Android development course – Exercise 1

Exercise goals:

Practice creation basic android application that use basic forms capabilities:

- TextEdit
- Button
- ImageView
- Toast

Instructions:

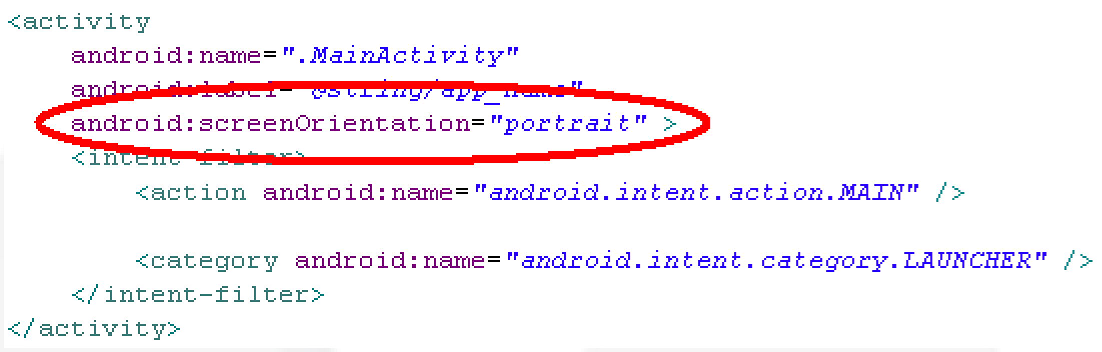
- Each question descries an Android app to create
- Each question descries the required app behavior
- Each question contains input and required output examples
- Solution examples will be provided next week
- As the questions don't depend on each other, if you encounter any problems solving a question it is recommended to move on to the next one.

<u>Tip:</u> Most of the tools and techniques you need in order to solve these questions we've covered in the class session. Use the code examples seen and reuse and alter them as needed. And as always – a quick Google search might help resolve any issue!

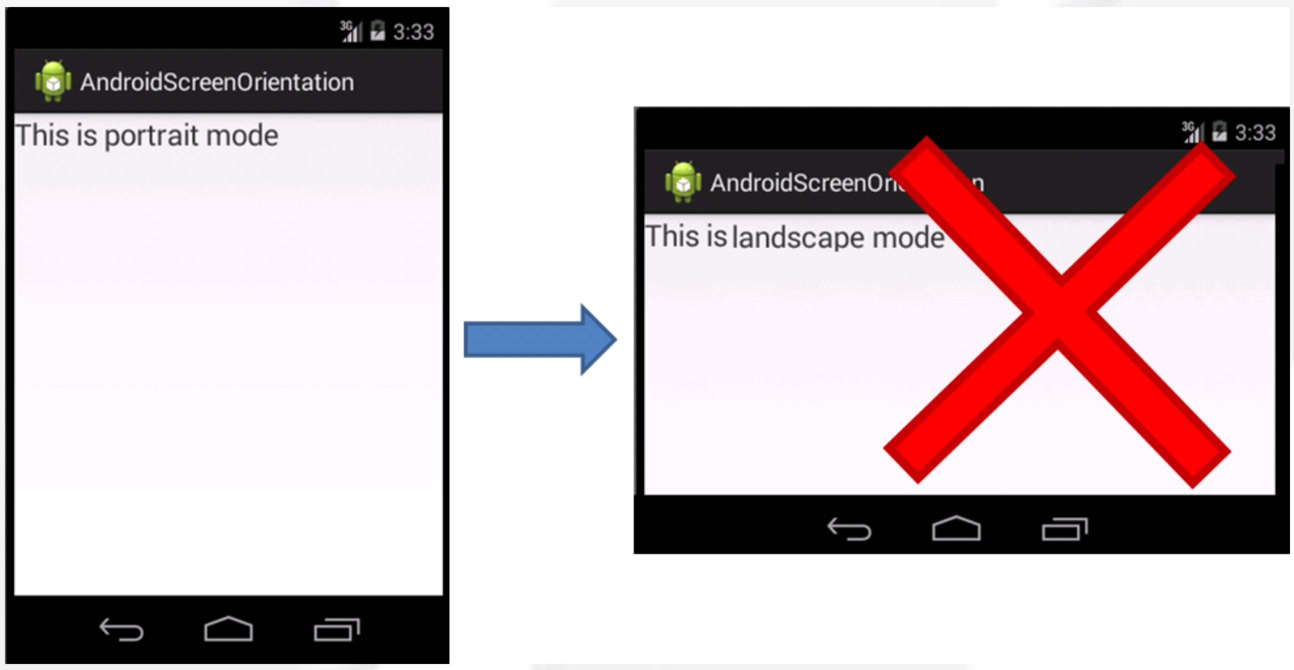
Another Tip: In order to avoid cases where your phone keeps rotating from vertical to horizontal mode (Which leads to application restart), you can add the following line to the AndroidManifest.xml file –

android:screenOrientation="portrait"

Example:



```
<activity
    android:name=".MainActivity"
    android:label="@string/app_name"
    android:screenOrientation="portrait" >
    <intent-filter>
        <action android:name="android.intent.action.MAIN" />

        <category android:name="android.intent.category.LAUNCHER" />
    </intent-filter>
</activity>
```



Taken from Google's formal documentation:

Caution: Your activity will be destroyed and recreated each time the user rotates the screen. When the screen changes orientation, the system destroys and recreates the foreground activity because the screen configuration has changed and your activity might need to load alternative resources (such as the layout).

Taken from: http://developer.android.com/training/basics/activity-lifecycle/recreating.html

Question 1 –

Write an application that shows the user a form where he is requested to fill in his age.

The form will include:

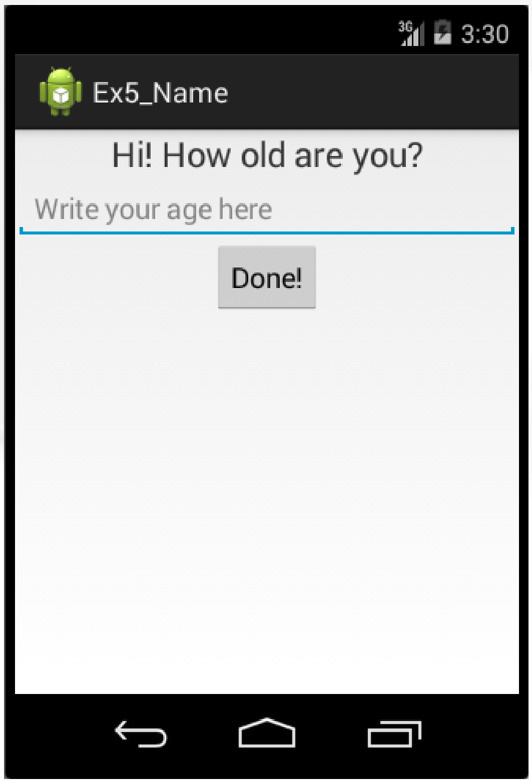
- TextView – asking for the user's age
- EditText – Where the user will fill in his age (Use the 'Hint' property to help the user understand what is expected of him, as seen in the following example)
- Button – clicking on it will show the user a Toast message with a different message according to the provided value:
    - If no value provided
    - A non integer numeric value was provided
    - A value smaller than 0 was provided
    - A value smaller than 18 was provided
    - A value smaller than 120 was provided
    - A value larger than 120 was provided

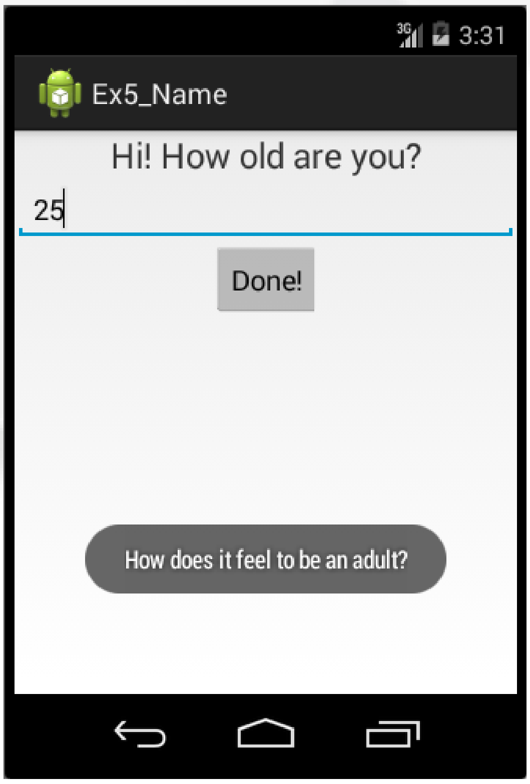
Tip: You can use the Horizontal Linear Layout to neatly organize the various controls in your form.

In the example below we defined it like this:

```
android:gravity="center_horizontal"
android:orientation="vertical"
```

Example:





Question 2 –

Write a calculator application that includes:

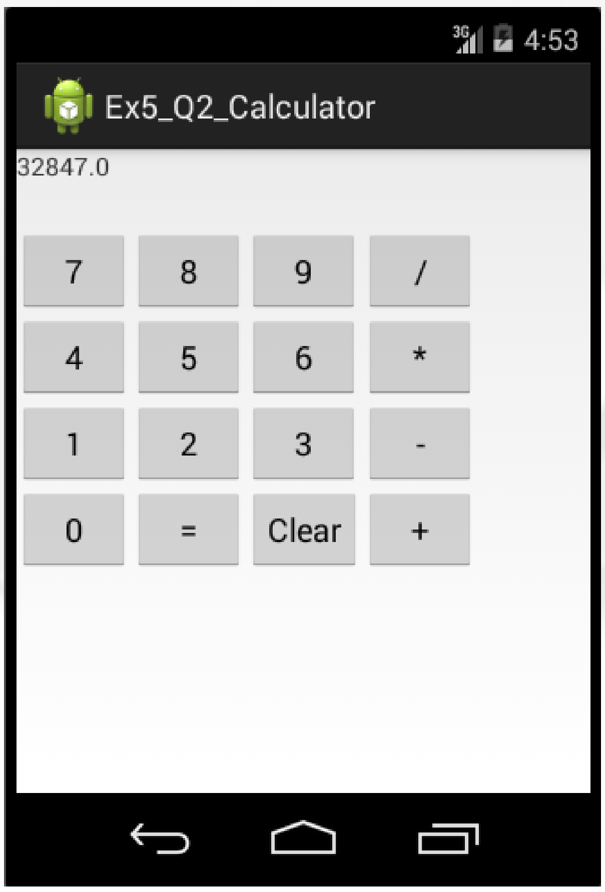
- Buttons for digits
- Buttons for the four basic math operations (sum, subtract, multiply, divide)
- Button for the equals sign =
- A clear button
- A result display field

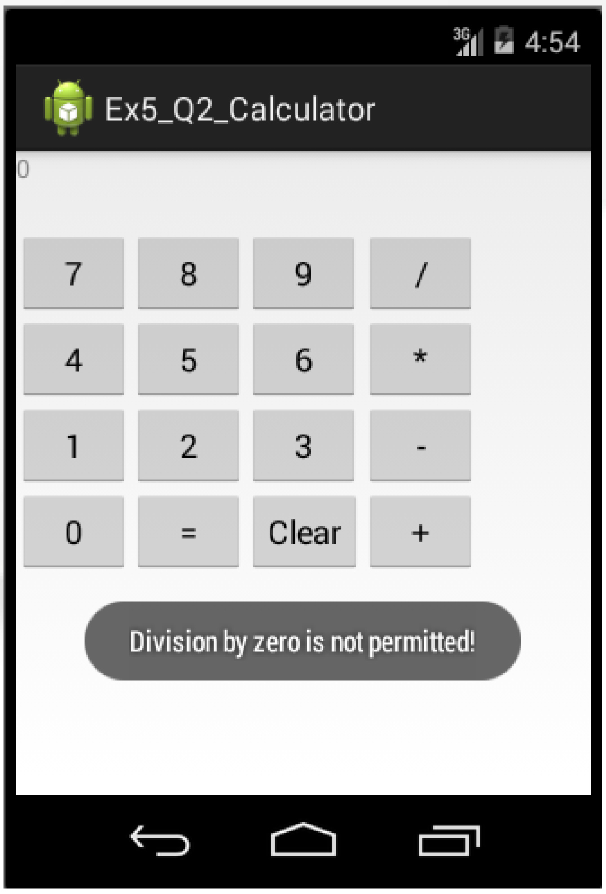
Try playing around a bit with the OS built-in calculator app to get a better understanding of how such an application is expected to behave. (For instance what happens when user clicks a digit and them double clicks the sum button)

Make sure you calculator handles exceptions correctly (such as division by zero)

Tip: In order to concatenate an additional digit in a TextView field, you can use the **append** function instead of **setText**.

The following example uses the GridLayout to keep the UI neat:





Division by zero is not permitted!

Question 3 –

Write an application that simulates a registration form in which user details are required.
It needs to include various fields, when every such field will have a TextView that describes it and a control to input the value.

In the bottom of the form place a button that upon clicking it will perform a validation of provided input for each field, and if any error occur in the validation, a proper message will be shown to the user. If all fields have been filled in correctly show a confirmation message.

The required fields are:

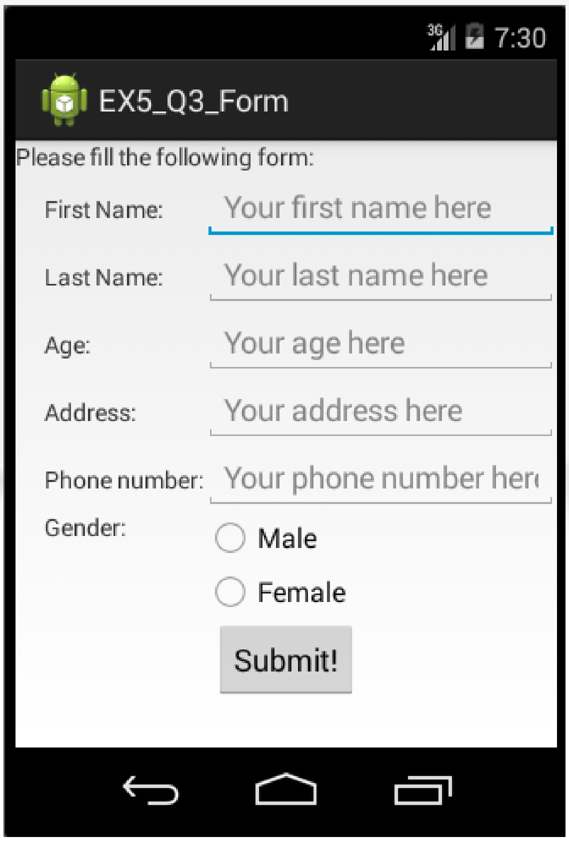
- First name (Minimal length of 3)
- Lat name (Minimal length of 3)
- Age (Value of 18 to 99. Numeric value only)
- Address (Minimal length of 5)
- Phone number (Must be numeric only and length of at least 8 digits)
- Gender (Use a Radio control to allow choosing only one value)

Tips:

- In numeric fields you can give the EditText the following property:

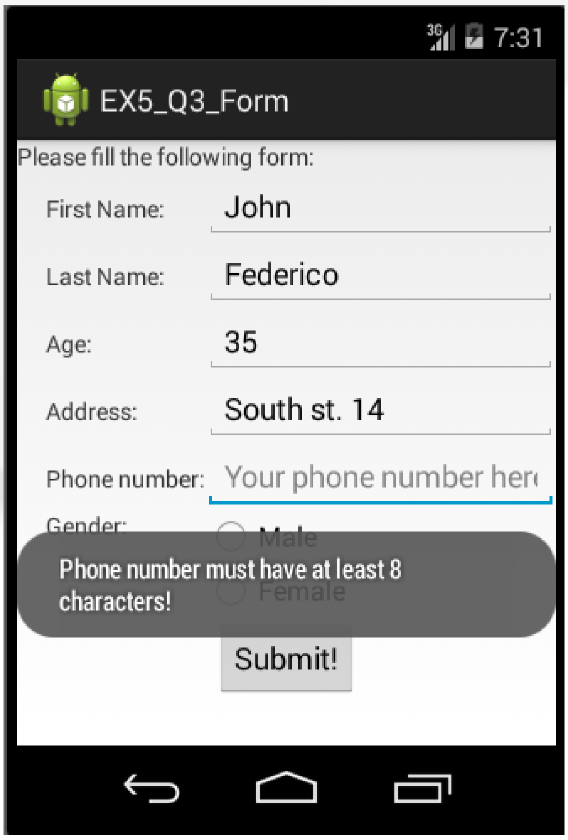
```
android:inputType="number"
```

- To Make sure only one value is allowed in a Radio control, make sure all the Radio controls are grouped under one RadioGroup control.


EX5_Q3_Form

Please fill the following form:

First Name:	Your first name here
Last Name:	Your last name here
Age:	Your age here
Address:	Your address here
Phone number:	Your phone number here
Gender:	Male / Female


EX5_Q3_Form

Please fill the following form:

First Name:	John
Last Name:	Federico
Age:	35
Address:	South st. 14
Phone number:	Your phone number here
Gender:	Male / Female

Phone number must have at least 8 characters!

Question 4 –

Create an app that contains four buttons for directions – up, down, left, right.

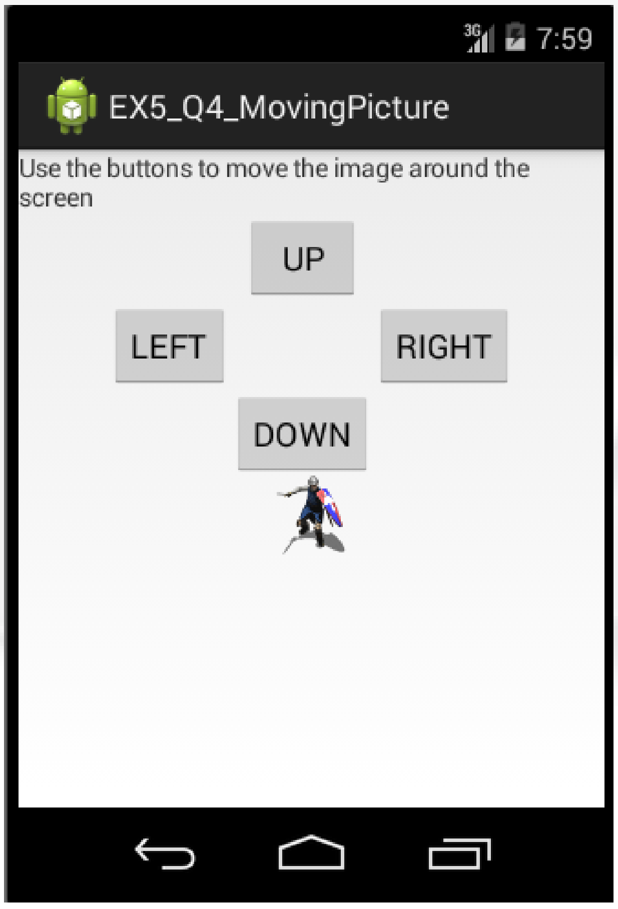
In addition place an image on the screen by using the ImageView control.

Each click of a button will move the image a few pixels in the desired direction by manipulating the x and y coordinates of the image.

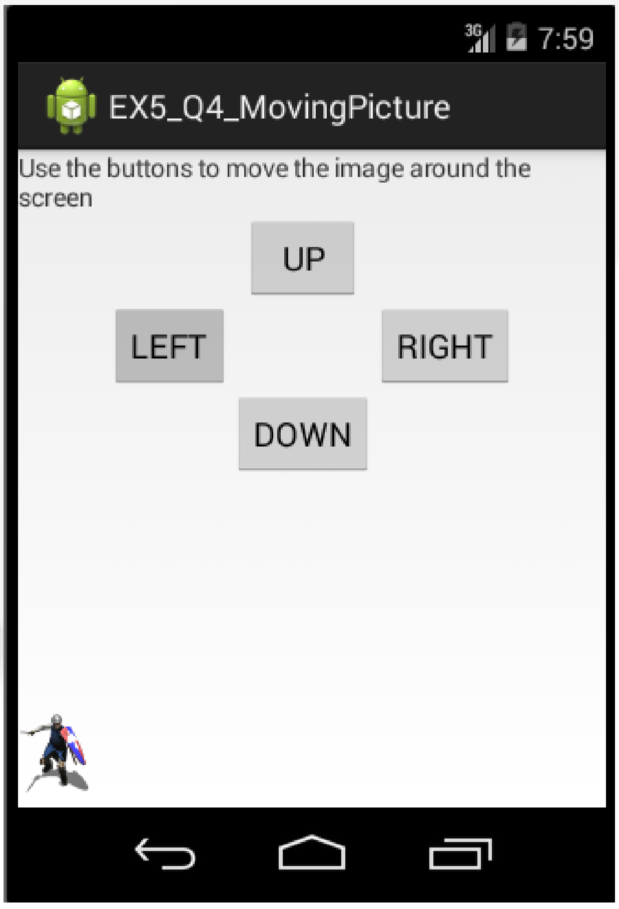
Make sure to import the image in various resolutions as seen in class.

Tip: Free game graphics can be found in: http://opengameart.org

Example:



Use the buttons to move the image around the screen



Use the buttons to move the image around the screen

Question 5 –

Write an application that displays an image of a ball that bounces around the screen. The ball moves automatically – one step every 10 milliseconds in some direction (Use a Timer).

When the ball hits a wall (one of the screen edges) it changes its direction – for instance, when hitting the right wall, the direction will be changed to the left.

Upon each collision, the ball will randomize the new movement angle – pixels to move in the direction can be in the range of 1-3, meaning that at every moment the ball moves in the x and the y axis between -3 to 3 pixels at a time. (Omitting 0 of course).

Comment1: In order to identify the wall hitting one of the walls, we need to check if the x or y coordinates become smaller than zero (Meaning hitting the left or top wall) or larger than the screen width or height (meaning hitting the bottom or right wall). We can obtain screen size with this code:

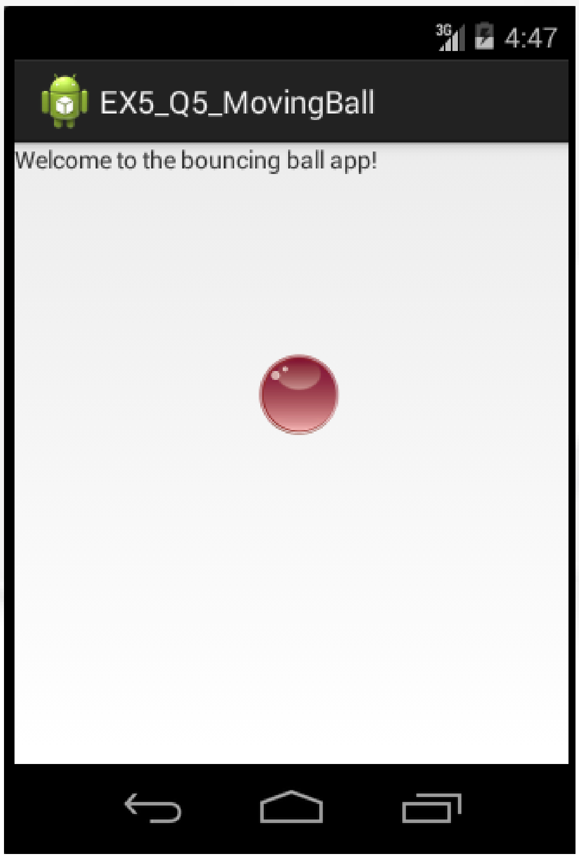
```
DisplayMetrics metrics = new DisplayMetrics();

getWindowManager().getDefaultDisplay().getMetrics(metrics);

int screenHeight = metrics.heightPixels;

int screenWidth = metrics.widthPixels;
```

Comment2: As the Timer is running in a different thread than the UI, it is forbidden for it to perform the UI changes himself. There are several ways to handle this, one of them is making the Timer thread signal the main UI thread it's time to make a UI change. This can be done like this:

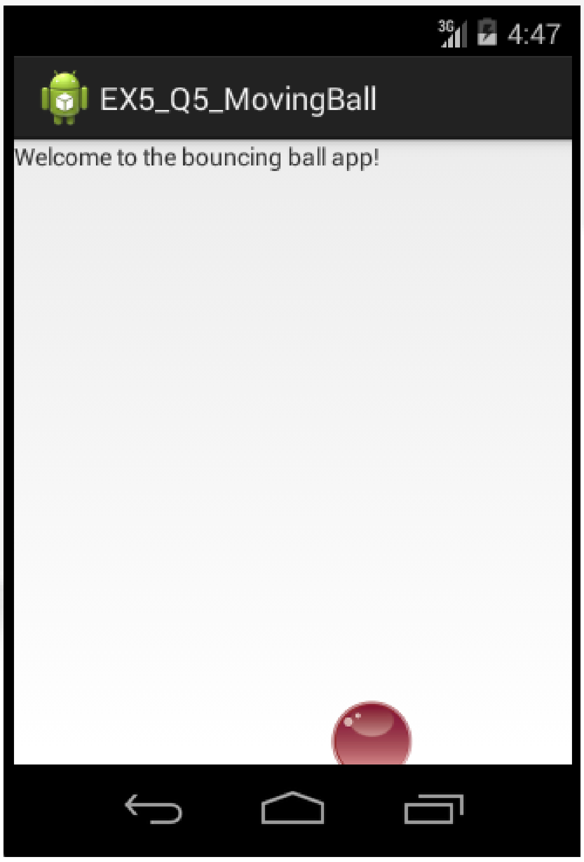
```
protected void startTimer() {

      timer.scheduleAtFixedRate(new TimerTask() {

            public void run() {

                  // signal UI thread to move ball

                  mHandler.obtainMessage(1).sendToTarget();

            }

      }, 500, 10);

}
```

```
public Handler mHandler = new Handler(new Handler.Callback() {
      @Override
      public boolean handleMessage(Message msg) {

            // ================================
            // Place your ball moving logic here
            // ================================

            return true;
      }
});
```

Example:



Welcome to the bouncing ball app!



Welcome to the bouncing ball app!

Question 6 –

In this question we will be creating a basic tic-tac-toe game, in which the player always starts and plays the X, and the computer always plays second with the O. The game ends when one of the players marked an entire row, column or a diagonal row.

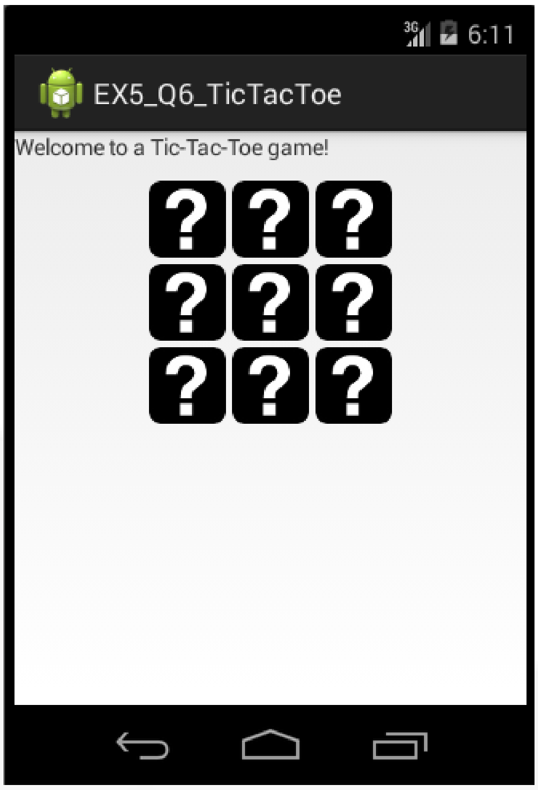
Comment1: We create the game board by placing 9 ImageView controls inside a GridLayout. Then we make sure that clicking each will cause an X to be marked in that imageView, this by changing the current image using this code:

```
public void onButtonsClick(View v) {

// Change image to X

ImageView clickedImageView = (ImageView)v;

clickedImageView.setImageResource(R.drawable.x);


// disable current image so it can't be clicked again

     clickedImageView.setEnabled(false);

}
```

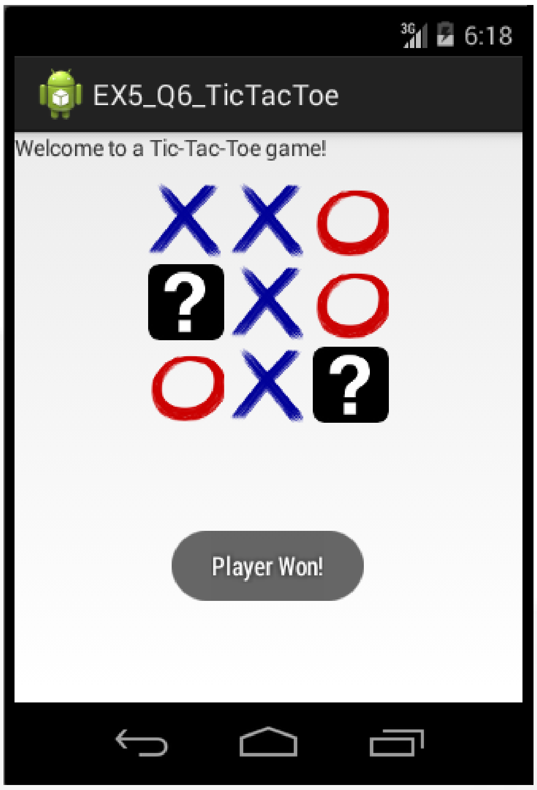
Comment2: After the player clicked the desired location we will run an algorithm that helps the computer choose his next location. To keep things simple will set the computer to choose a location randomly.

Comment3: A good way to manage the game in memory is by creating a 3x3 integer matrix, when a value of zero in a cell represents a non marked cell. A value of 1 indicates the cell was marked by the player, and a value of -1 represents a cell marked by the computer. The game ends when the sum of a row, column or diagonal row equals 3 (player won), equals -3 (computer won) or no more moves are available and no one won yet.

Example:



Welcome to a Tic-Tac-Toe game!



Welcome to a Tic-Tac-Toe game!

Good luck!