Mobile Programming - Lab1

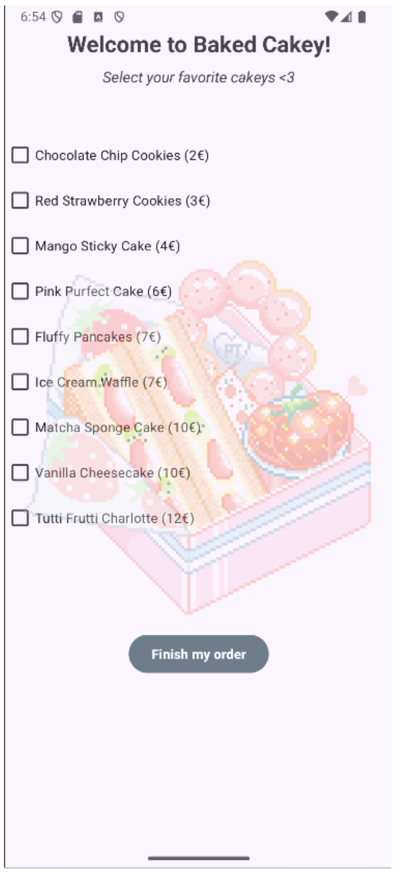
Joanne ANDRIAMAHANDRY - ID3A

Online Food Ordering System

Create the online food ordering system for the restaurant. Basic functions include selecting food, fill the delivery details, and button "Order" gives the total price, button "Back to Main" to intent to next activities.

For this project, I decided to add a theme to the ordering system and make it pink and girly, allowing user to order many cakes and snacks.

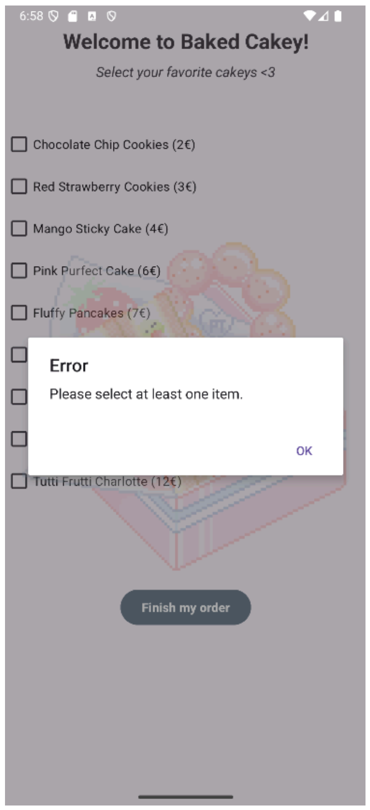
Visuals



Welcome to Baked Cakey!

Select your favorite cakeys <3

- ☐ Chocolate Chip Cookies (2€)
- ☐ Red Strawberry Cookies (3€)
- ☐ Mango Sticky Cake (4€)
- ☐ Pink Purfect Cake (6€)
- ☐ Fluffy Pancakes (7€)
- ☐ Ice Cream Waffle (7€)
- ☐ Matcha Sponge Cake (10€)
- ☐ Vanilla Cheesecake (10€)
- ☐ Tutti Frutti Charlotte (12€)



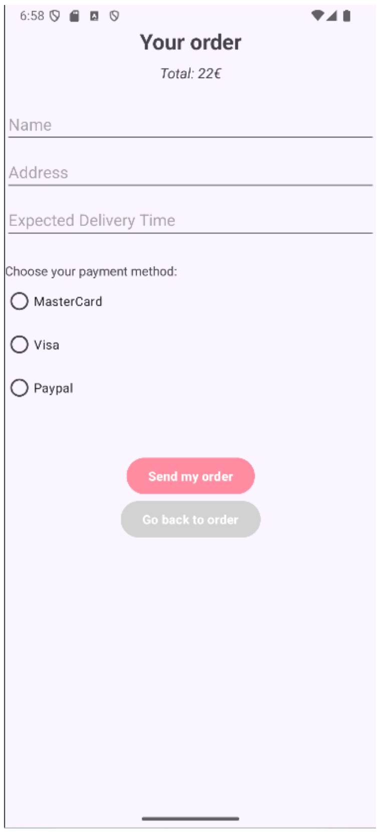
Welcome to Baked Cakey!

Select your favorite cakeys <3

- ☐ Chocolate Chip Cookies (2€)
- ☐ Red Strawberry Cookies (3€)
- ☐ Mango Sticky Cake (4€)
- ☐ Pink Purfect Cake (6€)
- ☐ Fluffy Pancakes (7€)
- ☐ Tutti Frutti Charlotte (12€)

Error

Please select at least one item.



Your order

Total: 22€

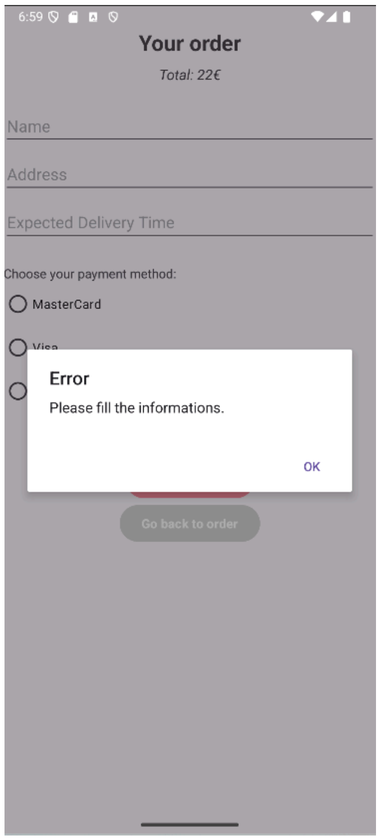
Name

Address

Expected Delivery Time

Choose your payment method:

- ○ MasterCard
- ○ Visa
- ○ Paypal


Your order

Total: 22€

Name

Address

Expected Delivery Time

Choose your payment method:

MasterCard

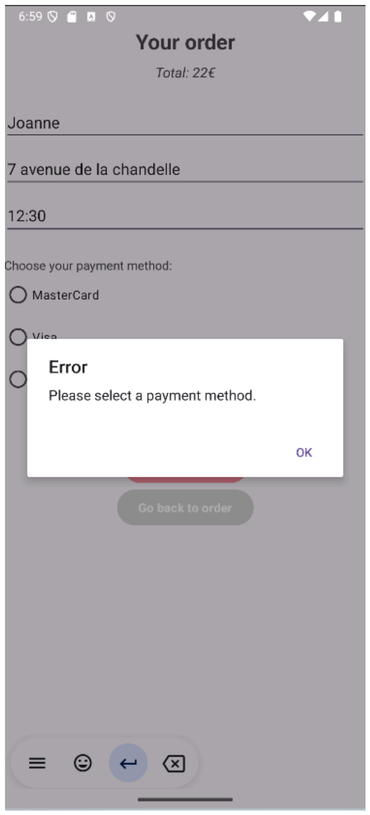
Visa

Error

Please fill the informations.

OK

Go back to order


Your order

Total: 22€

Joanne

7 avenue de la chandelle

12:30

Choose your payment method:

MasterCard

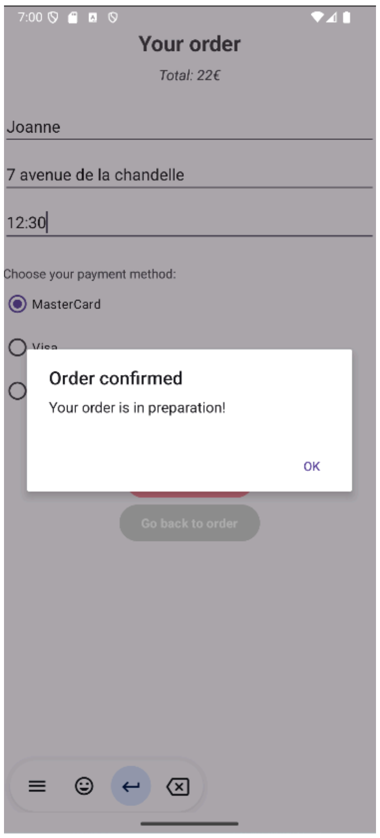
Visa

Error

Please select a payment method.

OK

Go back to order


Your order

Total: 22€

Joanne

7 avenue de la chandelle

12:30

Choose your payment method:

MasterCard

Visa

Order confirmed

Your order is in preparation!

OK

Go back to order

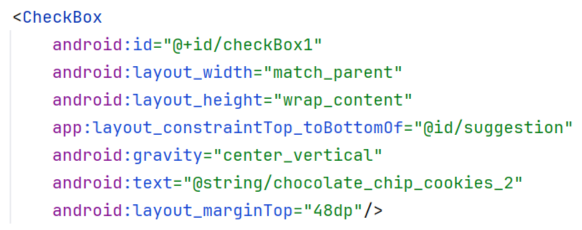
XML Code (Activity 1)

In this section, I will share every component I used, without repetition for checkboxes, radio buttons, and titles (I'll show them once each).

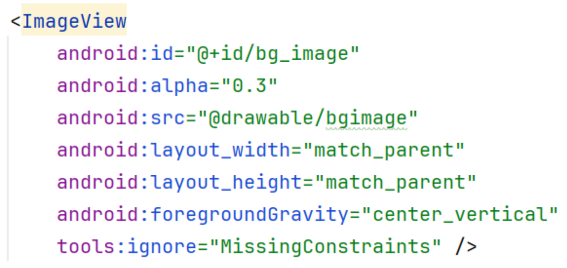
```
<TextView
    android:id="@+id/title"
    android:layout_width="0dp"
    android:layout_height="wrap_content"
    android:text="@string/welcome_to_baked_cakey"
    android:textSize="24sp"
    android:textStyle="bold"
    android:gravity="center"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintEnd_toEndOf="parent"
    android:layout_marginTop="50dp"
    tools:ignore="MissingConstraints" />
```

```
<TextView
    android:id="@+id/suggestion"
    android:layout_width="0dp"
    android:layout_height="wrap_content"
    android:text="@string/select_your_favorite_cakeys_3"
    android:textSize="16sp"
    android:textStyle="italic"
    android:gravity="center"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintTop_toBottomOf="@id/title"
    android:layout_marginTop="8dp"/>
```

Here are the TextViews I used to create the titles and subtitles in my activities. I stored the texts in the values file.

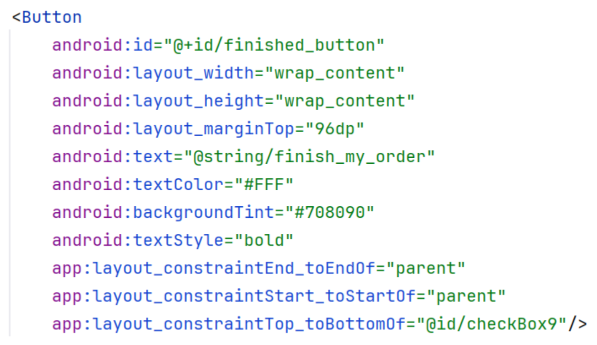


```
<CheckBox
    android:id="@+id/checkBox1"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    app:layout_constraintTop_toBottomOf="@id/suggestion"
    android:gravity="center_vertical"
    android:text="@string/chocolate_chip_cookies_2"
    android:layout_marginTop="48dp"/>
```



```
<ImageView
    android:id="@+id/bg_image"
    android:alpha="0.3"
    android:src="@drawable/bgimage"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:foregroundGravity="center_vertical"
    tools:ignore="MissingConstraints" />
```

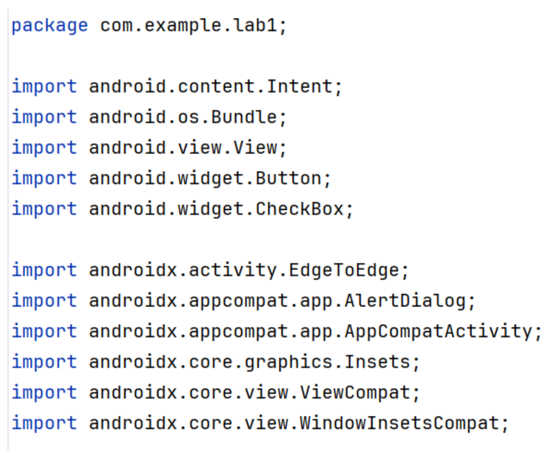
I created 9 checkboxes and an ImageView as the background image. I stored the texts in the values file.



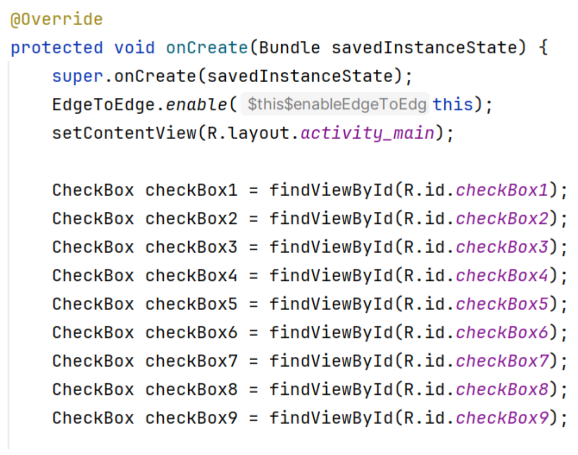
```
<Button
    android:id="@+id/finished_button"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginTop="96dp"
    android:text="@string/finish_my_order"
    android:textColor="#FFF"
    android:backgroundTint="#708090"
    android:textStyle="bold"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toBottomOf="@id/checkBox9"/>
```

Then I created a button that leads to the second activity, meaning the order is finished.

Java Code (Activity 1)



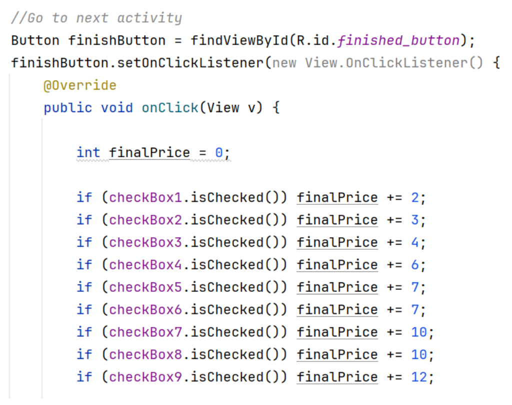
```java
package com.example.lab1;

import android.content.Intent;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.CheckBox;

import androidx.activity.EdgeToEdge;
import androidx.appcompat.app.AlertDialog;
import androidx.appcompat.app.AppCompatActivity;
import androidx.core.graphics.Insets;
import androidx.core.view.ViewCompat;
import androidx.core.view.WindowInsetsCompat;
```



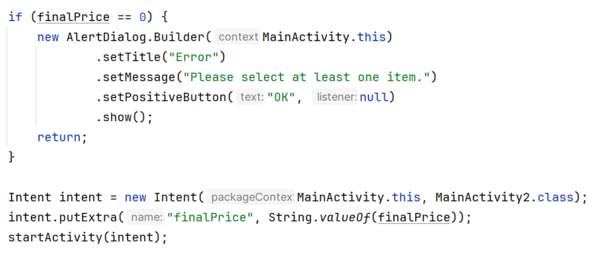
```java
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    EdgeToEdge.enable( $this$enableEdgeToEdg this);
    setContentView(R.layout.activity_main);

    CheckBox checkBox1 = findViewById(R.id.checkBox1);
    CheckBox checkBox2 = findViewById(R.id.checkBox2);
    CheckBox checkBox3 = findViewById(R.id.checkBox3);
    CheckBox checkBox4 = findViewById(R.id.checkBox4);
    CheckBox checkBox5 = findViewById(R.id.checkBox5);
    CheckBox checkBox6 = findViewById(R.id.checkBox6);
    CheckBox checkBox7 = findViewById(R.id.checkBox7);
    CheckBox checkBox8 = findViewById(R.id.checkBox8);
    CheckBox checkBox9 = findViewById(R.id.checkBox9);
```

For the Java code, I made many imports that will be useful for later use.

Then in the onCreate() method, I retrieved all XML elements using their IDs; I did it for all checkboxes.



```java
//Go to next activity
Button finishButton = findViewById(R.id.finished_button);
finishButton.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {

        int finalPrice = 0;

        if (checkBox1.isChecked()) finalPrice += 2;
        if (checkBox2.isChecked()) finalPrice += 3;
        if (checkBox3.isChecked()) finalPrice += 4;
        if (checkBox4.isChecked()) finalPrice += 6;
        if (checkBox5.isChecked()) finalPrice += 7;
        if (checkBox6.isChecked()) finalPrice += 7;
        if (checkBox7.isChecked()) finalPrice += 10;
        if (checkBox8.isChecked()) finalPrice += 10;
        if (checkBox9.isChecked()) finalPrice += 12;
```



```java
if (finalPrice == 0) {
    new AlertDialog.Builder( context MainActivity.this)
            .setTitle("Error")
            .setMessage("Please select at least one item.")
            .setPositiveButton( text: "OK",  listener: null)
            .show();
    return;
}

Intent intent = new Intent( packageContex MainActivity.this, MainActivity2.class);
intent.putExtra( name: "finalPrice", String.valueOf(finalPrice));
startActivity(intent);
```

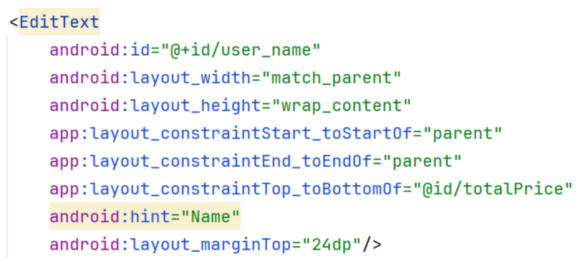
Now I retrieved the XML finish button and added an event listener to it: when the button is clicked, it will check if any checkbox has been checked and it will add the corresponding price to the final price.

Then it will check if the final price is 0, this means no checkbox has been checked and will ask the user to check at least one.

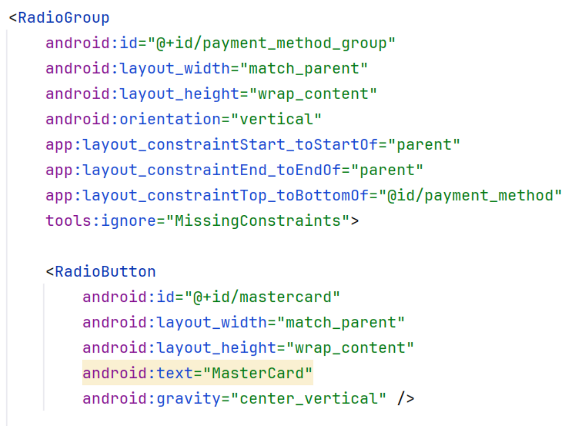
If the final price is different from 0, it will open the new activity, storing the finalPrice data to receive and use it in the activity 2.

XML Code (Activity 2)

I used the same system as activity 1 for title and subtitle, with a slight difference for subtitle text content that I will explain in the Java Code.



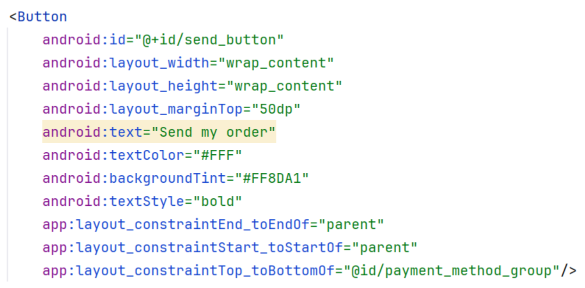
```
<EditText
    android:id="@+id/user_name"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintTop_toBottomOf="@id/totalPrice"
    android:hint="Name"
    android:layout_marginTop="24dp"/>
```



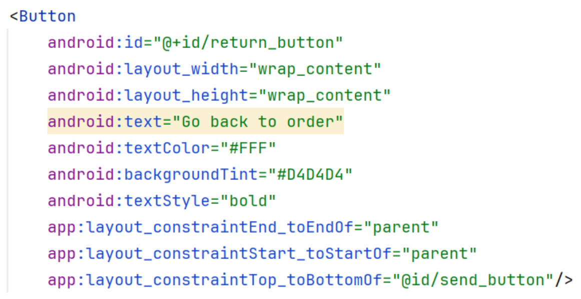
```
<RadioGroup
    android:id="@+id/payment_method_group"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:orientation="vertical"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintTop_toBottomOf="@id/payment_method"
    tools:ignore="MissingConstraints">

    <RadioButton
        android:id="@+id/mastercard"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="MasterCard"
        android:gravity="center_vertical" />
```

I created 3 EditTexts for user name, address and delivery time.

Then I created a RadioGroup with RadioButtons for each payment method.



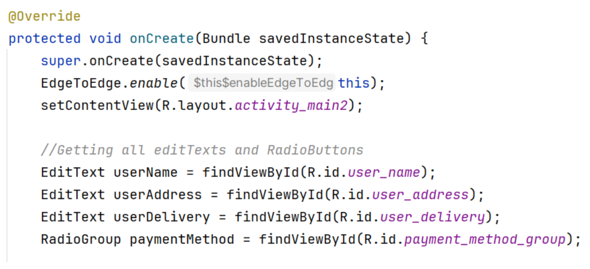
```
<Button
    android:id="@+id/send_button"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginTop="50dp"
    android:text="Send my order"
    android:textColor="#FFF"
    android:backgroundTint="#FF8DA1"
    android:textStyle="bold"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toBottomOf="@id/payment_method_group"/>
```



```
<Button
    android:id="@+id/return_button"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Go back to order"
    android:textColor="#FFF"
    android:backgroundTint="#D4D4D4"
    android:textStyle="bold"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toBottomOf="@id/send_button"/>
```

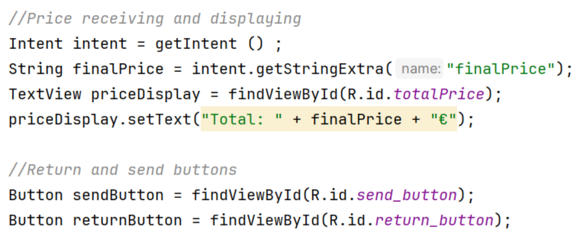
Finally, I created 2 buttons: one that you can use to go back to previous activity, and another one to confirm and send your order.

Java Code (Activity 2)



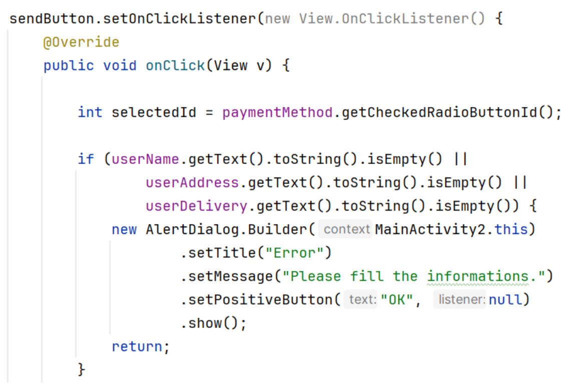
```java
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    EdgeToEdge.enable( $this$enableEdgeToEdg this);
    setContentView(R.layout.activity_main2);

    //Getting all editTexts and RadioButtons
    EditText userName = findViewById(R.id.user_name);
    EditText userAddress = findViewById(R.id.user_address);
    EditText userDelivery = findViewById(R.id.user_delivery);
    RadioGroup paymentMethod = findViewById(R.id.payment_method_group);
```

Here I retrieved all XML elements to the Java Code for later use.

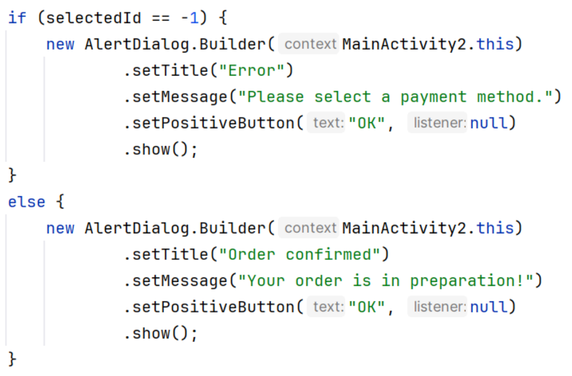


```java
//Price receiving and displaying
Intent intent = getIntent () ;
String finalPrice = intent.getStringExtra( name: "finalPrice");
TextView priceDisplay = findViewById(R.id.totalPrice);
priceDisplay.setText("Total: " + finalPrice + "€");

//Return and send buttons
Button sendButton = findViewById(R.id.send_button);
Button returnButton = findViewById(R.id.return_button);
```

Then I received the price data from the previous activity to display it in the totalPrice XML element. I also retrieved the XML buttons to the Java Code.

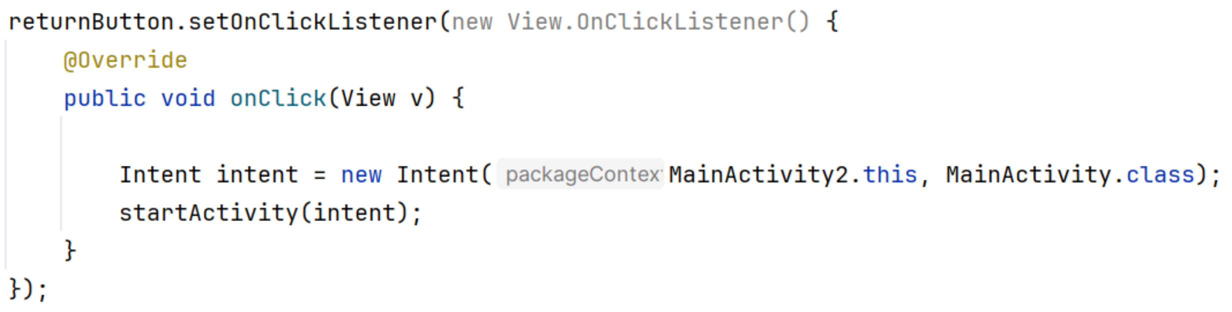


```java
sendButton.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {

        int selectedId = paymentMethod.getCheckedRadioButtonId();

        if (userName.getText().toString().isEmpty() ||
                userAddress.getText().toString().isEmpty() ||
                userDelivery.getText().toString().isEmpty()) {
            new AlertDialog.Builder( context MainActivity2.this)
                    .setTitle("Error")
                    .setMessage("Please fill the informations.")
                    .setPositiveButton( text: "OK",  listener: null)
                    .show();
            return;
        }
```



```java
if (selectedId == -1) {
    new AlertDialog.Builder( context MainActivity2.this)
            .setTitle("Error")
            .setMessage("Please select a payment method.")
            .setPositiveButton( text: "OK",  listener: null)
            .show();
}
else {
    new AlertDialog.Builder( context MainActivity2.this)
            .setTitle("Order confirmed")
            .setMessage("Your order is in preparation!")
            .setPositiveButton( text: "OK",  listener: null)
            .show();
}
```

For the send button, I made sure every text input was filled, and that a payment method has been checked, otherwise it will display an error.



```java
returnButton.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {

        Intent intent = new Intent( packageContex MainActivity2.this, MainActivity.class);
        startActivity(intent);
    }
});
```

Finally, I made the return button able to go back to previous activity.