Lab assignment 3

Due date: Saturday, June 4

20 points

Team of 4 students

The group works on the lab assignment and can discuss the code with another group. Each group demonstrates the lab to each other. The team then upload the solution from one of the subgroups.

Write the contribution to the lab assignment, the contribution percentage (maximum of 100%) on the comment box for each team member.

All the team members must agree on the percentage. If the team members have 50% and the lab assignment grade is 40/40, they get only 20 points. The instructor will review the paper summarizing and validate the percentage of each team member. If you have problems with team members, please let me know.

Problem

In this lab assignment you will learn how to set up transitions between them, and how to save the state of an app and retrieve it whenever the user starts the app again (i.e., how to make the data persistent). We continue to build a mortgage calculator app as a vehicle to learn all these concepts.

Version 1. Persistent Data

Coding

Mortgage class

```
  import android.content.Context;
import android.content.SharedPreferences;
import android.preference.PreferenceManager;

import java.text.DecimalFormat;

public class Mortgage {
    public final DecimalFormat MONEY
            = new DecimalFormat( "$#,##0.00" );
    private static final String PREFERENCE_AMOUNT = "amount";
    //YOUR CODE

    private float amount;
    private int years;
    private float rate;

    public Mortgage( ) {
        setAmount( 100000.0f );
        setYears( 30 );
        setRate( 0.035f );
    }
```

```
// Instantiate Mortgage from stored preferences
public Mortgage( Context context ) {
    SharedPreferences pref =
            ____________________________________________
    setAmount( pref.getFloat( PREFERENCE_AMOUNT, 100000.0f));
    ____________________________________________
    ____________________________________________
}


public void setAmount( float newAmount ) {
    if( newAmount >= 0 )
        amount = newAmount;
}
public void setYears( int newYears ) {
    if( newYears >= 0 )
        years = newYears;
}

public void setRate( float newRate ) {
    if( newRate >= 0 )
        rate = newRate;
}

public float getAmount( ) {
    return amount;
}

public String getFormattedAmount( ) {
    return MONEY.format( amount );
}

public int getYears( ) {
    return years;
}

public float getRate( ) {
    return rate;
}

public float monthlyPayment( ) {
    float mRate = rate / 12;  // monthly interest rate
    double temp = Math.pow( 1/( 1 + mRate ), years * 12 );
    return amount * mRate / ( float ) ( 1 - temp );
}

public String formattedMonthlyPayment( ) {
    return MONEY.format( monthlyPayment( ) );
}

public float totalPayment( ) {
    return monthlyPayment( ) * years * 12;
}

public String formattedTotalPayment( ) {
```

```
        return MONEY.format( totalPayment( ) );
    }
    // Write mortgage data to preferences
    public void setPreferences( Context context ) {
        SharedPreferences pref =
                ____________________________________________________
        ____________________________________________
        editor.putFloat( PREFERENCE_AMOUNT, amount );
        //YOUR CODE
    }
}
```

MainActivity class

```
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
   // mortgage = new Mortgage( );
    mortgage = ____________________________
```

DataActivity class

```
try {
    float amount = Float.parseFloat( amountString );
    mortgage.setAmount( amount );
    float rate = Float.parseFloat( rateString );
    mortgage.setRate( rate );
    ____________________________________
} catch( NumberFormatException nfe ) {
    mortgage.setAmount( 100000.0f );
    mortgage.setRate( .035f );
}
```

Version 2 – Transitions

We want to improve Version 2 by adding animated transitions between the two screens.
A transition is typically an animation special effect when going from one screen to another: for example, we can fade out of the current screen into the new one, or fade in to present the new screen, or bring a screen with a sliding motion from left to right (or right to left). Two types of animations can be used: tween animation and frame animation. A tween animation is defined with its starting and ending points, and intermediary frames are automatically generated. A frame animation is defined by using a sequence of various images from the beginning to the end of the animation.

In Version 2, we make a tween animation that slides from left to right to go from the first to the second screen, and a combination of fade in and scaling transitions to come back from the

second screen to the first one. Like a layout, a String, or a style, a transition can be defined as a resource defined in an XML file. It can also be defined programmatically.

Create a folder called anim under the folder res. Create two files called fade_in_and_scale.xml and slide_from_left.xml in the folder anim.

fade_in_and_scale.xml

```
<?xml version="1.0" encoding="UTF-8"?>

   <set xmlns:android="http://schemas.android.com/apk/res/android">

   <alpha android:duration="3000" android:toAlpha="1.0" android:fromAlpha="0.0"/>

   <scale android:duration="3000" android:pivotY="50%" android:pivotX="50%"
android:toYScale="1.0" android:fromYScale="0.0" android:toXScale="1.0"
android:fromXScale="0.0"/>

</set>
```

slide_from_left.xml

```
<?xml version="1.0" encoding="UTF-8"?>

   <set xmlns:android="http://schemas.android.com/apk/res/android">

   <translate android:duration="4000" android:toXDelta="0" android:fromXDelta="-100%p"/>

</set>
```

MainActivity class

```
public void modifyData( View v ) {
    Intent myIntent = new Intent( this, DataActivity.class );
    this.startActivity(myIntent);
    overridePendingTransition( _____________________________ );
}
```

DataActivity class

```
public void goBack( View v ) {
    updateMortgageObject();
    this.finish( );
    overridePendingTransition( ____________________________ );
}
```

Grading:

1. Submit the project in zip file (extension zip)
2. A pdf document that contains the content of the following files: AndroidMainifest.xml, activity_main.xml, activity_data.xml, themes.xml, MainActivity class, and DataActivity class
3. A zoom link or YouTube that demonstrate the features of the Mortgage Calculator app (transitions and persistent data) less than 3 minutes. Write the link on the comment section in the Dropbox.
   Run the application first time.

   Modify the following data:
   Amount: 400000.00
   Years: 10
   Interest rate: 3.5%

   Then rerun the application

   (When we run the app the second time, the data that we entered on the second screen the first time we ran the app is now shown on the first screen. The app is pulling data from the preferences that were written into the first time we ran the app)

Note:
App is running correctly and met all the requirements is 20 points, but
Incomplete item 1(-20 points)
Incomplete item 2 (-5 points)
Incomplete item 3 (-5 points)