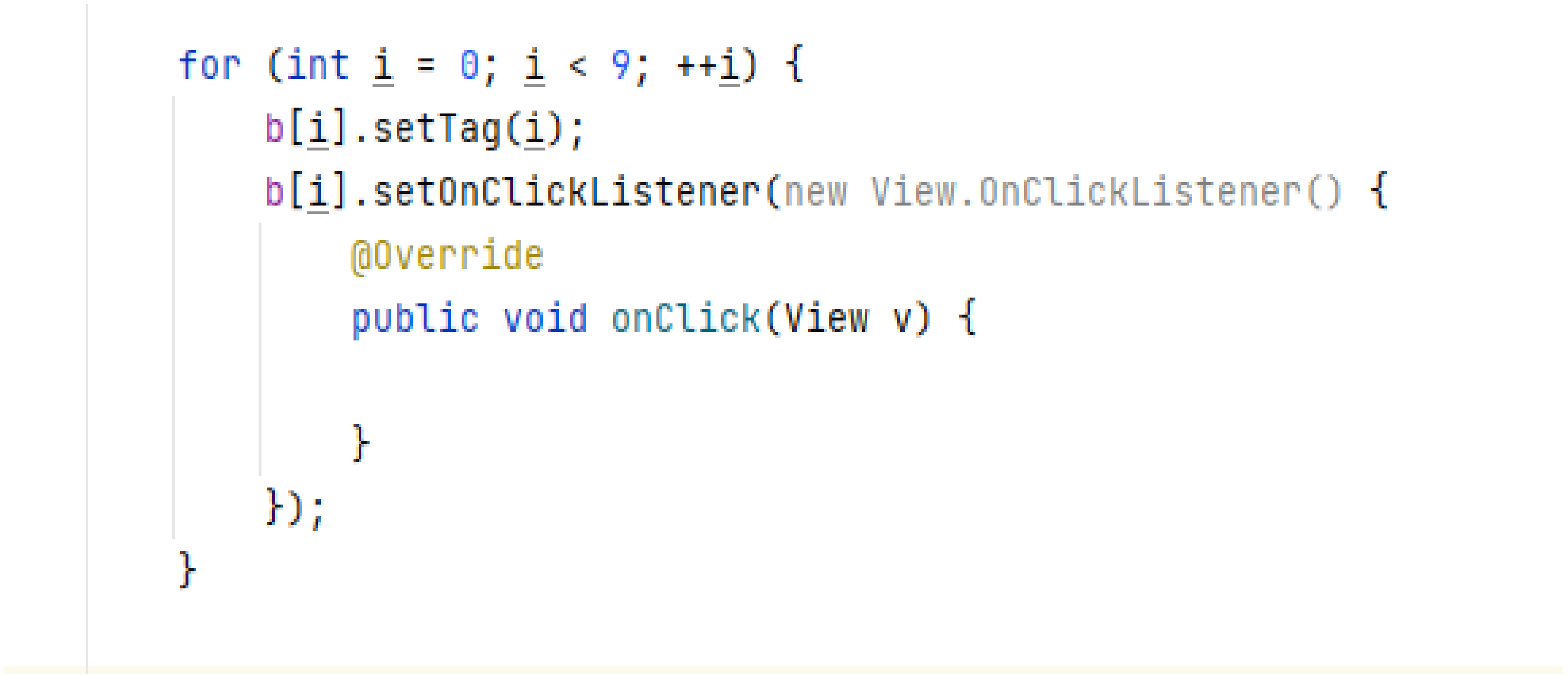
```
public class MainActivity extends AppCompatActivity {
    Button blank;
    Button[] b=new Button[9];
```

Step 2: Map the ID of the buttons we created in activity_main.xml to the elements of our array b. Initialize the blank object with b[8], which represents the current blank button.



```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);

    b[0]=(Button) findViewById(R.id.button0);
    b[1]=(Button) findViewById(R.id.button1);
    b[2]=(Button) findViewById(R.id.button2);
    b[3]=(Button) findViewById(R.id.button3);
    b[4]=(Button) findViewById(R.id.button4);
    b[5]=(Button) findViewById(R.id.button5);
    b[6]=(Button) findViewById(R.id.button6);
    b[7]=(Button) findViewById(R.id.button7);
    b[8]=(Button) findViewById(R.id.button8);
    blank=b[8];
```

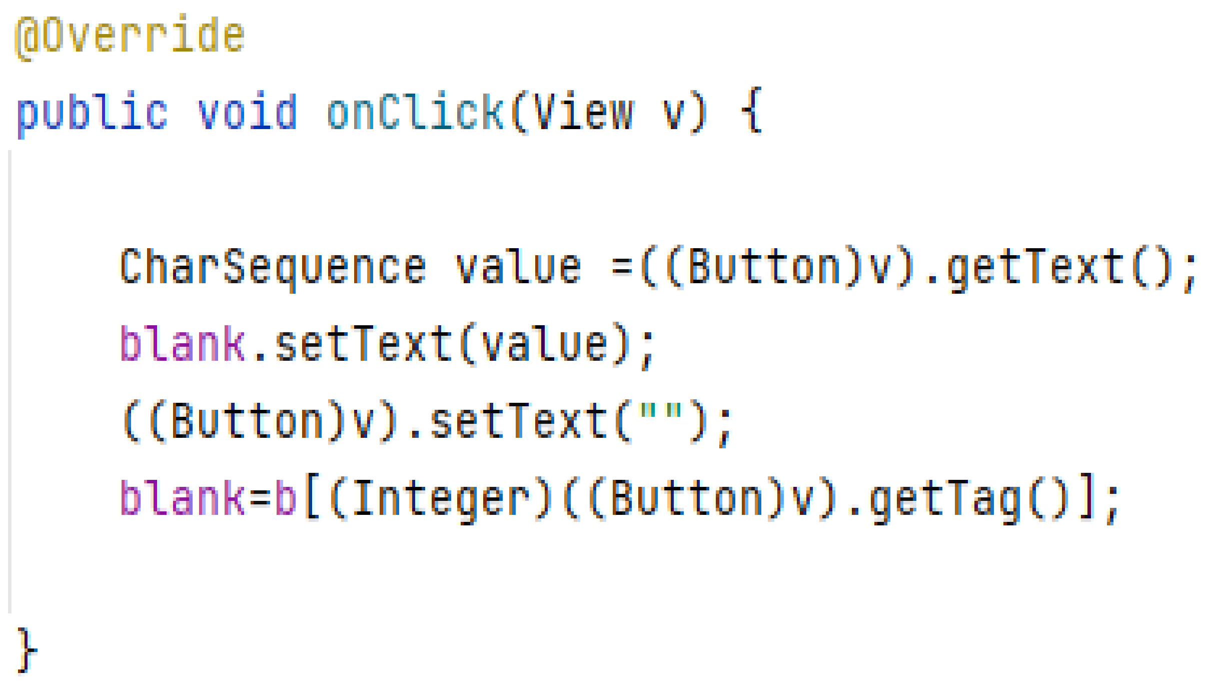
Step 3: Create a loop with 9 iterations (one for each button). Add a tag to each button, to track their position in the array regardless of their current text. Add an onClickListener to each Button so that it executes code when clicked.



```
for (int i = 0; i < 9; ++i) {
    b[i].setTag(i);
    b[i].setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {

        }
    });
}
```

Step 4: Inside the onClick method add code to switch the button’s current text with that of the blank button.

Use getText to retrieve the number (0, 1, 2,…,7) from the button that was pressed, and setText to apply it to the blank button. Set the text of the pressed button to blank (“”) and update the variable that keeps track of which button is the current blank button.

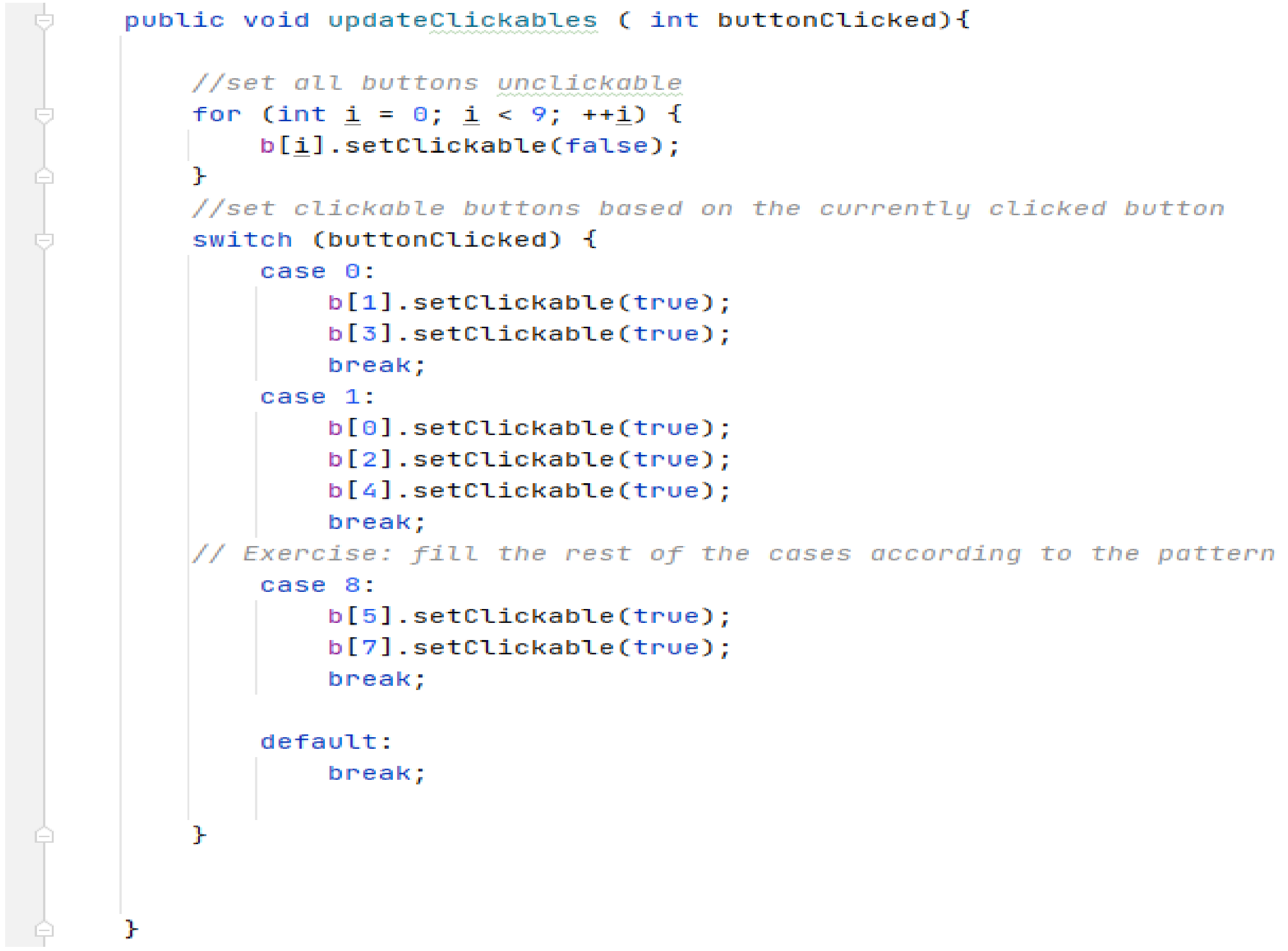


```
@Override
public void onClick(View v) {

    CharSequence value =((Button)v).getText();
    blank.setText(value);
    ((Button)v).setText("");
    blank=b[(Integer)((Button)v).getTag()];

}
```

Step 5: Add a method named updateClickables inside the MainActivity class to enable and disable “clickability” based on the last clicked button.

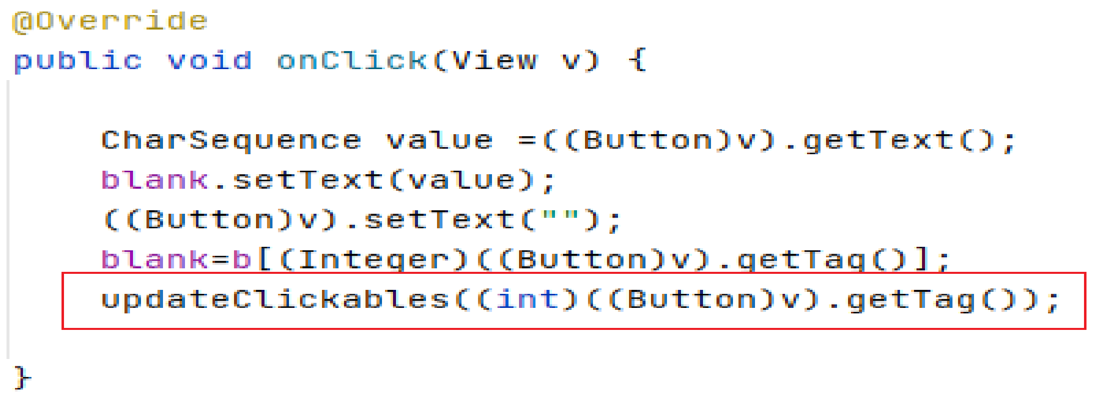
Create a for loop to disable all the buttons.

Use a case based, switch structure to handle the logic of which buttons should be enabled.



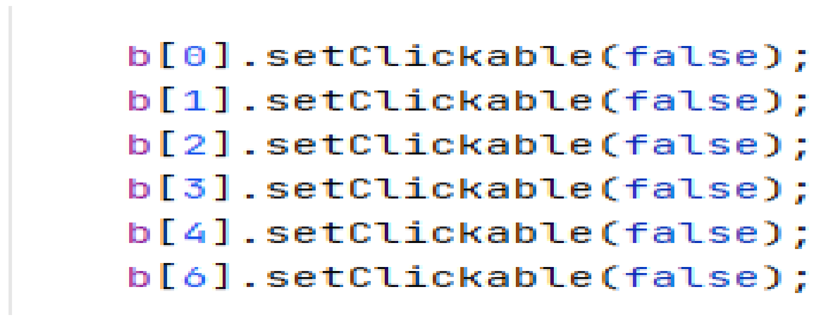
```
public void updateClickables ( int buttonClicked){

    //set all buttons unclickable
    for (int i = 0; i < 9; ++i) {
        b[i].setClickable(false);
    }
    //set clickable buttons based on the currently clicked button
    switch (buttonClicked) {
        case 0:
            b[1].setClickable(true);
            b[3].setClickable(true);
            break;
        case 1:
            b[0].setClickable(true);
            b[2].setClickable(true);
            b[4].setClickable(true);
            break;
    // Exercise: fill the rest of the cases according to the pattern
        case 8:
            b[5].setClickable(true);
            b[7].setClickable(true);
            break;

        default:
            break;

    }


}
```

Step 6: Go back to the onClick method and call the updateClickables method.



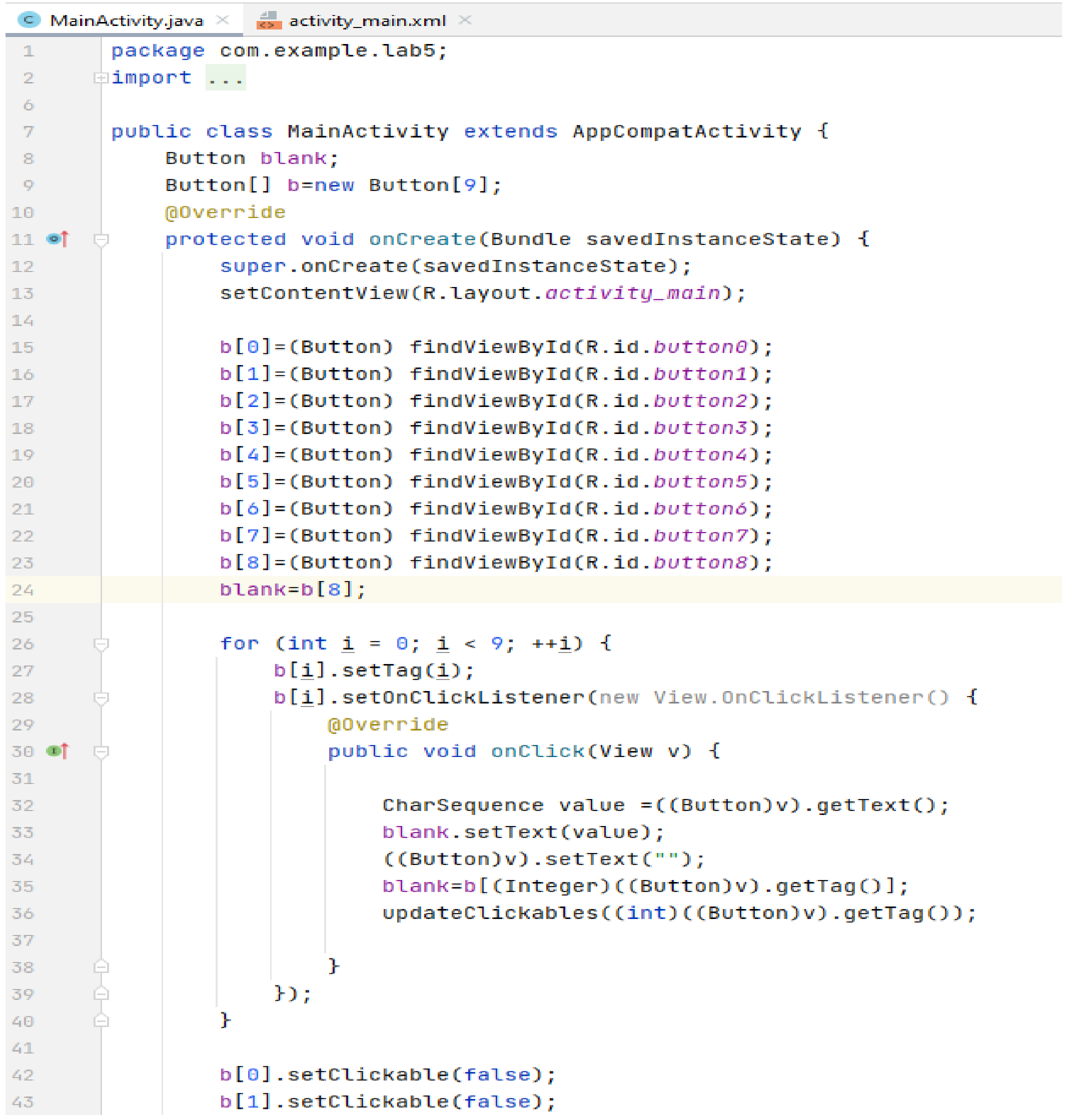
```
@Override
public void onClick(View v) {

    CharSequence value =((Button)v).getText();
    blank.setText(value);
    ((Button)v).setText("");
    blank=b[(Integer)((Button)v).getTag()];
    updateClickables((int)((Button)v).getTag());

}
```

In order to make only buttons 5 and 7 clickable when you first run the App, disable b[0] to b[6] after you set the click listeners inside the for loop.

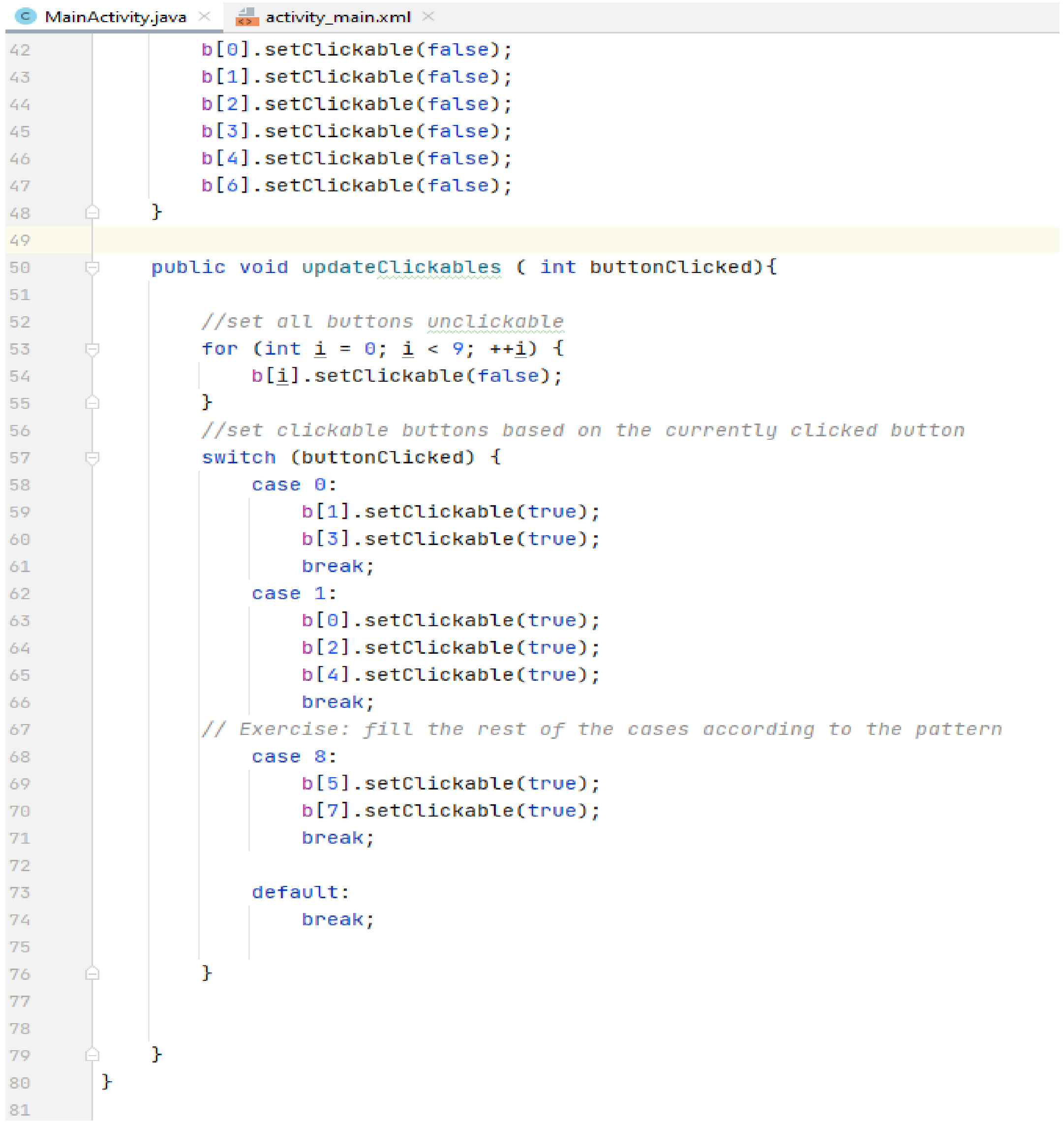


```
        b[0].setClickable(false);
        b[1].setClickable(false);
        b[2].setClickable(false);
        b[3].setClickable(false);
        b[4].setClickable(false);
        b[6].setClickable(false);
```

In the end your MainActivity.java file should look like the screenshots below:



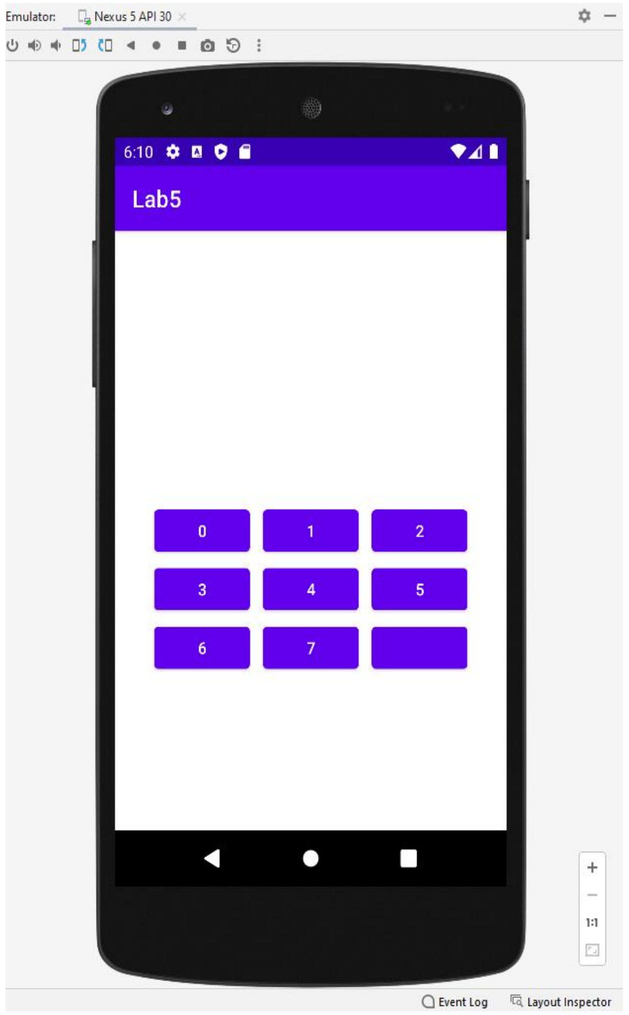
```java
package com.example.lab5;
import ...

public class MainActivity extends AppCompatActivity {
    Button blank;
    Button[] b=new Button[9];
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        b[0]=(Button) findViewById(R.id.button0);
        b[1]=(Button) findViewById(R.id.button1);
        b[2]=(Button) findViewById(R.id.button2);
        b[3]=(Button) findViewById(R.id.button3);
        b[4]=(Button) findViewById(R.id.button4);
        b[5]=(Button) findViewById(R.id.button5);
        b[6]=(Button) findViewById(R.id.button6);
        b[7]=(Button) findViewById(R.id.button7);
        b[8]=(Button) findViewById(R.id.button8);
        blank=b[8];

        for (int i = 0; i < 9; ++i) {
            b[i].setTag(i);
            b[i].setOnClickListener(new View.OnClickListener() {
                @Override
                public void onClick(View v) {

                    CharSequence value =((Button)v).getText();
                    blank.setText(value);
                    ((Button)v).setText("");
                    blank=b[(Integer)((Button)v).getTag()];
                    updateClickables((int)((Button)v).getTag());

                }
            });
        }

        b[0].setClickable(false);
        b[1].setClickable(false);
```

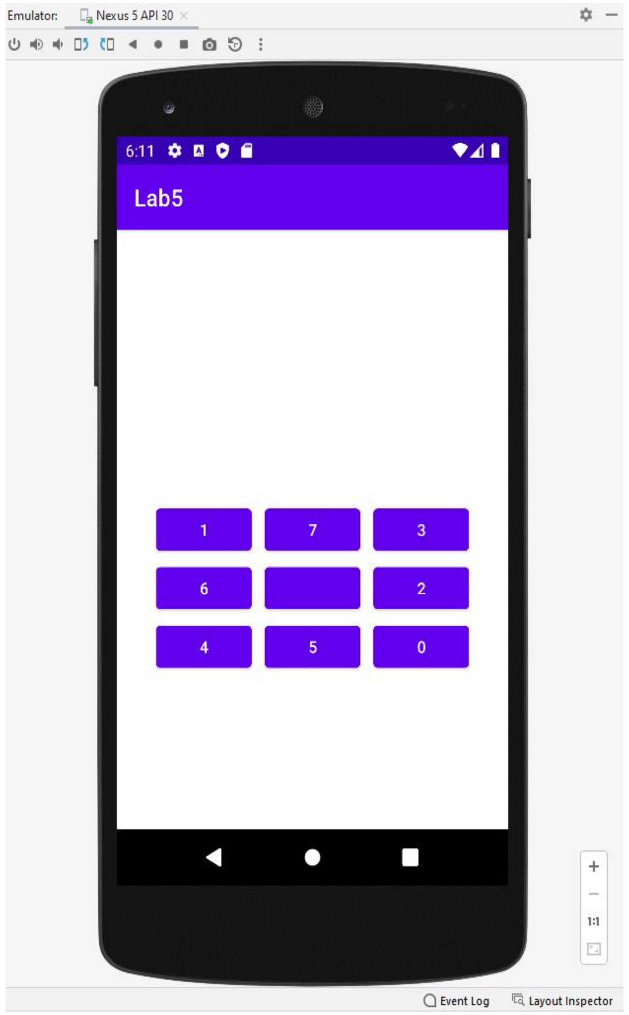


```java
        b[0].setClickable(false);
        b[1].setClickable(false);
        b[2].setClickable(false);
        b[3].setClickable(false);
        b[4].setClickable(false);
        b[6].setClickable(false);
    }

    public void updateClickables ( int buttonClicked){

        //set all buttons unclickable
        for (int i = 0; i < 9; ++i) {
            b[i].setClickable(false);
        }
        //set clickable buttons based on the currently clicked button
        switch (buttonClicked) {
            case 0:
                b[1].setClickable(true);
                b[3].setClickable(true);
                break;
            case 1:
                b[0].setClickable(true);
                b[2].setClickable(true);
                b[4].setClickable(true);
                break;
        // Exercise: fill the rest of the cases according to the pattern
            case 8:
                b[5].setClickable(true);
                b[7].setClickable(true);
                break;

            default:
                break;
        }


    }
}
```

UWS UNIVERSITY OF THE WEST of SCOTLAND

Task 2 Emulator Result

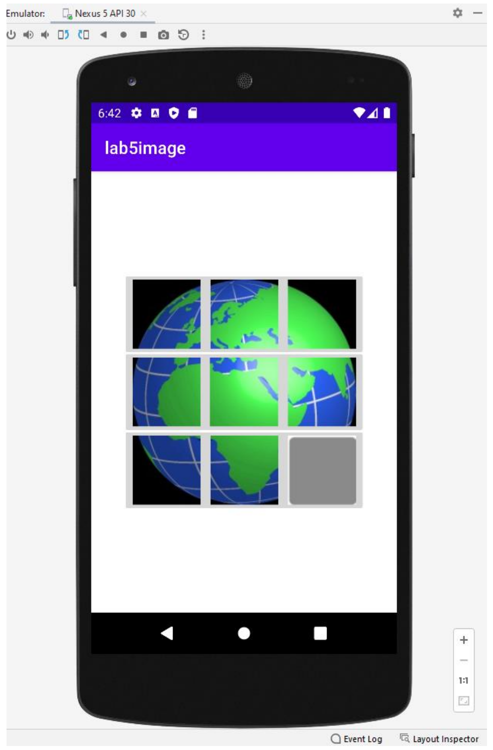


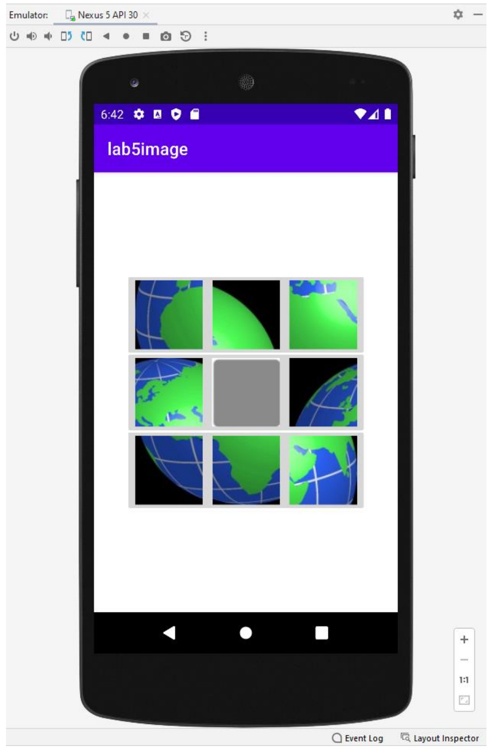


Exercise: Replace the text with pictures

- Download the 9 puzzle pieces from this lab lesson.(Alternatively create your own with an online tool such as http://imagesplitter.net/)
- Follow the instructions from earlier labs about how to import pictures to your project.
- Use ImageButton instead of Button.
- Find out what changes you have to make to the code to make it compile. (hint: the class Drawable might be useful and creating a new project might produce the result quicker)

Emulator Result





Submission: Submit screen shots of your emulator showing your Banner ID and output of exercise. You can create word document and convert to PDF as you must submit a single PDF for all the labs.