ELEC 279 - Winter 2022
Introduction to Object Oriented Programming
Assignment III

Due Date: Thursday April 7th, 2022 at 9:00 PM.

Objective

Practise Android programming by implementing a game with GUI.

Submission Guide

- Submit a **zip** folder containing your entire **Android Project Folder** plus a packaged app **.apk** file to "Assignment III" Dropbox folder on onQ.
- **Late submission:** If you are late, for every 2 hours, you will lose **15**% of your assignment marks.
- Add a comment at the beginning of your source code to include your name and student number as a proof of authorship.
- Ensure that your Java source code has good indentation, proper white spaces, clear variable names, and appropriate comments. Add comments for important parts of your code to explain what the code is doing.
- **CAUTION:** This is an individual assignment. You are allowed to discuss the assignment but any form of plagiarism will lead to a score of 0 for your assignment. Please kindly understand that we have zero tolerance to plagiarism.

Grading

- 100% - Solve the problem as expected
  - 40% - Correctly implement *GuessMaster.java* as the main activity file (or launcher) in Android
  - 15% - Correctly define all app components in Android Manifest file
  - 25% - Correctly implement GUI with an acitivty **.xml** file
  - 20% - Game works as expected using the GUI

Problem: GuessMaster Version 3.0

Assignment III is based on assignments I and II. You can continue with your assignment II source code or use the starting code we provide together with this assignment. In the former case, you may consider saving a copy of your assignment II code somewhere else for future reference.

This new version of game, GuessMaster 3.0, is played with the same rule as in assignment I and II. However, you will build a graphical user interface (GUI) in Android Studio. You do **NOT** need to modify the Date, Entity, Person, Politician, Country and Singer classes. You will basically be reusing these classes in this assignment and modifying GuessMaster.java class.

Moving from a Command Line program to GUI, you need to use **AlertDialog** in Android to print welcome message and a closing message to your user. As in Assignment II, your user will be awarded some tickets if his or her guess is correct. When a user's guess is correct or incorrect, you will use **AlertDialog** to inform the user. While working on this assignment, keep in mind that all initial actions or instances of variables should be implemented in the **onCreate()** method as discussed in class.

Figure 1: Sample Screenshot

A Recap of Assignment II

In assignment II, you designed a simple game that allows users to guess the birthday of a named entity, e.g., a celebrity, country, or prominent leader. It requires a player to guess the correct birthday held secretly in the computer program (e.g. Justin Trudeau was born on December 25, 1971). In this assignment, you will create a GUI that looks similar to the one above in Fig. 1(*Feel free to be as creative as possible with the interface color, layout etc., but do not make your project too complicated*).

To start, the program welcomes the player and inform him/her whose birthday to guess. More specifically, from a set of entities saved or hard-coded in the program, the program randomly picks one entity (e.g., a celebrity or country), show the corresponding image of the entity using **ImageView** and print the name on the screen using **TextView** and ask the user to guess the birth date using an **EditText**. After reading the answer from the **EditText**, the program responds with *"Incorrect. Try an earlier date."* or *"Incorrect. Try a later date."*, if the user's guess is incorrect, or *"BINGO. You got it!"* if the user is right. All these responses will be shown to user using **AlertDialog**.

Assignment III Instructions

- Create a new project in Android studio call GuessMaster, keep the default SDK version and choose Phone and Tablet as the platforms. Name the activity java class as GuessMaster.java.

- When you create your project, specify **activity_guess_activity** as the **.xml** layout file name and modify it as follows (*Create these items in this order to obtain layout similar to Fig. 1*):

  (i) Create a TextView with **android:id="@+id/ticket"** to store user ticket later in the game. Remember from our Android lectures that **id** allows us to reference each item in our .xml file in the Java or Activity class file.

  (ii) Create an ImageView with an id **entityImage** to display the image of an entity later on.

  (iii) To print the name of current entity, add a TextView with id **entityName**

  (iv) To accept input from user, add an EditText with an id **guessinput**.

  (v) Now, we need two Buttons to control user interaction with our game app. For the layout in Fig. 1, add **TableLayout** and inside the **TableLayout** tags, add **TableRow**. Inside the **TableRow** tags, add one Button with id **btnGuess** and below it, add another Button with id **btnClear**.

  (vi) We are now done with the layout file. Make sure the main container of your layout file is **LinearLayout**. If you get stuck, please, see sample code (Android In-class Example) on onQ under Week 9.

- Before you modify the GuessMaster class, copy your .java classes (Date, Entity, Person, Politician, Country and Singer) from Assignment II into the current Android project directory, paste them in the src folder where you have GuessMaster.java

class under your package directory. **Note:** You might need to remove assignment II package name and add the current package name for Android project in each class you just copied.

- Next, modify your *GuessMaster* class to be an **activity** .java class that **extends** AppCompatActivity specify it as the main acitivity class in the **AndroidManifest.xml** file if Android studio hasn't done that for you when you created the project.

  (i) Specify the GuessMaster as an activity class using the AppCompatActivity as thus: **public class GuessMaster extends AppCompatActivity**. If you follow the new project creation dialog when creating your project, Android studio does this for you by default. If not, let your GuessMaster class extends AppCompatActivity and add **import android.support.v7.app.AppCompatActivity**;

  (ii) Import the widget class object since we will be using **TextView**, **EditText**, **Button** and **ImageView**. add **import android.widget.***;

  (iii) In the class-level of your GuessMaster class, define the following view components:
  private TextView entityName;
  private TextView ticketsum;
  private Button guessButton;
  private EditText userIn;
  private Button btnclearContent;
  private String user_input;
  private ImageView entityImage;
  String answer;

  (iv) Copy your instances and variables from GuessMaster file in Assignment II and paste in the class-level as well; your instances might look something like this:
  private int numOfEntities;
  private Entity[] entities;
  private int[] tickets;
  private int numOfTickets;
  //Stores Entity Name
  String entName;
  int entityid = 0;
  int currentTicketWon = 0;

  (v) Now, let us create the entity objects right after the view components. You can copy the codes from your Assignment II GuessMaster class and paste it right before the onClickListener methods. Your code to create the objects might look like this snippet:
  Country usa= new Country("United States", new Date("July", 4, 1776), "Washingston DC", 0.1);
  Person myCreator= new Person("myCreator", new Date("May", 6, 1800), "Male", 1);
  Politician trudeau = new Politician("Justin Trudeau", new Date("December",25,1971),"Male",

"Liberal", 0.25);
Singer dion= new Singer("Celine Dion", new Date("March", 30, 1961), "Female", "La voix du bon Dieu", new Date("November",6,1981),0.5);
Add final keyword so that we can access the class within. **final** GuessMaster gm = new GuessMaster();

(vi) Also, copy all methods (e.g. playGame(), addEntity(), genRandomEntityId() etc.) and constructor (public GuessMaster() from GuessMaster file in Assignment II and add them to the class-level of GuessMaster class in Android project. Remove the while() loops in the two playGame() methods, we will not that since most actions or events in the game will be triggered by Button click. Make sure these methods are pasted outside of the **onCreate()** call-back method.

(vii) Now, dive into the **onCreate()** method in your GuessMaster acitivity class. In the setContentView( ) method, define the layout file to be used by your activity:
//Set the xml as the activity UI view
setContentView(R.layout.activity_guess_master);

(viii) Right after the setContentView( ) method in the **onCreate()** , define the view components defined earlier in ((iii)):
//Specify the button in the view
guessButton = (Button) findViewById(R.id.btnGuess); (*Please note that the R.id references the name of the Guess Button defined in the xml file*)
//EditText for user input
userIn = (EditText) findViewById(R.id.guessinput);

//TextView for total tickets
ticketsum = (TextView) findViewById(R.id.ticket);
Define the remaining view components yourself.

(ix) Think about the importance of **onCreate()** method in the life-cycle of an app as discussed in class (refer to lecture slides on Android programming). While you are still in the **onCreate()**, we need to define actions to perform when the two Buttons defined earlier are clicked by the user. Define the actions as follows each button using the setOnClickListener() method:

```
//OnClick Listener action for clear button
    btnclearContent.setOnClickListener(new View.OnClickListener(){
        @Override
        public void onClick(View v){
            changeEntity();
        }
    });
```

```
//OnClick Listener action for submit button
    guessButton.setOnClickListener(new View.OnClickListener(){
        @Override
        public void onClick(View v){
            //playing game
```

```
                playGame();
            }
        });
```

(x) Now, create the a method: **changeEntity()** to clear user entries from the **userIn** EditText and randomly choose another entity.

(xi) **Setting Images:** Each entity that is randomly selected, there should be a corresponding image displayed in the **ImageView** defined earlier. Download the three (3) images provided with this assignment instruction. One is a USA flag, one is Trudeau's picture and the third one is Celine's picture. Copy this pictures to the **drawable** folder in your project directory. You should include any picture for the **myCreator** entity because you will need four (4) pictures in total. Then, create a method called **ImageSetter()** that sets appropriate picture for each entity when selected; you can use **switch..case** or **if..else** statements and use **equals()** method to compare strings.

(xii) In this version, you should print the welcome message once as opposed to printing it multiple times in version 2. Create a method called **welcomeToGame(Entity entity)** and add the AlertDialog that shows your welcome message as follows:
You can remove the Scanner instances since we are no longer using the console. To welcome user to your game, you will print the message to screen using **AlertDialog** instead of the **System.out.println**. To use **AlertDialog**,

- Import android.content.DialogInterface;
- Define your welcome message inside welcomeToGame(Entity Entity) as follows:

```
        //Welcome Alert
AlertDialog.Builder welcomealert =
 new AlertDialog.Builder(GuessMasterActivity.this);
        //System.out.println("(mm/dd/yyyy)");
        welcomealert.setTitle("GuessMaster Game v3");
        welcomealert.setMessage(entity.welcomeMessage());
        welcomealert.setCancelable(false); //No Cancel Button

     welcomealert.setNegativeButton("START GAME", new
DialogInterface.OnClickListener() {
            @Override
 public void onClick(DialogInterface dialop, int which) {
 Toast.makeText(getBaseContext(), "Game is Starting...
Enjoy", Toast.LENGTH_SHORT).show();
            }
        });
        //Show Dialog
        AlertDialog dialog = welcomealert.create();
        dialog.show();
```

(xiii) Modify your playGame(Entity Entity) where the game engine is located. First, remove the **while(true)** except for its contents. We do need not a while() loop

here. As long as the activity is the current user view, the playGame(Entity Entity) method will be called every time a user clicks on the guessButton. While inside this method, remove the Scanner.nextLine() since we are not getting input from command line. Print the entity name to the entityName TextView and get the input from the EdiText userIn:

```
//Name of the entity to be guessed in the entityName textview
        entityName.setText(entity.getName());
            //Get Input from the EdiText
            answer = userIn.getText().toString();
            answer = answer.replace("\n", "").replace("\r", "");
            Date date = new Date(answer);
```

- For the two incorrect cases shown below in the playGame(Entity Entity) , create AlertDialog for each. Set their titles as "Incorrect." The alert message for the first one should be "Try a later date," and "Try an earlier date" for the second one. Then, use setNegativeButton("Ok", new ...) for each dialog.

```
//Check User Date Input
            if (date.precedes(entity.getBorn())) {
               // System.out.println("Incorrect. Try a later date.");
                ...
 } else if (Entity.getBorn().precedes(date)) {
...
} else {

              tickets[numOfTickets++] = Entity.getAwardedTicketNumber();
                 for (int i = 0; i < 100; i++) {
                     totaltik = totaltik + tickets[i];
                 }
//Use alert here to let user know they have won

//Call the ContinueGame() method inside the onClick() method
//of the DialogInterface.
}
```

- After the if...else block, use AlertDialog to inform users that they have won. Use the setNegativeButton("Continue", new ...) to continue playing the game. set title as "You won" and message should be "BINGO! "+ Entity.closingMessage(). In the onClick() method of the setNegativeButton(), make sure you add the awarded ticket to the Toast.makeText() using the getAwardedTicketNumber() method in Entity class.

- Lastly, using the setText() method update the ticketsum TextView with the total ticket obtained. To ensure that the game **Continues**, create a method called **ContinueGame()** with the below sample code and call this method in the if...else statement that checks if user wins. Logically, the ContinueGame() method should be called after a user wins. Call the ContinueGame() method inside the onClick() method of the DialogInterface.

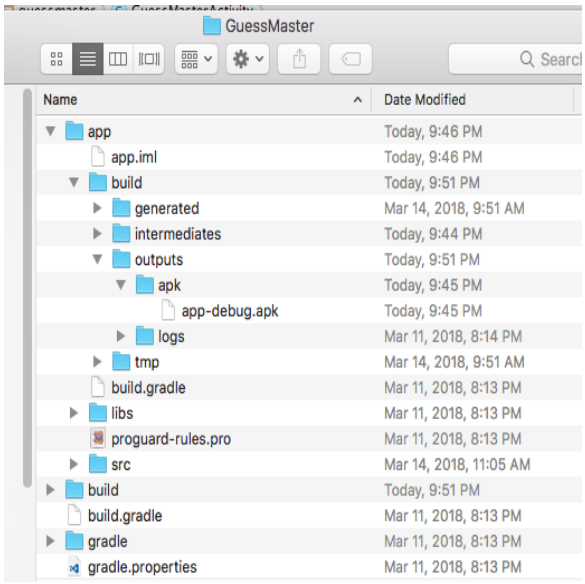

Figure 2: Sample Screenshot

```
    //Continue Game Method - This method gets another Entity randomly
    //Updates the Image view and the Entity Name Field
    public void ContinueGame(){
        entityid = genRandomEntityId();
        Entity entity = entities[entityid];

        entName = entity.getName();

        //Call the ImageSetter method
        ImageSetter();

        //Print the name of the entity to be guessed
        //in the entityName textview
        entityName.setText(entName);
        //Clear Previous Entry
        userIn.getText().clear();
    }
```

- Test your code using physical device or the Emulator. See sample GUI on onQ under Week 11 Assignment III folder.
- Packaging your app as .apk: Go to Build → Build apk. Find apk in the project directory, see example in Fig 2. Submit this apk file along with your project file. COPY THIS apk file to the home directory of your project folder (THANK YOU :))

HOPE YOU ENJOY ANDROID PROGRAMMING :)