UNIVERSITI TUNKU ABDUL RAHMAN

Faculty of Information and Communication Technology

UCCD3223 Mobile Applications Development

(Jan 2024 Trimester)

Individual Practical Assignment

Name	Liong Jun Yong
Student ID	20ACB04719
Course	CS
Practical Group	P9
Lecturer	Mr. Tan Chiang Kang @ Thang Chiang Kang

Marking scheme	Marks	Remarks
Correctness	× **2.5**	
Design	× **3.5**	
User Friendliness	× **2**	
Neat Program Documentation		
Report Format		
TOTAL		

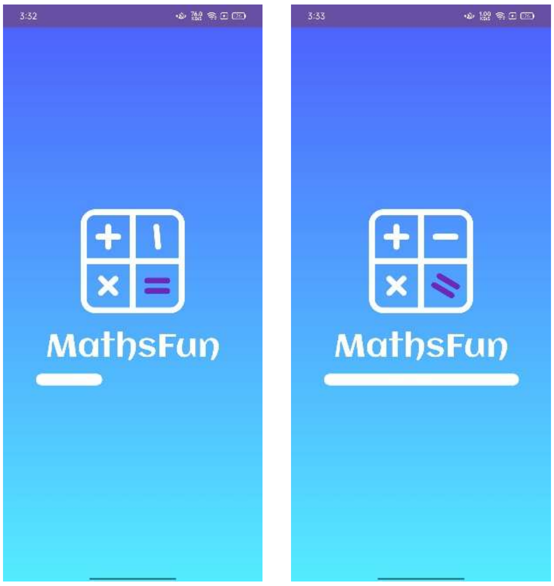
TABLE OF CONTENTS

Introduction

The Android application "MathsFun" developed by Liong Jun Yong is a mathematics-learning application specifically designed for school-going children. It primarily consists of three modules aimed to provide children with foundational mathematical skills, including comparing numbers, ordering numbers, and composing numbers. To guarantee user-friendliness to the children, numbers appearing in the application will never be more than 3 digits and strictly in the form of whole numbers only. The application is developed using Java with Android Studio and can be found in the GitHub repository.

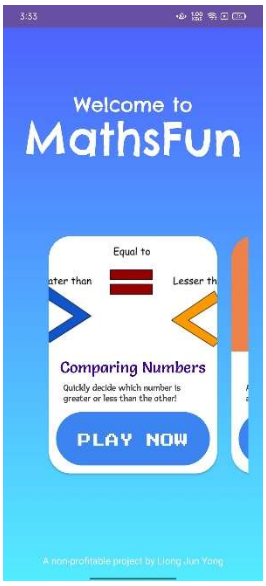
Flow and application highlights

1. *Splash screen loading animation*



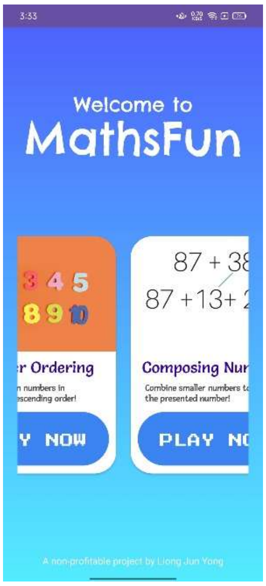
Upon the first launch of the application, users will be greeted with a loading animation. The animation resource file is in the form of JSON (.json) found on LottieFiles. The mathematical elements inside the rounded square will rotate at their fixed axes until the end of the animation. The progress bar will simulate the process of loading necessary content as the progress goes from 0% (left) to 100% (right).

2. *Main page / Home / Homepage*



Quickly decide which number is greater or less than the other!

A non-profitable project by Liong Jun Yong



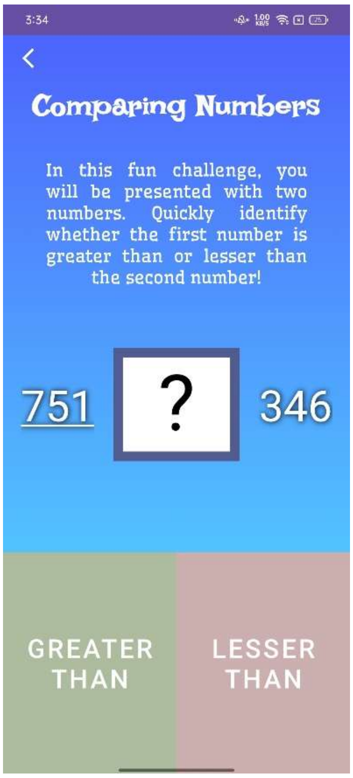
n numbers in
scending order!

Combine smaller numbers t
the presented number!

A non-profitable project by Liong Jun Yong

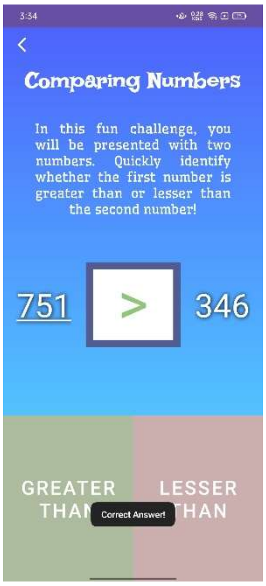
The main page or the home page is the home to the MathsFun application. A simple description of the playable mini games or modules along with their respective illustrations exist in the form of card widgets. Three cards indicating one module each, are switchable or interchangeable by swiping among them. At the bottom of each card lies a “PLAY NOW” clickable button that directs the user to the cards’ respective activities. If users perform backward navigations in any other activities or pages other than the main page, they will be directed from their current page back to the main page.

3. *Module 1: Comparing Numbers*



In this fun challenge, you will be presented with two numbers. Quickly identify whether the first number is greater than or lesser than the second number!

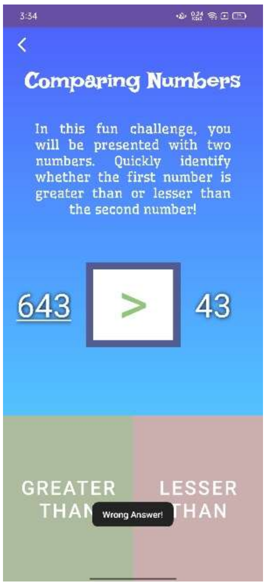
Upon entering the first module “Comparing Numbers” by clicking on the button on its respective card, a simple description of the game and its instructions are presented for the better understanding of users. In each execution of the ComparingNumbersActivity.java, users will be presented with two randomly-generated integers where they are required to identify the relationship between them by comparing both given numbers. At the bottom of the layout lies two clickable buttons for the users to choose from, namely “greater than” and “lesser than”. Users will need to identify the first number given and then compare it with the second to determine whether the first is greater than the second or vice versa.



Comparing Numbers

In this fun challenge, you will be presented with two numbers. Quickly identify whether the first number is greater than or lesser than the second number!

Correct Answer!

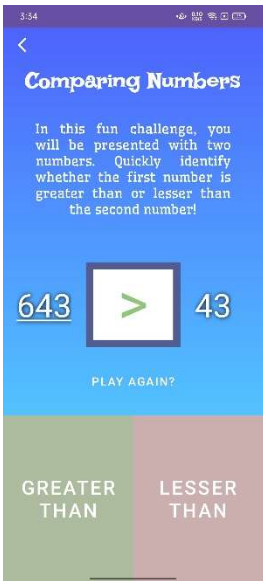


Comparing Numbers

In this fun challenge, you will be presented with two numbers. Quickly identify whether the first number is greater than or lesser than the second number!

Wrong Answer!

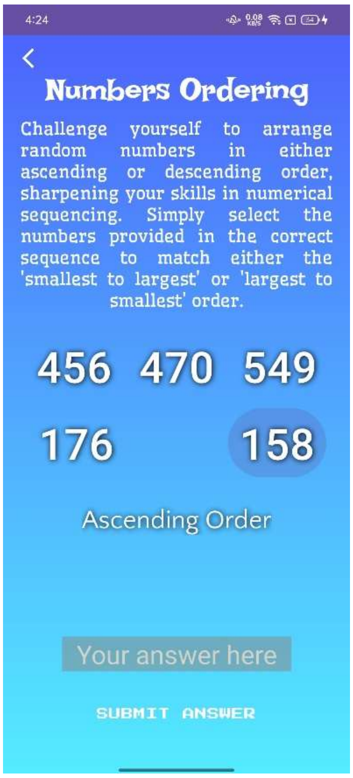
No matter which answers the user decides to submit through clicking their respective buttons, a toast notification will be displayed after an answer has been made. If the user provided the correct answer, a toast notification telling the user has gotten a correct answer will appear. In contrast, a “Wrong Answer!” toast notification will appear to each incorrect answer from the user.



In this fun challenge, you will be presented with two numbers. Quickly identify whether the first number is greater than or lesser than the second number!

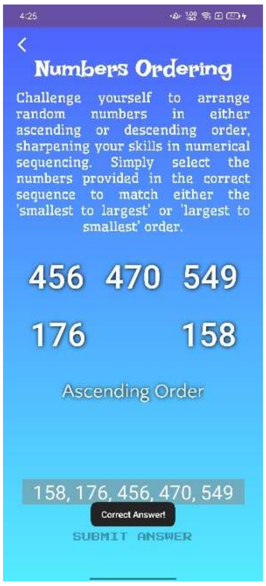
After the system provides feedback on every answering attempt the user has made, a "PLAY AGAIN?" button will appear, restarting ComparingNumbersActivity.java on click to allow users to replay or retry the game again. When the user decides to quit or exit this current page, back navigation is made possible by clicking the "back" icon button located at the top right corner of the layout or using Android's built-in back navigations.

4. Module 2: Number Ordering



Challenge yourself to arrange random numbers in either ascending or descending order, sharpening your skills in numerical sequencing. Simply select the numbers provided in the correct sequence to match either the 'smallest to largest' or 'largest to smallest' order.

Clicking the "PLAY NOW" button on the second card leads the user into the second module "Numbers Ordering". Similarly, a simple description of the game and its instructions are presented for the better understanding of users. In each execution of the NumberOrderingActivity.java, users will be presented with five randomly-generated integers in the form of clickable buttons and one specified sorting sequence among "Ascending Order" and "Descending Order" where they are required to reorder the numbers given in the correct sequence according to the requirement specified by the system. There is a text view to display the numbers that the user has chosen and inserted into an array at the bottom of the layout. Upon clicking an option among the five given numbers and hold, the background of the selected option will change color to indicate that this option is currently selected by the user before the user confirms and releases the button to insert the selected number into the "chosen" array. As users continues to pick the numbers according to their desired sequence, the text view below will display the numbers chosen by the user in their chosen sequence.



4:25

Numbers Ordering

Challenge yourself to arrange random numbers in either ascending or descending order, sharpening your skills in numerical sequencing. Simply select the numbers provided in the correct sequence to match either the 'smallest to largest' or 'largest to smallest' order.

456 470 549

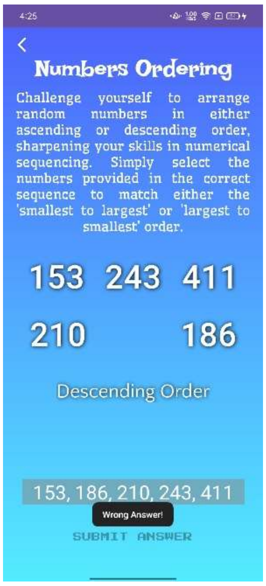
176 158

Ascending Order

158, 176, 456, 470, 549

Correct Answer!

SUBMIT ANSWER



4:25

Numbers Ordering

Challenge yourself to arrange random numbers in either ascending or descending order, sharpening your skills in numerical sequencing. Simply select the numbers provided in the correct sequence to match either the 'smallest to largest' or 'largest to smallest' order.

153 243 411

210 186

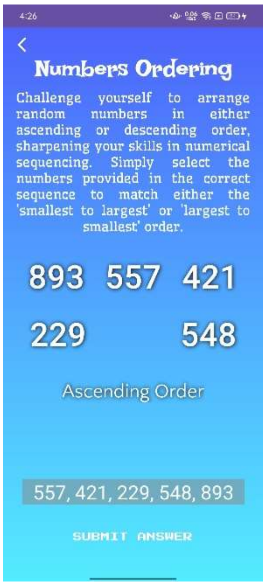
Descending Order

153, 186, 210, 243, 411

Wrong Answer!

SUBMIT ANSWER

Upon submitting an answer, the system performs a validation check. If the user gave an answer that is detected to be in ascending order when the system asks for indeed ascending order, the system will display a toast notification to indicate the user's correct answer attempt. In contrast, when the system asks for descending order but the answer that the user gave is detected to be opposite, the system shows a "Wrong Answer!" toast notification to tell the user that he/she has given an incorrect answer.



4:26

Numbers Ordering

Challenge yourself to arrange random numbers in either ascending or descending order, sharpening your skills in numerical sequencing. Simply select the numbers provided in the correct sequence to match either the 'smallest to largest' or 'largest to smallest' order.

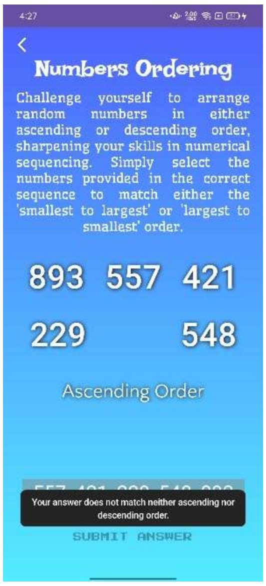
893 557 421

229 548

Ascending Order

557, 421, 229, 548, 893

SUBMIT ANSWER



4:27

Numbers Ordering

Challenge yourself to arrange random numbers in either ascending or descending order, sharpening your skills in numerical sequencing. Simply select the numbers provided in the correct sequence to match either the 'smallest to largest' or 'largest to smallest' order.

893 557 421

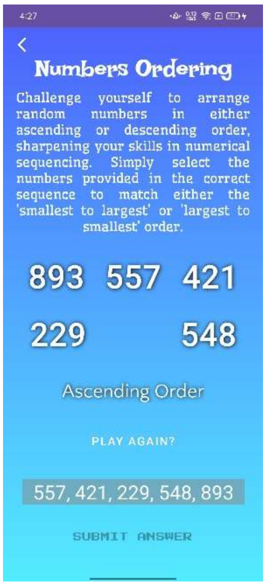
229 548

Ascending Order

Your answer does not match neither ascending nor descending order.

SUBMIT ANSWER

Another test case is when the user provides an answer that is not detected to be in either ascending order or descending order. If the user's answer cannot fulfil either one of the conditions, a special toast notification appears, informing the user that his/her answer does not match any of the sorting sequences available to the system's knowledge.

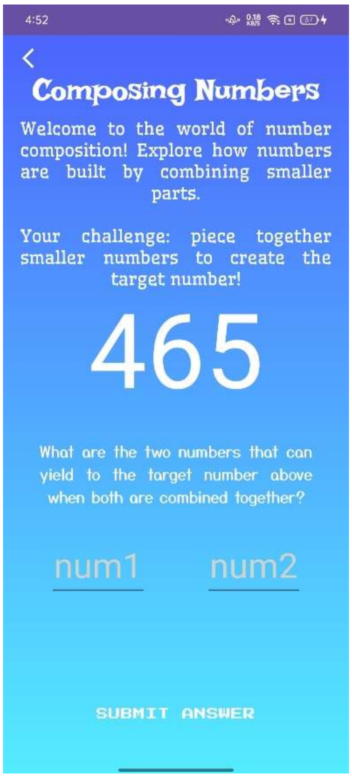


Numbers Ordering

Challenge yourself to arrange random numbers in either ascending or descending order, sharpening your skills in numerical sequencing. Simply select the numbers provided in the correct sequence to match either the 'smallest to largest' or 'largest to smallest' order.

After the system provides feedback on every answering attempt the user has made, a "PLAY AGAIN?" button will appear, restarting NumberOrderingActivity.java on click to allow users to replay or retry the game again. The "SUBMIT ANSWER" button will also be greyed out, stopping users from submitting any extra answers. When the user decides to quit or exit this current page, back navigation is made possible by clicking the "back" icon button located at the top right corner of the layout or using Android's built-in back navigations.

5. *Module 3: Composing Numbers*



Composing Numbers

Welcome to the world of number composition! Explore how numbers are built by combining smaller parts.

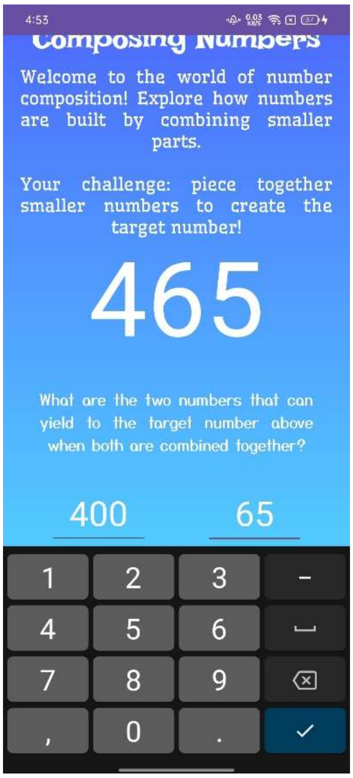
Your challenge: piece together smaller numbers to create the target number!

465

What are the two numbers that can yield to the target number above when both are combined together?

num1 num2

SUBMIT ANSWER



Composing Numbers

Welcome to the world of number composition! Explore how numbers are built by combining smaller parts.

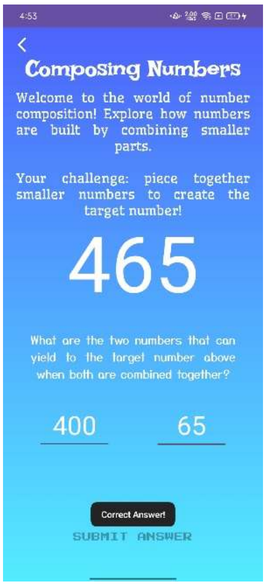
Your challenge: piece together smaller numbers to create the target number!

465

What are the two numbers that can yield to the target number above when both are combined together?

400 65

In the third also the last module “Composing Numbers”, a simple description of the game and its instructions are provided for the better understanding of users in the same way the other two modules did. In each execution of the ComposingNumbersActivity.java, users will be presented with one randomly-generated integer where they are required to figure out how the provided number can be decomposed into two smaller integer numbers. Two EditText input fields with grey hints are provided to receive the user’s input entries. In the input fields, only numbers can be selected and submitted for entry.


4:53

Composing Numbers

Welcome to the world of number composition! Explore how numbers are built by combining smaller parts.

Your challenge: piece together smaller numbers to create the target number!

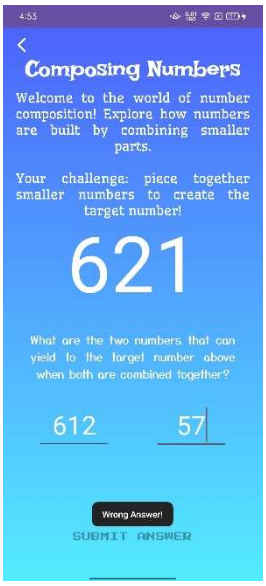
465

What are the two numbers that can yield to the target number above when both are combined together?

400 65

Correct Answer!

SUBMIT ANSWER


4:53

Composing Numbers

Welcome to the world of number composition! Explore how numbers are built by combining smaller parts.

Your challenge: piece together smaller numbers to create the target number!

621

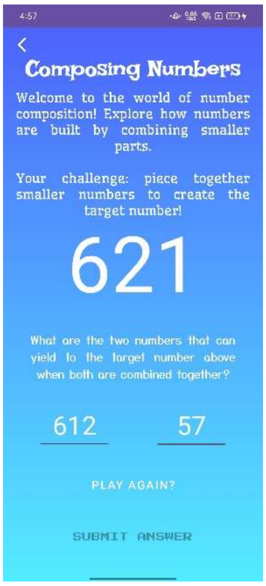
What are the two numbers that can yield to the target number above when both are combined together?

612 57

Wrong Answer!

SUBMIT ANSWER

After hitting the "SUBMIT ANSWER" button, the system will also perform a validation check to determine whether both user inputs can be combined together to obtain the target number which was specified by the system. If the user has provided a correct answer, a "Correct Answer!" toast notification will appear, indicating a correct answer attempt while if the user answered incorrectly, that is, both of his/her input entries do not form the target number when combined, a "Wrong Answer!" toast notification will be displayed.


4:57

Composing Numbers

Welcome to the world of number composition! Explore how numbers are built by combining smaller parts.

Your challenge: piece together smaller numbers to create the target number!

621

What are the two numbers that can yield to the target number above when both are combined together?

612 57

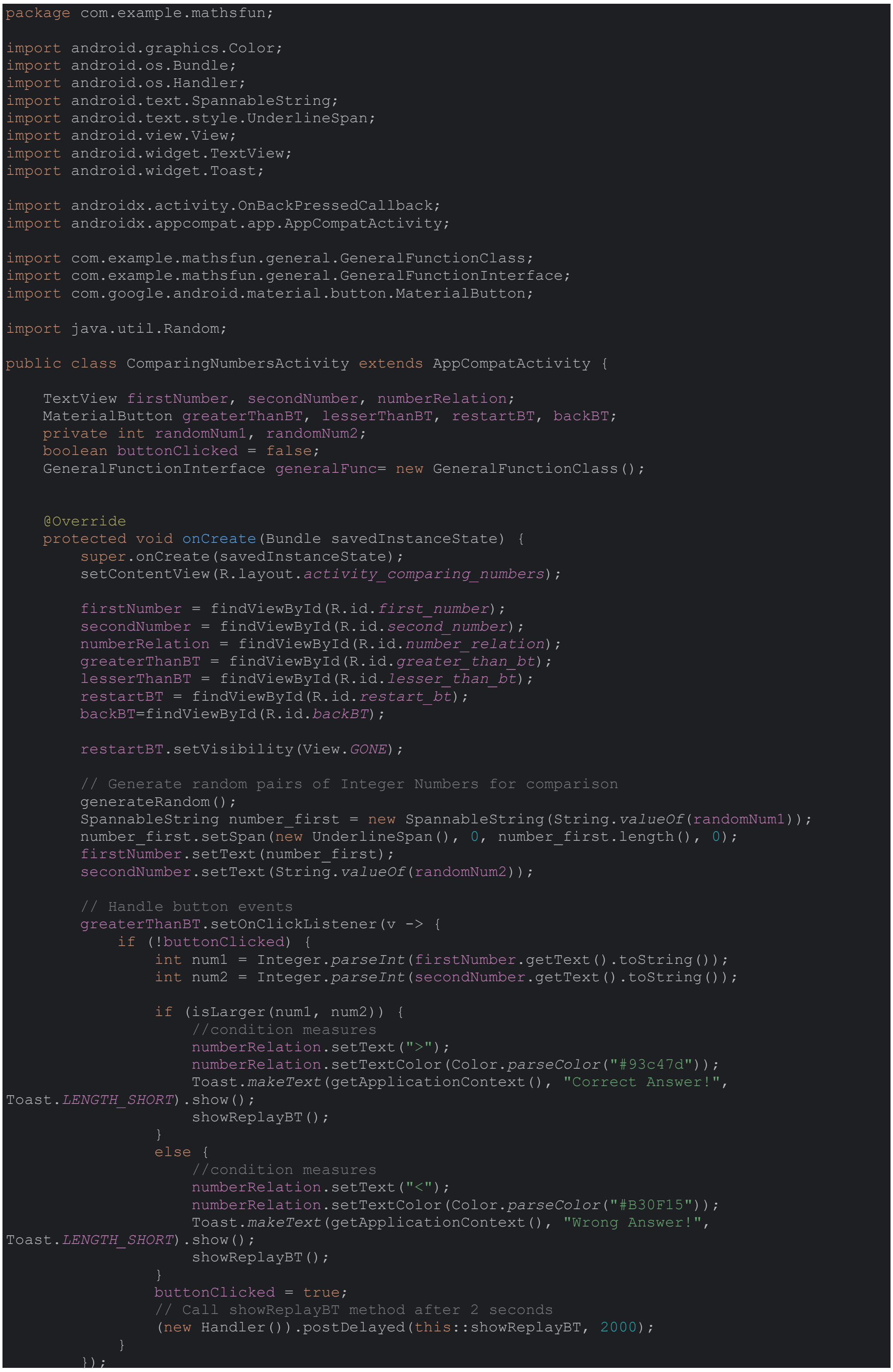
PLAY AGAIN?

SUBMIT ANSWER

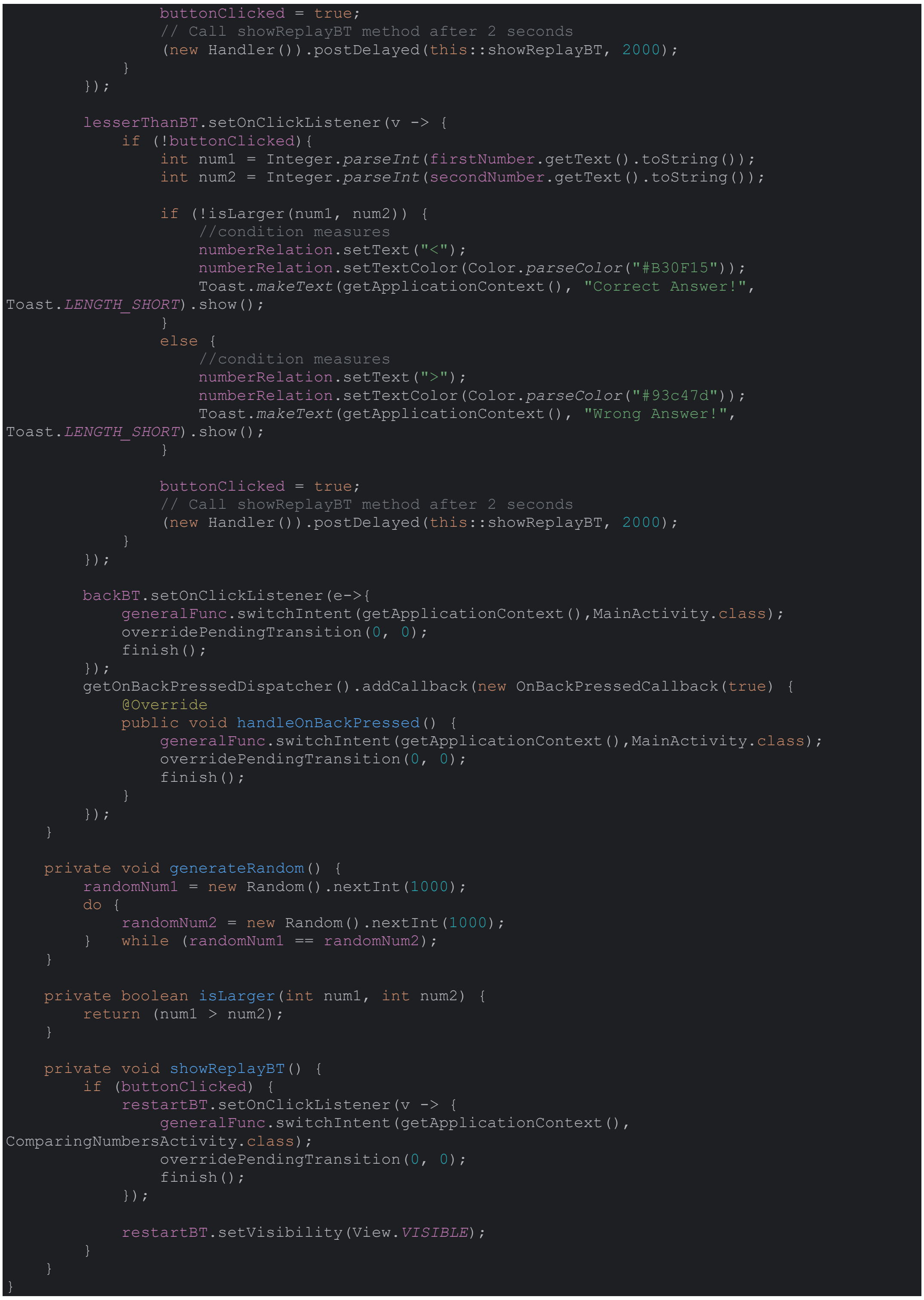
After the system provides feedback on every answering attempt the user has made, a "PLAY AGAIN?" button will appear, restarting ComposingNumbersActivity.java on click to allow users to replay or retry the game again. The "SUBMIT ANSWER" button will also be greyed out, stopping users from submitting any extra answers. When the user decides to quit or exit this current page, back navigation is made possible by clicking the “back” icon button located at the top right corner of the layout or using Android’s built-in back navigations.

Appendices

1. ComparingNumbersActivity.java



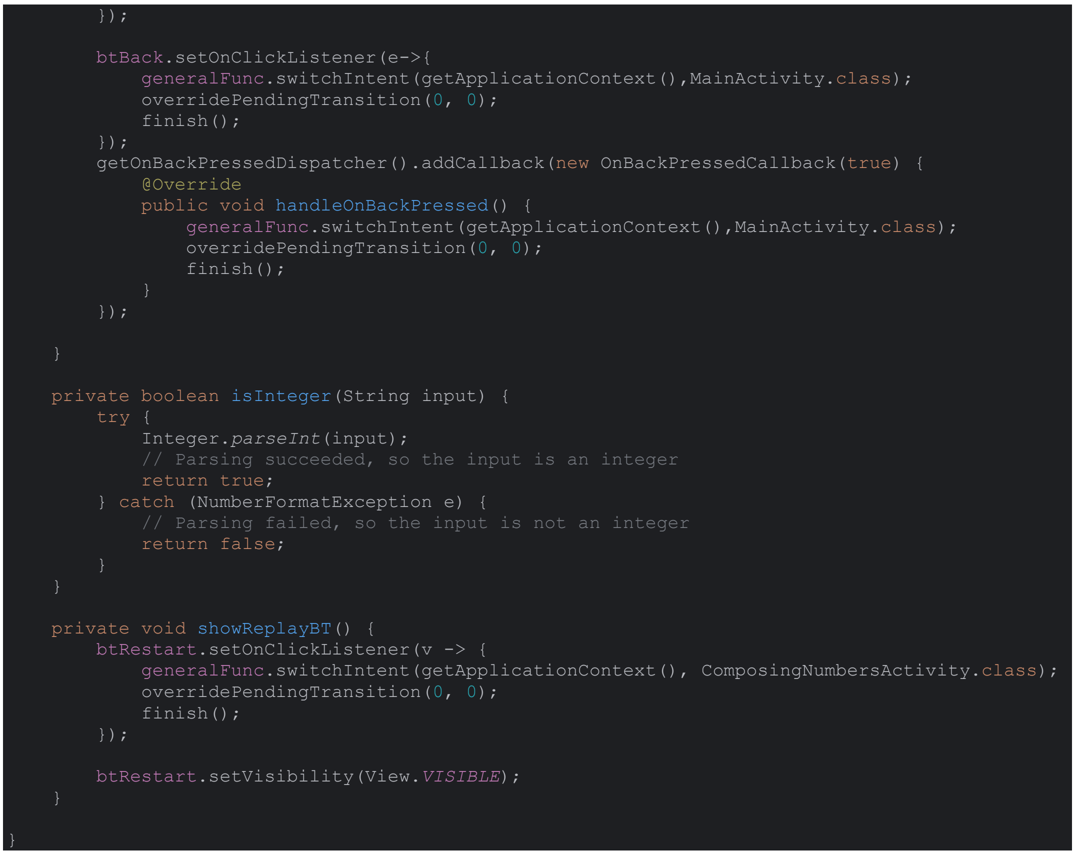
```java
package com.example.mathsfun;

import android.graphics.Color;
import android.os.Bundle;
import android.os.Handler;
import android.text.SpannableString;
import android.text.style.UnderlineSpan;
import android.view.View;
import android.widget.TextView;
import android.widget.Toast;

import androidx.activity.OnBackPressedCallback;
import androidx.appcompat.app.AppCompatActivity;

import com.example.mathsfun.general.GeneralFunctionClass;
import com.example.mathsfun.general.GeneralFunctionInterface;
import com.google.android.material.button.MaterialButton;

import java.util.Random;

public class ComparingNumbersActivity extends AppCompatActivity {

    TextView firstNumber, secondNumber, numberRelation;
    MaterialButton greaterThanBT, lesserThanBT, restartBT, backBT;
    private int randomNum1, randomNum2;
    boolean buttonClicked = false;
    GeneralFunctionInterface generalFunc= new GeneralFunctionClass();


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

        firstNumber = findViewById(R.id.first_number);
        secondNumber = findViewById(R.id.second_number);
        numberRelation = findViewById(R.id.number_relation);
        greaterThanBT = findViewById(R.id.greater_than_bt);
        lesserThanBT = findViewById(R.id.lesser_than_bt);
        restartBT = findViewById(R.id.restart_bt);
        backBT=findViewById(R.id.backBT);

        restartBT.setVisibility(View.GONE);

        // Generate random pairs of Integer Numbers for comparison
        generateRandom();
        SpannableString number_first = new SpannableString(String.valueOf(randomNum1));
        number_first.setSpan(new UnderlineSpan(), 0, number_first.length(), 0);
        firstNumber.setText(number_first);
        secondNumber.setText(String.valueOf(randomNum2));

        // Handle button events
        greaterThanBT.setOnClickListener(v -> {
            if (!buttonClicked) {
                int num1 = Integer.parseInt(firstNumber.getText().toString());
                int num2 = Integer.parseInt(secondNumber.getText().toString());

                if (isLarger(num1, num2)) {
                    //condition measures
                    numberRelation.setText(">");
                    numberRelation.setTextColor(Color.parseColor("#93c47d"));
                    Toast.makeText(getApplicationContext(), "Correct Answer!",
Toast.LENGTH_SHORT).show();
                    showReplayBT();
                }
                else {
                    //condition measures
                    numberRelation.setText("<");
                    numberRelation.setTextColor(Color.parseColor("#B30F15"));
                    Toast.makeText(getApplicationContext(), "Wrong Answer!",
Toast.LENGTH_SHORT).show();
                    showReplayBT();
                }
                buttonClicked = true;
                // Call showReplayBT method after 2 seconds
                (new Handler()).postDelayed(this::showReplayBT, 2000);
            }
        });
```



```
                buttonClicked = true;
                // Call showReplayBT method after 2 seconds
                (new Handler()).postDelayed(this::showReplayBT, 2000);
            }
        });

        lesserThanBT.setOnClickListener(v -> {
            if (!buttonClicked){
                int num1 = Integer.parseInt(firstNumber.getText().toString());
                int num2 = Integer.parseInt(secondNumber.getText().toString());

                if (!isLarger(num1, num2)) {
                    //condition measures
                    numberRelation.setText("<");
                    numberRelation.setTextColor(Color.parseColor("#B30F15"));
                    Toast.makeText(getApplicationContext(), "Correct Answer!",
Toast.LENGTH_SHORT).show();
                }
                else {
                    //condition measures
                    numberRelation.setText(">");
                    numberRelation.setTextColor(Color.parseColor("#93c47d"));
                    Toast.makeText(getApplicationContext(), "Wrong Answer!",
Toast.LENGTH_SHORT).show();
                }

                buttonClicked = true;
                // Call showReplayBT method after 2 seconds
                (new Handler()).postDelayed(this::showReplayBT, 2000);
            }
        });

        backBT.setOnClickListener(e->{
            generalFunc.switchIntent(getApplicationContext(),MainActivity.class);
            overridePendingTransition(0, 0);
            finish();
        });
        getOnBackPressedDispatcher().addCallback(new OnBackPressedCallback(true) {
            @Override
            public void handleOnBackPressed() {
                generalFunc.switchIntent(getApplicationContext(),MainActivity.class);
                overridePendingTransition(0, 0);
                finish();
            }
        });
    }

    private void generateRandom() {
        randomNum1 = new Random().nextInt(1000);
        do {
            randomNum2 = new Random().nextInt(1000);
        }   while (randomNum1 == randomNum2);
    }

    private boolean isLarger(int num1, int num2) {
        return (num1 > num2);
    }

    private void showReplayBT() {
        if (buttonClicked) {
            restartBT.setOnClickListener(v -> {
                generalFunc.switchIntent(getApplicationContext(),
ComparingNumbersActivity.class);
                overridePendingTransition(0, 0);
                finish();
            });

            restartBT.setVisibility(View.VISIBLE);
        }
    }
}
```

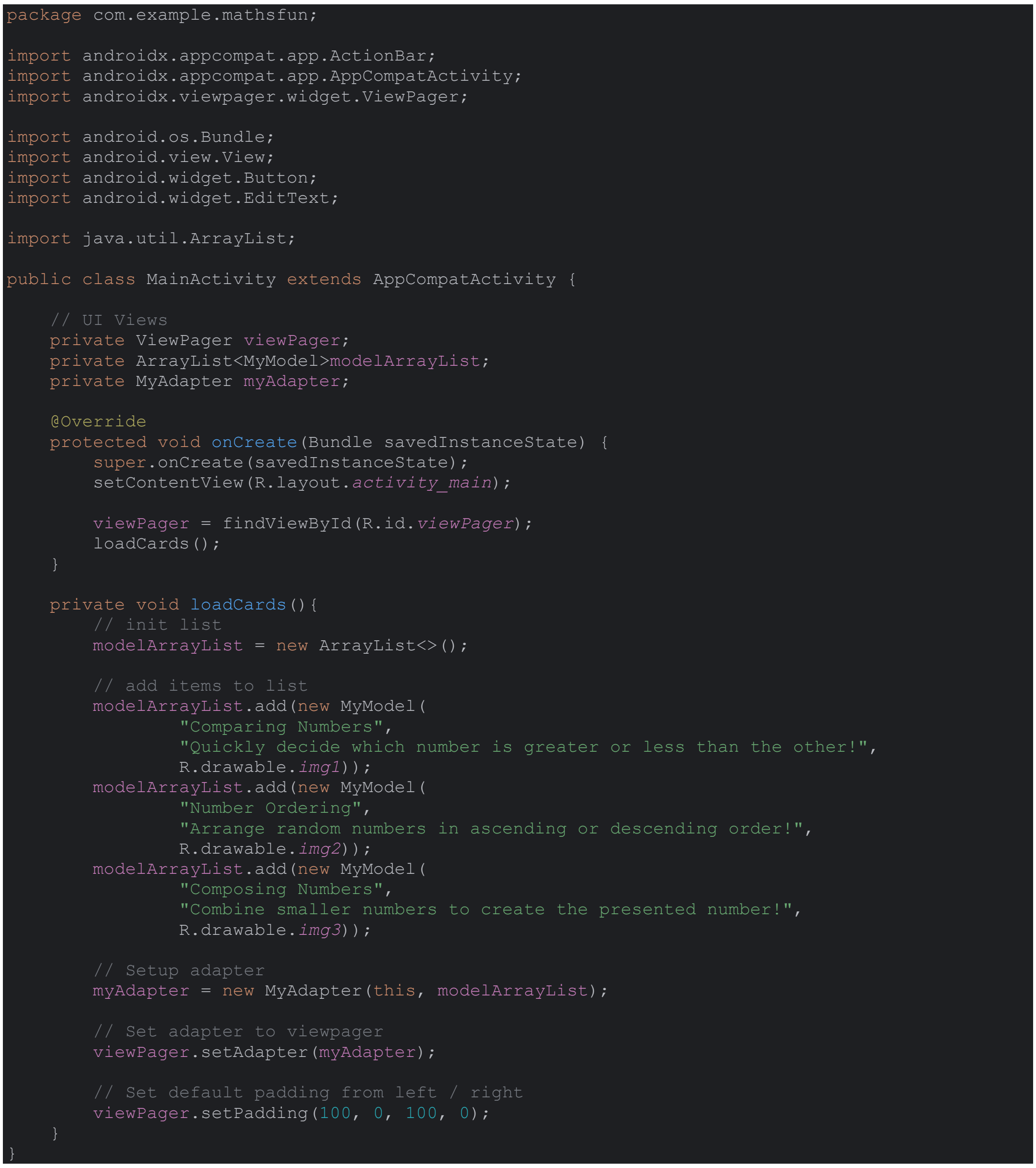
2. ComposingNumbersActivity.java

```
package com.example.mathsfun;

import android.os.Bundle;
import android.os.Handler;
import android.view.View;
import android.widget.EditText;
import android.widget.TextView;
import android.widget.Toast;

import androidx.activity.OnBackPressedCallback;
import androidx.appcompat.app.AppCompatActivity;

import com.example.mathsfun.general.GeneralFunctionClass;
import com.example.mathsfun.general.GeneralFunctionInterface;
import com.google.android.material.button.MaterialButton;

import java.util.ArrayList;
import java.util.Random;

public class ComposingNumbersActivity extends AppCompatActivity {

    MaterialButton btBack, btRestart, btSubmit;
    TextView targetNumberTv;
    EditText inputText_1, inputText_2;
    private int randomNum;
    GeneralFunctionInterface generalFunc= new GeneralFunctionClass();

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

        btBack = findViewById(R.id.backBT);
        btRestart = findViewById(R.id.restart_bt);
        btSubmit = findViewById(R.id.bt_submit);
        targetNumberTv = findViewById(R.id.targetNumberTV);
        inputText_1 = findViewById(R.id.inputText1);
        inputText_2 = findViewById(R.id.inputText2);

        btRestart.setVisibility(View.GONE);

        // Generate random number and set text to TextView targetNumberTv
        randomNum = new Random().nextInt(1000);
        targetNumberTv.setText(String.valueOf(randomNum));

        // Handle button events
        btSubmit.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                ArrayList<String> temp = new ArrayList<>();
                temp.add(inputText_1.getText().toString());
                temp.add(inputText_2.getText().toString());

                int num1, num2, targetNum;
                targetNum = Integer.parseInt(targetNumberTv.getText().toString());
                if (isInteger(temp.get(0))) {
                    num1 = Integer.parseInt(temp.get(0));
                    if (isInteger(temp.get(1))) {
                        num2 = Integer.parseInt(temp.get(1));

                        // Check if composition condition correct
                        if ((num1 + num2) == targetNum) { // If two input correctly makes up
targetNum
                            Toast.makeText(getApplicationContext(), "Correct Answer!",
Toast.LENGTH_SHORT).show();
                        } else
                            Toast.makeText(getApplicationContext(), "Wrong Answer!",
Toast.LENGTH_SHORT).show();

                    } else {
                        Toast.makeText(getApplicationContext(), "Your input does not match the
format of an integer number.\n" +
                                "Please retry again.", Toast.LENGTH_SHORT).show();
                    }
                } else Toast.makeText(getApplicationContext(), "Your input does not match the
format of an integer number.\n" +
                            "Please retry again.", Toast.LENGTH_SHORT).show();

                btSubmit.setEnabled(false);

                // Call showReplayBT method after 2 seconds
                (new Handler()).postDelayed(ComposingNumbersActivity.this::showReplayBT,
2000);
            }
        });
```



```java
        });

        btBack.setOnClickListener(e->{
            generalFunc.switchIntent(getApplicationContext(),MainActivity.class);
            overridePendingTransition(0, 0);
            finish();
        });
        getOnBackPressedDispatcher().addCallback(new OnBackPressedCallback(true) {
            @Override
            public void handleOnBackPressed() {
                generalFunc.switchIntent(getApplicationContext(),MainActivity.class);
                overridePendingTransition(0, 0);
                finish();
            }
        });

    }

    private boolean isInteger(String input) {
        try {
            Integer.parseInt(input);
            // Parsing succeeded, so the input is an integer
            return true;
        } catch (NumberFormatException e) {
            // Parsing failed, so the input is not an integer
            return false;
        }
    }

    private void showReplayBT() {
        btRestart.setOnClickListener(v -> {
            generalFunc.switchIntent(getApplicationContext(), ComposingNumbersActivity.class);
            overridePendingTransition(0, 0);
            finish();
        });

        btRestart.setVisibility(View.VISIBLE);
    }

}
```

3. MainActivity.java



```java
package com.example.mathsfun;

import androidx.appcompat.app.ActionBar;
import androidx.appcompat.app.AppCompatActivity;
import androidx.viewpager.widget.ViewPager;

import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;

import java.util.ArrayList;

public class MainActivity extends AppCompatActivity {

    // UI Views
    private ViewPager viewPager;
    private ArrayList<MyModel>modelArrayList;
    private MyAdapter myAdapter;

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

        viewPager = findViewById(R.id.viewPager);
        loadCards();
    }

    private void loadCards(){
        // init list
        modelArrayList = new ArrayList<>();

        // add items to list
        modelArrayList.add(new MyModel(
                "Comparing Numbers",
                "Quickly decide which number is greater or less than the other!",
                R.drawable.img1));
        modelArrayList.add(new MyModel(
                "Number Ordering",
                "Arrange random numbers in ascending or descending order!",
                R.drawable.img2));
        modelArrayList.add(new MyModel(
                "Composing Numbers",
                "Combine smaller numbers to create the presented number!",
                R.drawable.img3));

        // Setup adapter
        myAdapter = new MyAdapter(this, modelArrayList);

        // Set adapter to viewpager
        viewPager.setAdapter(myAdapter);

        // Set default padding from left / right
        viewPager.setPadding(100, 0, 100, 0);
    }
}
```

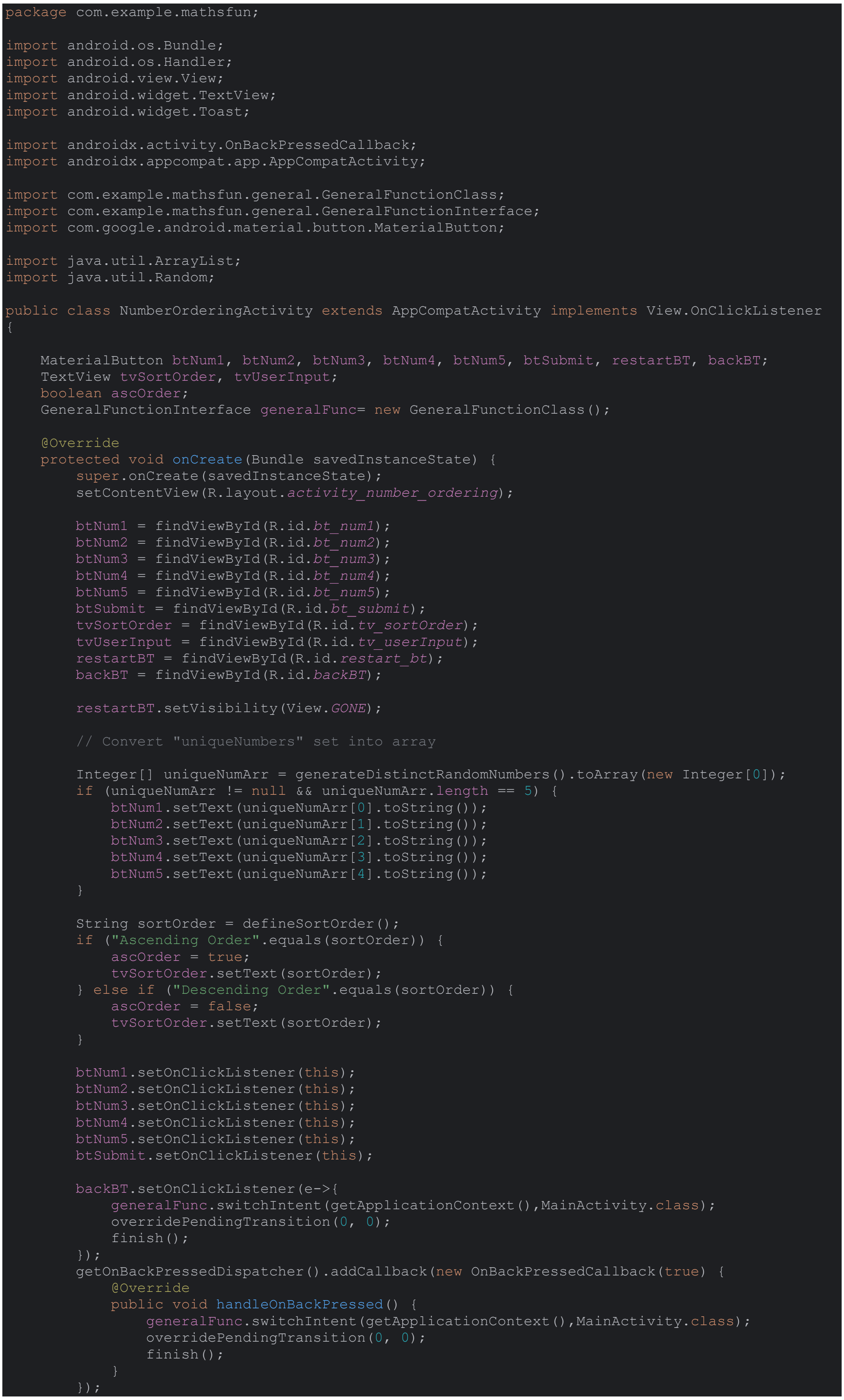
4. MyAdapter.java

```
package com.example.mathsfun;

import static androidx.core.content.ContextCompat.startActivity;

import android.content.Context;
import android.content.Intent;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.Button;
import android.widget.ImageView;
import android.widget.TextView;
import android.widget.Toast;

import androidx.annotation.NonNull;
import androidx.viewpager.widget.PagerAdapter;

import java.util.ArrayList;

public class MyAdapter extends PagerAdapter {
    private Context context;
    private ArrayList<MyModel> ModelArrayList;

    public MyAdapter(Context context, ArrayList<MyModel> ModelArrayList) {
        this.context = context;
        this.ModelArrayList = ModelArrayList;
    }

    @Override
    public int getCount() {
        return ModelArrayList.size(); // return list of records/items
    }

    @Override
    public boolean isViewFromObject(@NonNull View view, @NonNull Object object) {

        return view.equals(object);
    }

    @NonNull
    @Override
    public Object instantiateItem(@NonNull ViewGroup container, int position) {
        // inflate layout card_item.xml
        View view = LayoutInflater.from(context).inflate(R.layout.card_item, container,
false);

        // init uid views from card_item.xml
        ImageView bannerTV = view.findViewById(R.id.bannerTV);
        TextView titleTV = view.findViewById(R.id.titleTV);
        TextView descriptionTV = view.findViewById(R.id.descTV);
        Button playBT = view.findViewById(R.id.playBT);

        // get data
        MyModel model = ModelArrayList.get(position);
        final String title = model.getTitle();
        final String description = model.getDescription();
        int image = model.getImage();

        // Set data to UI views
        bannerTV.setImageResource(image);
        titleTV.setText(title);
        descriptionTV.setText(description);

        // Handle card button click
        playBT.setOnClickListener(new View.OnClickListener()
        {
            @Override
            public void onClick(View view) {
                if(position == 0){
                    // Toast.makeText(context,"Playing 'Comparing
Numbers'...",Toast.LENGTH_SHORT).show();
                    Intent intent = new Intent(context, ComparingNumbersActivity.class);
                    context.startActivity(intent);
                }
                else if (position == 1) {
                    // Toast.makeText(context,"Playing 'Number
Ordering'...",Toast.LENGTH_SHORT).show();
                    Intent intent = new Intent(context, NumberOrderingActivity.class);
                    context.startActivity(intent);
                }
                else {
                    // Toast.makeText(context,"Playing 'Composing
Numbers'...",Toast.LENGTH_SHORT).show();
                    Intent intent = new Intent(context, ComposingNumbersActivity.class);
                    context.startActivity(intent);
                }
            }
        });

        // Add view to container
        container.addView(view, position);

        return view;
    }

    @Override
    public void destroyItem(@NonNull ViewGroup container, int position, @NonNull Object
object) {
        // Remove the view from the container
        container.removeView((View) object);
    }
}
```

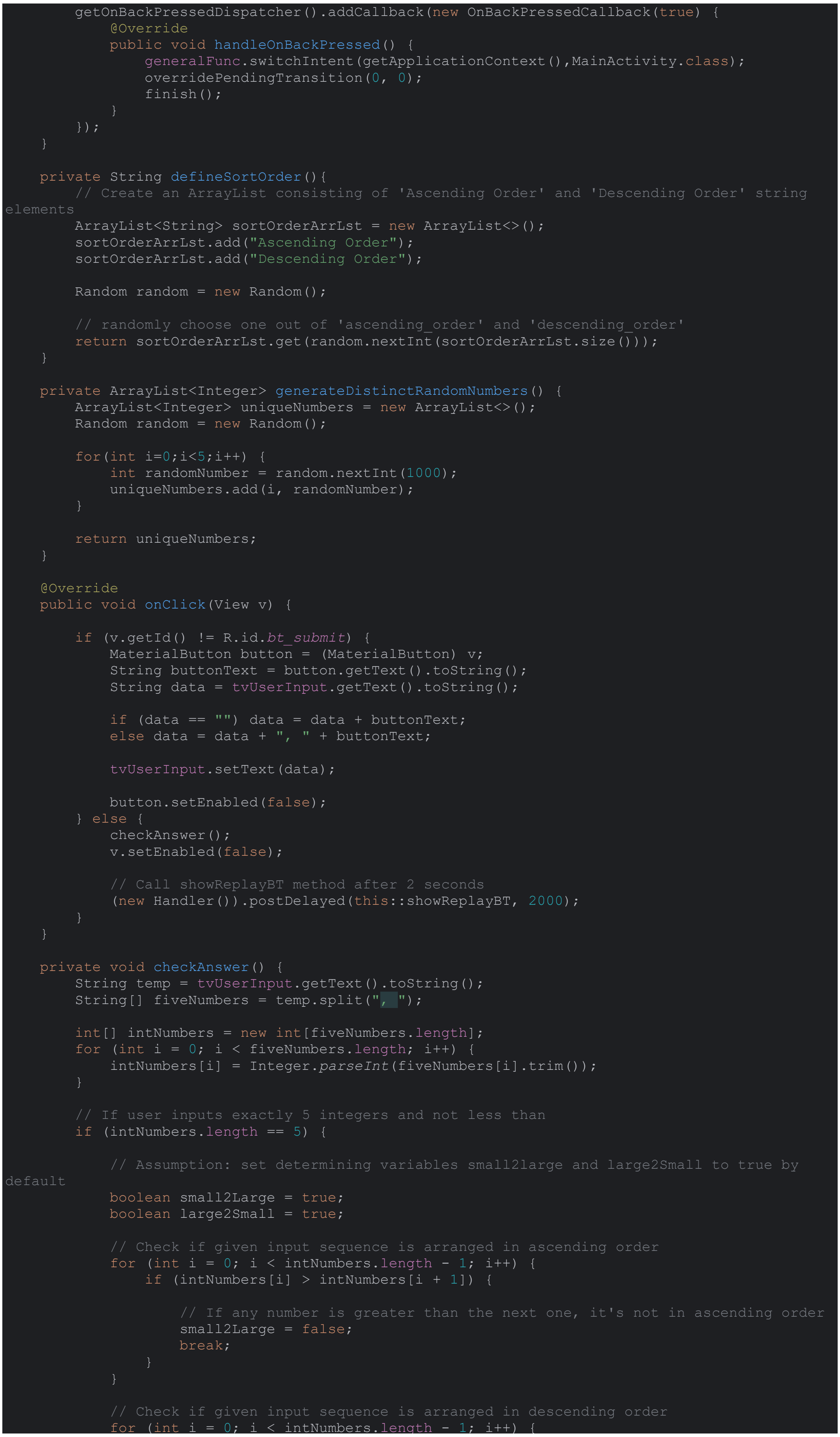
5. MyModel.java

```
package com.example.mathsfun;

import android.widget.Button;

public class MyModel {

    String title, description;
    int image;
    Button button;

    public MyModel(String title, String description, int image) {
        this.title = title;
        this.description = description;
        this.image = image;
    }

    public String getTitle() {
        return title;
    }

    public void setTitle(String title) {
        this.title = title;
    }

    public String getDescription() {
        return description;
    }

    public void setDescription(String description) {
        this.description = description;
    }

    public int getImage() {
        return image;
    }

    public void setImage(int image) {
        this.image = image;
    }
}
```

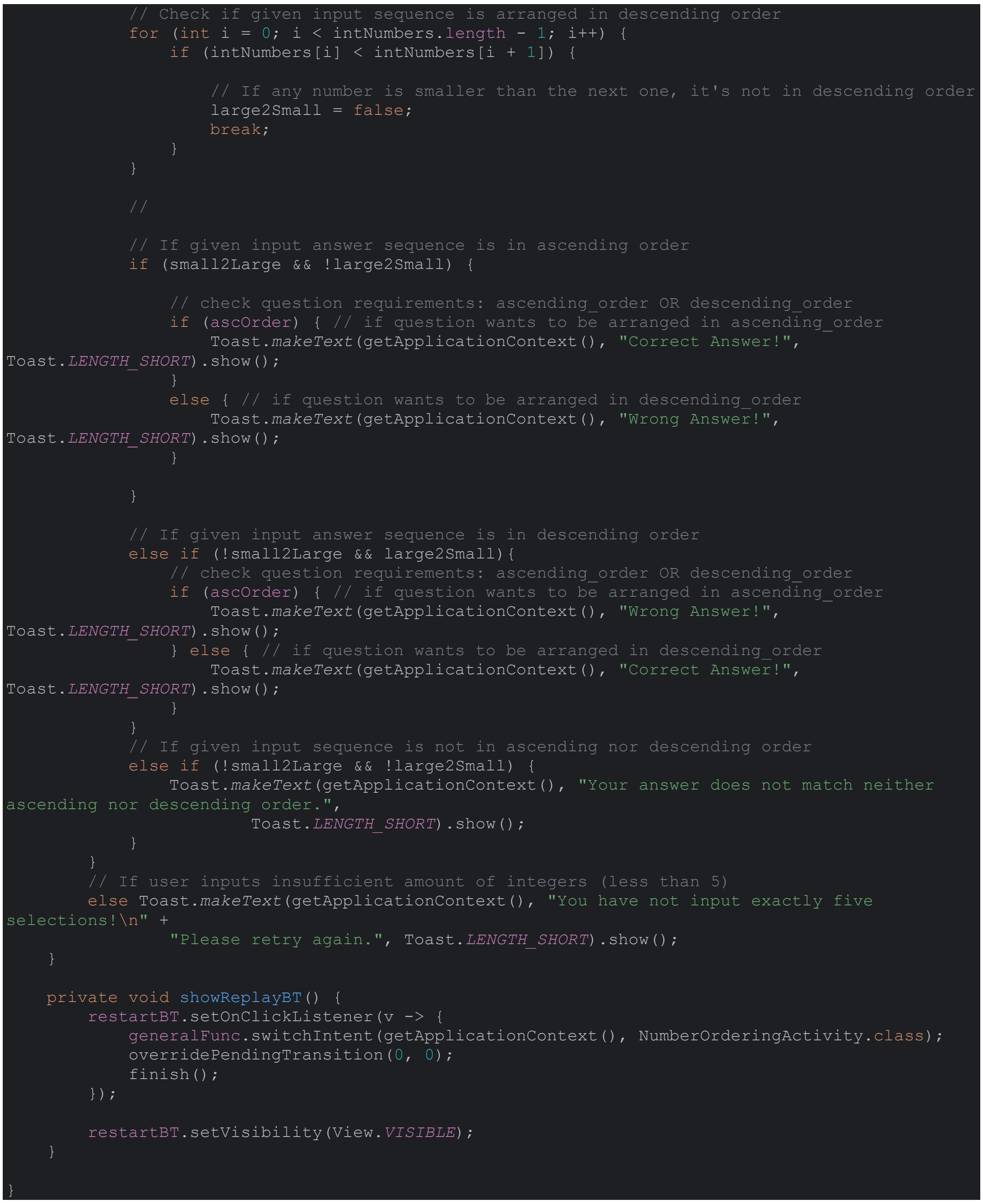
6. NumberOrderingActivity.java



```java
package com.example.mathsfun;

import android.os.Bundle;
import android.os.Handler;
import android.view.View;
import android.widget.TextView;
import android.widget.Toast;

import androidx.activity.OnBackPressedCallback;
import androidx.appcompat.app.AppCompatActivity;

import com.example.mathsfun.general.GeneralFunctionClass;
import com.example.mathsfun.general.GeneralFunctionInterface;
import com.google.android.material.button.MaterialButton;

import java.util.ArrayList;
import java.util.Random;

public class NumberOrderingActivity extends AppCompatActivity implements View.OnClickListener
{

    MaterialButton btNum1, btNum2, btNum3, btNum4, btNum5, btSubmit, restartBT, backBT;
    TextView tvSortOrder, tvUserInput;
    boolean ascOrder;
    GeneralFunctionInterface generalFunc= new GeneralFunctionClass();

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

        btNum1 = findViewById(R.id.bt_num1);
        btNum2 = findViewById(R.id.bt_num2);
        btNum3 = findViewById(R.id.bt_num3);
        btNum4 = findViewById(R.id.bt_num4);
        btNum5 = findViewById(R.id.bt_num5);
        btSubmit = findViewById(R.id.bt_submit);
        tvSortOrder = findViewById(R.id.tv_sortOrder);
        tvUserInput = findViewById(R.id.tv_userInput);
        restartBT = findViewById(R.id.restart_bt);
        backBT = findViewById(R.id.backBT);

        restartBT.setVisibility(View.GONE);

        // Convert "uniqueNumbers" set into array

        Integer[] uniqueNumArr = generateDistinctRandomNumbers().toArray(new Integer[0]);
        if (uniqueNumArr != null && uniqueNumArr.length == 5) {
            btNum1.setText(uniqueNumArr[0].toString());
            btNum2.setText(uniqueNumArr[1].toString());
            btNum3.setText(uniqueNumArr[2].toString());
            btNum4.setText(uniqueNumArr[3].toString());
            btNum5.setText(uniqueNumArr[4].toString());
        }

        String sortOrder = defineSortOrder();
        if ("Ascending Order".equals(sortOrder)) {
            ascOrder = true;
            tvSortOrder.setText(sortOrder);
        } else if ("Descending Order".equals(sortOrder)) {
            ascOrder = false;
            tvSortOrder.setText(sortOrder);
        }

        btNum1.setOnClickListener(this);
        btNum2.setOnClickListener(this);
        btNum3.setOnClickListener(this);
        btNum4.setOnClickListener(this);
        btNum5.setOnClickListener(this);
        btSubmit.setOnClickListener(this);

        backBT.setOnClickListener(e->{
            generalFunc.switchIntent(getApplicationContext(),MainActivity.class);
            overridePendingTransition(0, 0);
            finish();
        });
        getOnBackPressedDispatcher().addCallback(new OnBackPressedCallback(true) {
            @Override
            public void handleOnBackPressed() {
                generalFunc.switchIntent(getApplicationContext(),MainActivity.class);
                overridePendingTransition(0, 0);
                finish();
            }
        });
```

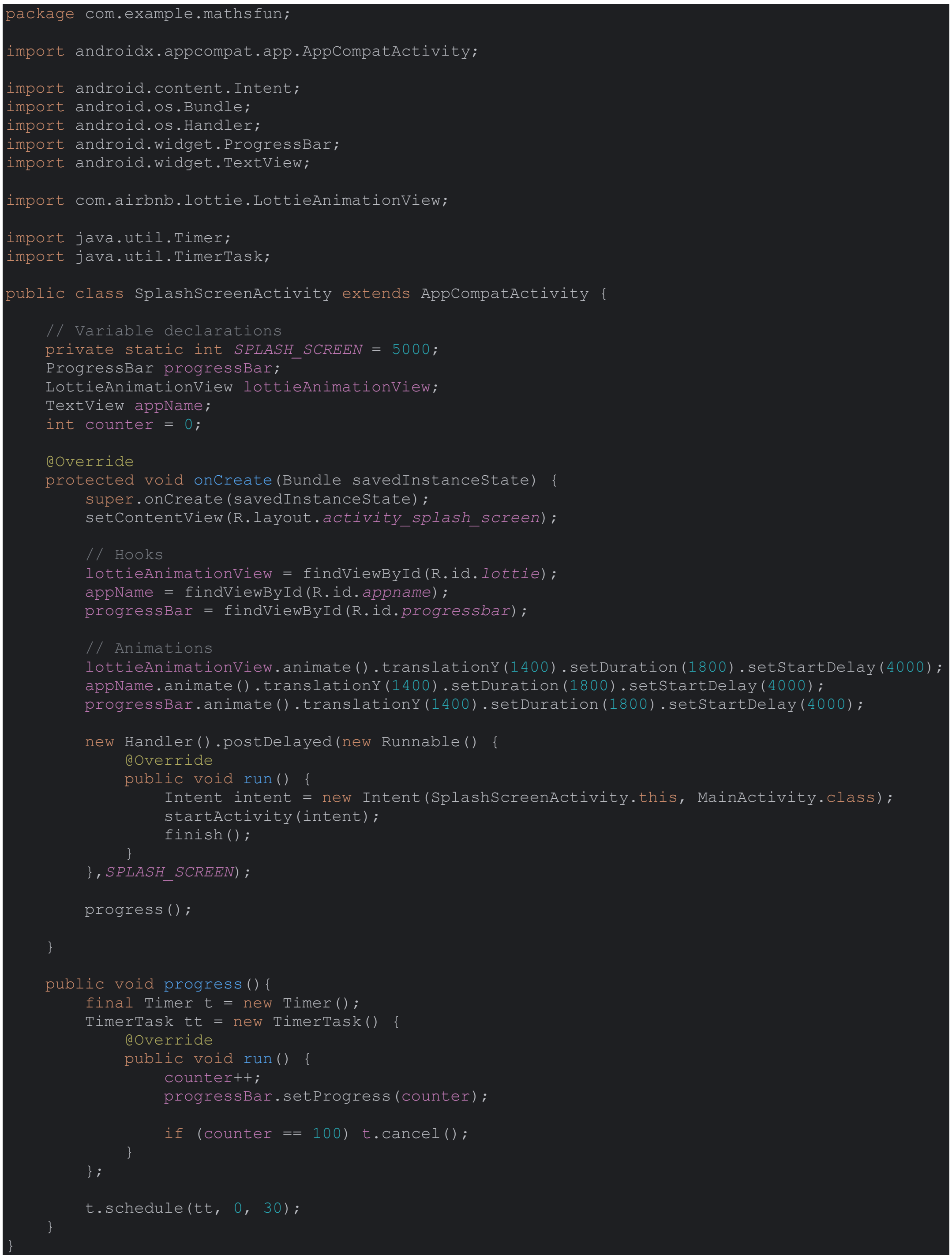


```
        getOnBackPressedDispatcher().addCallback(new OnBackPressedCallback(true) {
            @Override
            public void handleOnBackPressed() {
                generalFunc.switchIntent(getApplicationContext(),MainActivity.class);
                overridePendingTransition(0, 0);
                finish();
            }
        });
    }

    private String defineSortOrder(){
        // Create an ArrayList consisting of 'Ascending Order' and 'Descending Order' string
elements
        ArrayList<String> sortOrderArrLst = new ArrayList<>();
        sortOrderArrLst.add("Ascending Order");
        sortOrderArrLst.add("Descending Order");

        Random random = new Random();

        // randomly choose one out of 'ascending_order' and 'descending_order'
        return sortOrderArrLst.get(random.nextInt(sortOrderArrLst.size()));
    }

    private ArrayList<Integer> generateDistinctRandomNumbers() {
        ArrayList<Integer> uniqueNumbers = new ArrayList<>();
        Random random = new Random();

        for(int i=0;i<5;i++) {
            int randomNumber = random.nextInt(1000);
            uniqueNumbers.add(i, randomNumber);
        }

        return uniqueNumbers;
    }

    @Override
    public void onClick(View v) {

        if (v.getId() != R.id.bt_submit) {
            MaterialButton button = (MaterialButton) v;
            String buttonText = button.getText().toString();
            String data = tvUserInput.getText().toString();

            if (data == "") data = data + buttonText;
            else data = data + ", " + buttonText;

            tvUserInput.setText(data);

            button.setEnabled(false);
        } else {
            checkAnswer();
            v.setEnabled(false);

            // Call showReplayBT method after 2 seconds
            (new Handler()).postDelayed(this::showReplayBT, 2000);
        }
    }

    private void checkAnswer() {
        String temp = tvUserInput.getText().toString();
        String[] fiveNumbers = temp.split(", ");

        int[] intNumbers = new int[fiveNumbers.length];
        for (int i = 0; i < fiveNumbers.length; i++) {
            intNumbers[i] = Integer.parseInt(fiveNumbers[i].trim());
        }

        // If user inputs exactly 5 integers and not less than
        if (intNumbers.length == 5) {

            // Assumption: set determining variables small2large and large2Small to true by
default
            boolean small2Large = true;
            boolean large2Small = true;

            // Check if given input sequence is arranged in ascending order
            for (int i = 0; i < intNumbers.length - 1; i++) {
                if (intNumbers[i] > intNumbers[i + 1]) {

                    // If any number is greater than the next one, it's not in ascending order
                    small2Large = false;
                    break;
                }
            }

            // Check if given input sequence is arranged in descending order
            for (int i = 0; i < intNumbers.length - 1; i++) {
```



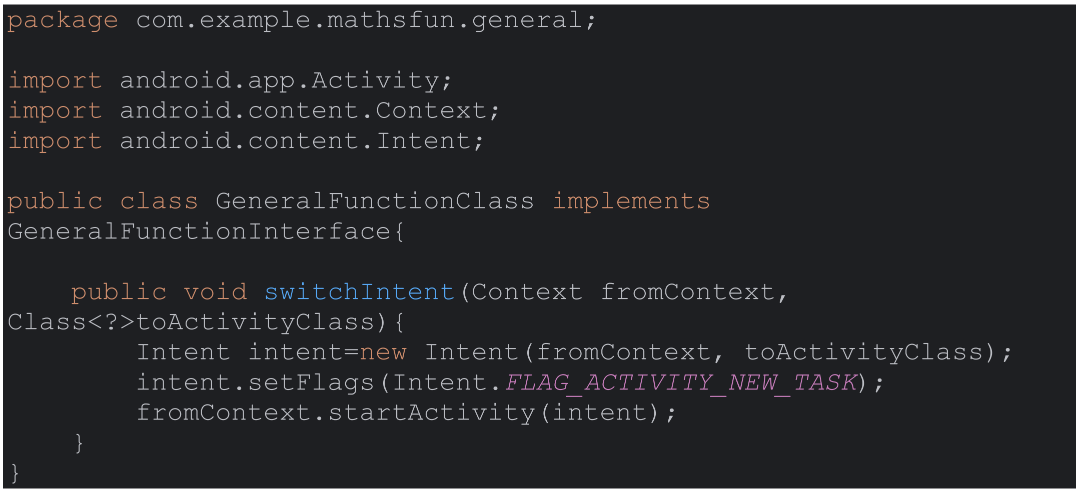
```
            // Check if given input sequence is arranged in descending order
            for (int i = 0; i < intNumbers.length - 1; i++) {
                if (intNumbers[i] < intNumbers[i + 1]) {

                    // If any number is smaller than the next one, it's not in descending order
                    large2Small = false;
                    break;
                }
            }

            //

            // If given input answer sequence is in ascending order
            if (small2Large && !large2Small) {

                // check question requirements: ascending_order OR descending_order
                if (ascOrder) { // if question wants to be arranged in ascending_order
                    Toast.makeText(getApplicationContext(), "Correct Answer!",
Toast.LENGTH_SHORT).show();
                }
                else { // if question wants to be arranged in descending_order
                    Toast.makeText(getApplicationContext(), "Wrong Answer!",
Toast.LENGTH_SHORT).show();
                }

            }

            // If given input answer sequence is in descending order
            else if (!small2Large && large2Small){
                // check question requirements: ascending_order OR descending_order
                if (ascOrder) { // if question wants to be arranged in ascending_order
                    Toast.makeText(getApplicationContext(), "Wrong Answer!",
Toast.LENGTH_SHORT).show();
                } else { // if question wants to be arranged in descending_order
                    Toast.makeText(getApplicationContext(), "Correct Answer!",
Toast.LENGTH_SHORT).show();
                }
            }
            // If given input sequence is not in ascending nor descending order
            else if (!small2Large && !large2Small) {
                Toast.makeText(getApplicationContext(), "Your answer does not match neither
ascending nor descending order.",
                        Toast.LENGTH_SHORT).show();
            }
        }
        // If user inputs insufficient amount of integers (less than 5)
        else Toast.makeText(getApplicationContext(), "You have not input exactly five
selections!\n" +
                "Please retry again.", Toast.LENGTH_SHORT).show();
    }

    private void showReplayBT() {
        restartBT.setOnClickListener(v -> {
            generalFunc.switchIntent(getApplicationContext(), NumberOrderingActivity.class);
            overridePendingTransition(0, 0);
            finish();
        });

        restartBT.setVisibility(View.VISIBLE);
    }

}
```

7. SplashScreenActivity.java



```java
package com.example.mathsfun;

import androidx.appcompat.app.AppCompatActivity;

import android.content.Intent;
import android.os.Bundle;
import android.os.Handler;
import android.widget.ProgressBar;
import android.widget.TextView;

import com.airbnb.lottie.LottieAnimationView;

import java.util.Timer;
import java.util.TimerTask;

public class SplashScreenActivity extends AppCompatActivity {

    // Variable declarations
    private static int SPLASH_SCREEN = 5000;
    ProgressBar progressBar;
    LottieAnimationView lottieAnimationView;
    TextView appName;
    int counter = 0;

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

        // Hooks
        lottieAnimationView = findViewById(R.id.lottie);
        appName = findViewById(R.id.appname);
        progressBar = findViewById(R.id.progressbar);

        // Animations
        lottieAnimationView.animate().translationY(1400).setDuration(1800).setStartDelay(4000);
        appName.animate().translationY(1400).setDuration(1800).setStartDelay(4000);
        progressBar.animate().translationY(1400).setDuration(1800).setStartDelay(4000);

        new Handler().postDelayed(new Runnable() {
            @Override
            public void run() {
                Intent intent = new Intent(SplashScreenActivity.this, MainActivity.class);
                startActivity(intent);
                finish();
            }
        },SPLASH_SCREEN);

        progress();

    }

    public void progress(){
        final Timer t = new Timer();
        TimerTask tt = new TimerTask() {
            @Override
            public void run() {
                counter++;
                progressBar.setProgress(counter);

                if (counter == 100) t.cancel();
            }
        };

        t.schedule(tt, 0, 30);
    }
}
```

8. GeneralFunctionClass.java



```java
package com.example.mathsfun.general;

import android.app.Activity;
import android.content.Context;
import android.content.Intent;

public class GeneralFunctionClass implements
GeneralFunctionInterface{

    public void switchIntent(Context fromContext,
Class<?>toActivityClass){
        Intent intent=new Intent(fromContext, toActivityClass);
        intent.setFlags(Intent.FLAG_ACTIVITY_NEW_TASK);
        fromContext.startActivity(intent);
    }
}
```

9. GeneralFunctionInterface.java

```java
package com.example.mathsfun.general;

import android.content.Context;

public interface GeneralFunctionInterface {
    public void switchIntent(Context context, Class<?>activity);
}
```