UNIVERSITI MALAYSIATERENGGANU

Faculty of Computer Sciences and Mathematics

CSM3123-Native Mobile Programming

Lab report 1

Prepared By :

Muhammad Harith bin Zulkifli (S67335)

Prepared For :

Dr Rabiei Mamat

25 October 2024

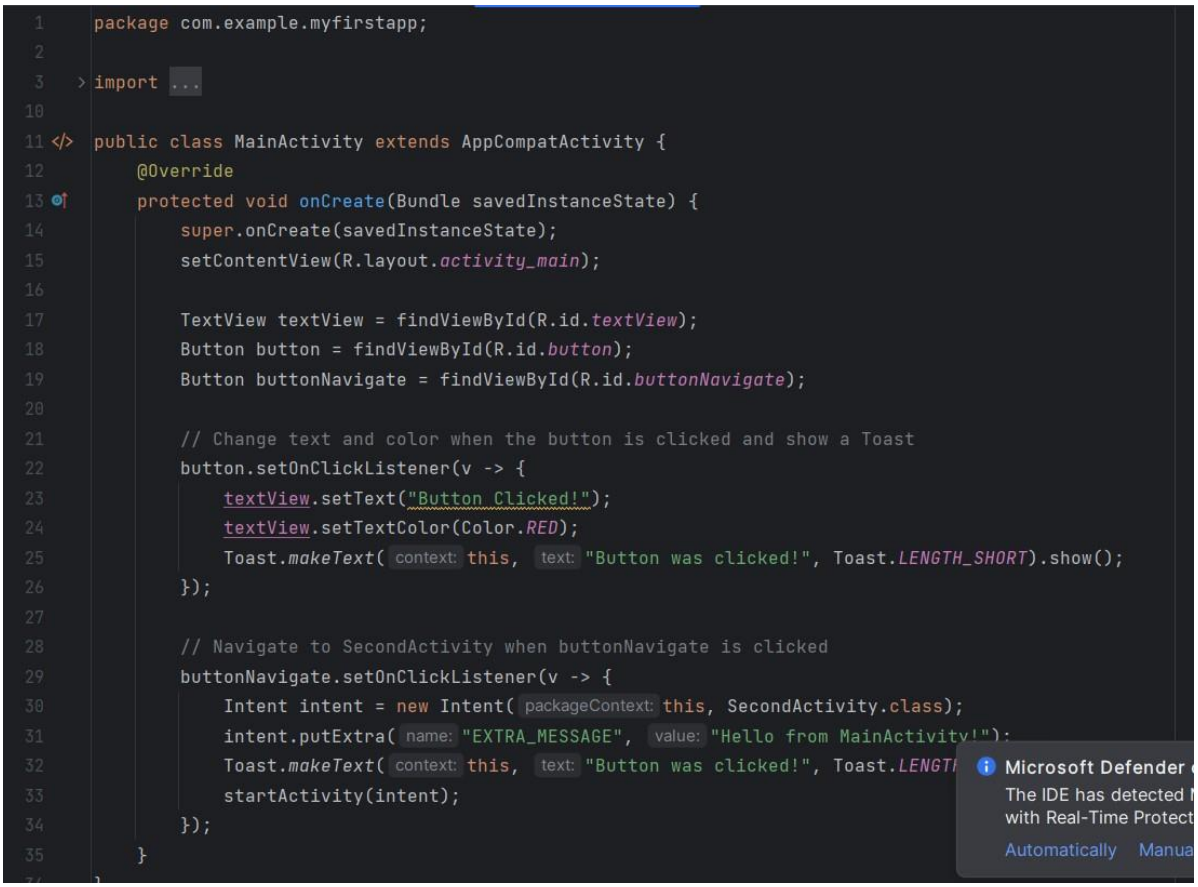
Bachelor of Computer Science (Mobile Computing) with Honors

Semester I 2024/2025

Github link : [riezuuuu/CSM3123-Native]

Task 1

Java



```java
package com.example.myfirstapp;

import ...

public class MainActivity extends AppCompatActivity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        TextView textView = findViewById(R.id.textView);
        Button button = findViewById(R.id.button);
        Button buttonNavigate = findViewById(R.id.buttonNavigate);

        // Change text and color when the button is clicked and show a Toast
        button.setOnClickListener(v -> {
            textView.setText("Button Clicked!");
            textView.setTextColor(Color.RED);
            Toast.makeText( context: this,  text: "Button was clicked!", Toast.LENGTH_SHORT).show();
        });

        // Navigate to SecondActivity when buttonNavigate is clicked
        buttonNavigate.setOnClickListener(v -> {
            Intent intent = new Intent( packageContext: this, SecondActivity.class);
            intent.putExtra( name: "EXTRA_MESSAGE",  value: "Hello from MainActivity!");
            Toast.makeText( context: this,  text: "Button was clicked!", Toast.LENGTH
            startActivity(intent);
        });
    }
```

Microsoft Defender
The IDE has detected
with Real-Time Protect
Automatically Manua

```java
package com.example.myfirstapp;

import ...

public class SecondActivity extends AppCompatActivity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_second);

        // Retrieve and display the passed message
        String message = getIntent().getStringExtra( name: "EXTRA_MESSAGE");
        TextView textView = findViewById(R.id.textViewSecond);
        textView.setText(message);

        // Set up the "Back to Main Activity" button
        Button buttonBack = findViewById(R.id.buttonBack);
        buttonBack.setOnClickListener(v -> finish()); // This will close SecondActivity and return to
    }
}
```

Kotlin

```kotlin
import ...

class MainActivity : AppCompatActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)

        val textView = findViewById<TextView>(R.id.textView)
        val button = findViewById<Button>(R.id.button)
        val buttonNavigate = findViewById<Button>(R.id.buttonNavigate)

        // Change text and color when the button is clicked and show a Toast
        button.setOnClickListener { it: View!
            textView.text = "Button Clicked!"
            textView.setTextColor(Color.RED)
            Toast.makeText( context: this,  text: "Button was clicked!", Toast.LENGTH_SHORT).show()
        }

        // Navigate to SecondActivity when buttonNavigate is clicked
        buttonNavigate.setOnClickListener { it: View!
            val intent = Intent( packageContext: this, SecondActivity::class.java)
            intent.putExtra( name: "EXTRA_MESSAGE",  value: "Hello from MainActivity!")
            Toast.makeText( context: this,  text: "Button was clicked!", Toast.LENGTH_SHORT).show()
            startActivity(intent)
        }
    }
}
```

MainActivity.kt | SecondActivity.kt | activity_main.xml | activity_second.xml | AndroidMani

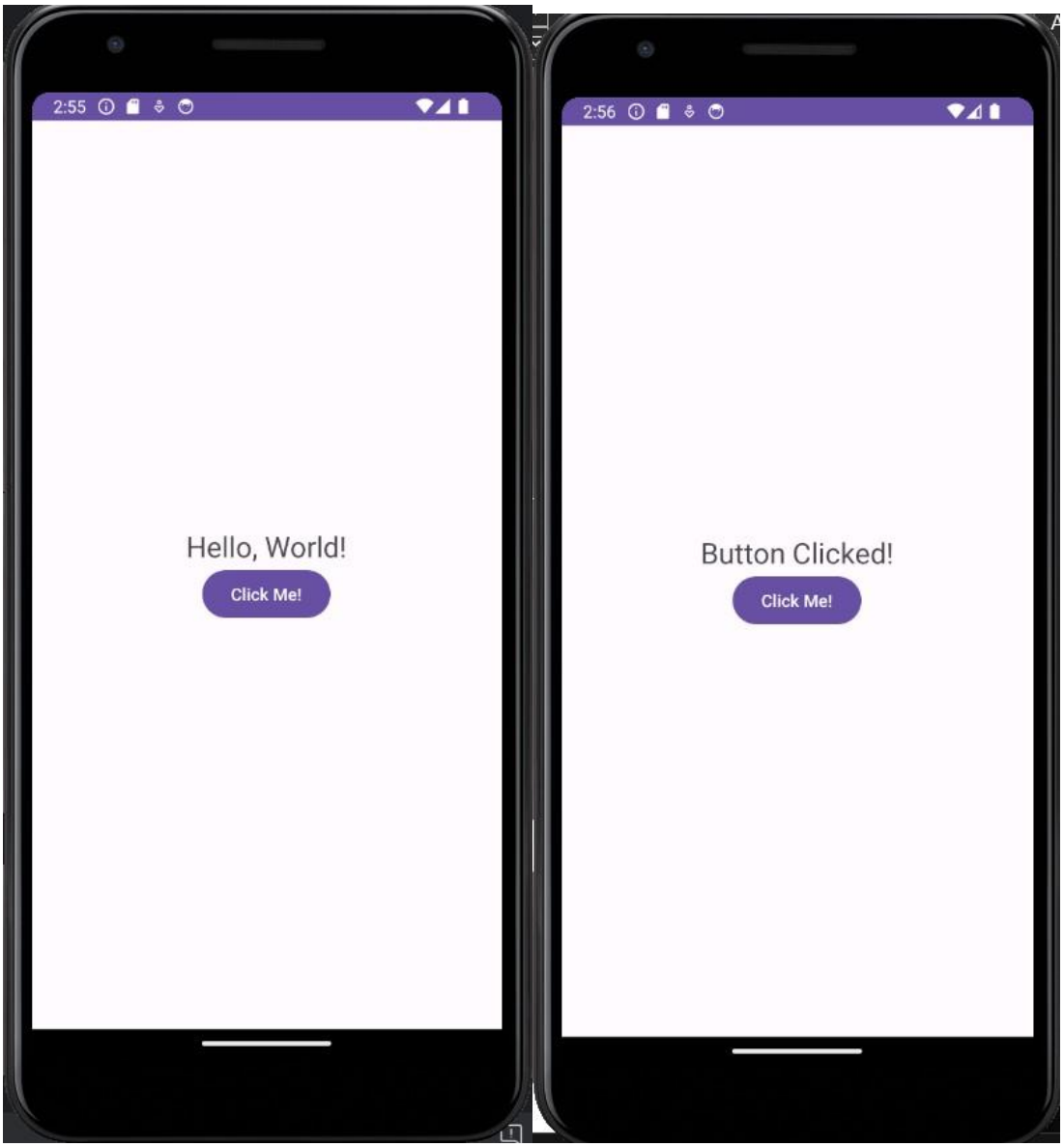
```kotlin
package com.example.myfirstapp

import ...

class SecondActivity : AppCompatActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_second)

        // Retrieve and display the passed message
        val message = intent.getStringExtra( name: "EXTRA_MESSAGE")
        val textView: TextView = findViewById(R.id.textViewSecond)
        textView.text = message

        // Set up the "Back to Main Activity" button
        val buttonBack = findViewById<Button>(R.id.buttonBack)
        buttonBack.setOnClickListener { it: View!
            finish() // This will close SecondActivity and return to MainActivity
        }
    }
}

```

XML

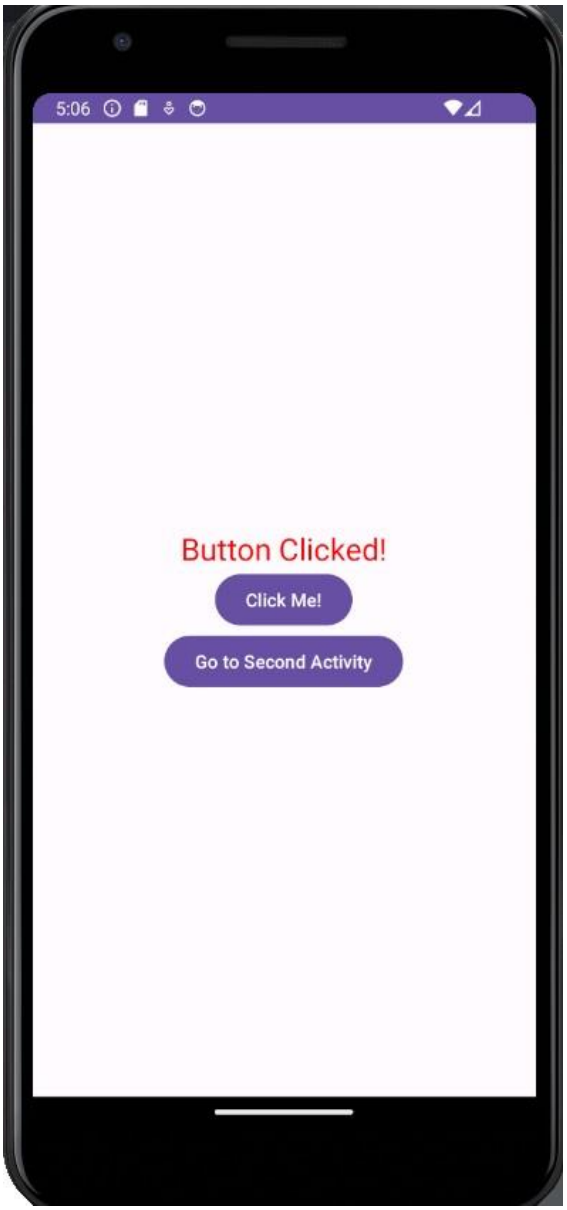
```xml
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:gravity="center">
    <TextView
        android:id="@+id/textView"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Hello, World!"
        android:textColor="@android:color/black"
        android:textSize="24sp" />
    <Button
        android:id="@+id/button"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Click Me!" />

    <Button
        android:id="@+id/buttonNavigate"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Go to Second Activity" />

</LinearLayout>
```

activity_main.xml | activity_second.xml | MainActivity.java | AndroidManifest.xml

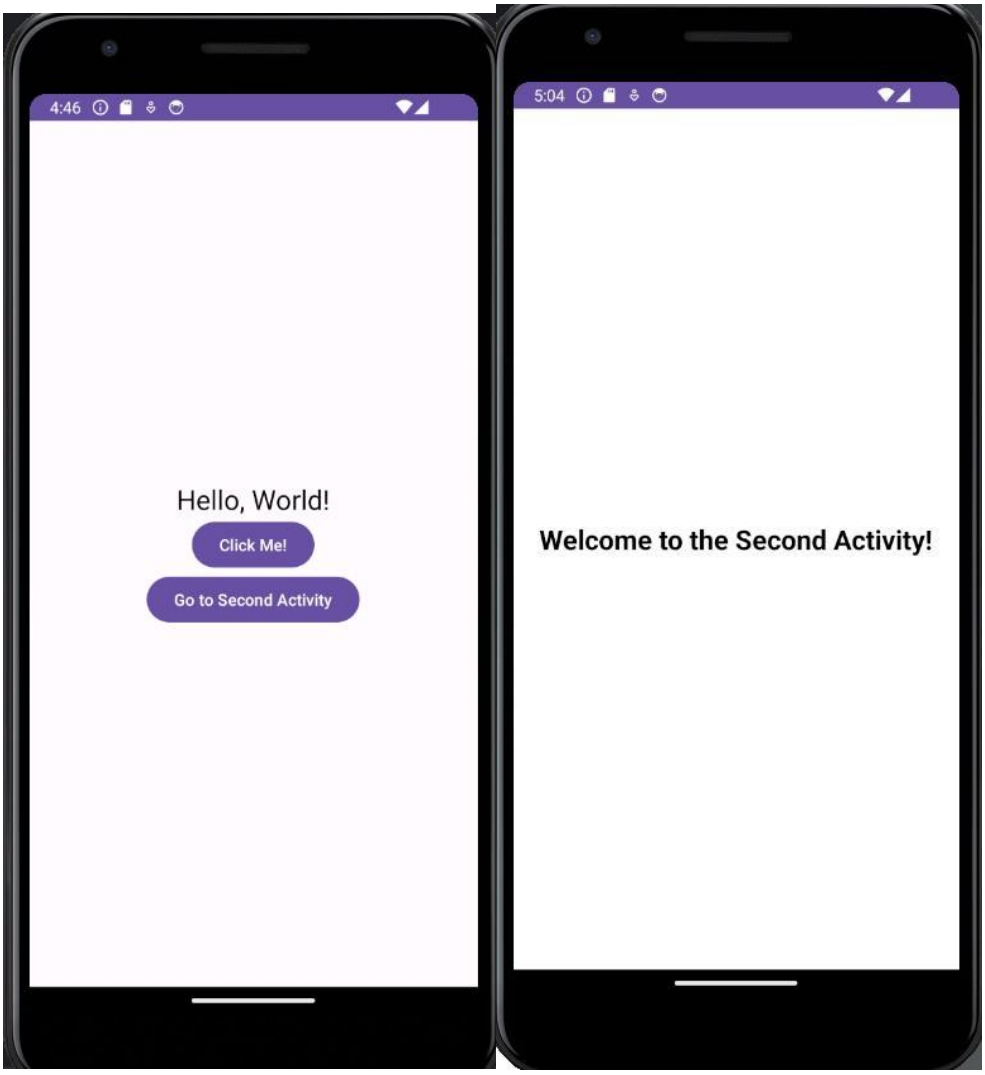
```xml
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:padding="16dp"
    android:background="#FFFFFF">

    <TextView
        android:id="@+id/textViewSecond"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_centerInParent="true"
        android:text="Welcome to the Second Activity!"
        android:textSize="24sp"
        android:textColor="#000000"
        android:textStyle="bold" />

    <!-- Back Button below the TextView -->
    <Button
        android:id="@+id/buttonBack"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_below="@id/textViewSecond"
        android:layout_centerHorizontal="true"
        android:layout_marginTop="16dp"
        android:text="Back to Main Activity" />
</RelativeLayout>
```


2:55
Hello, World!
Click Me!
2:56
Button Clicked!
Click Me!

Activity 1

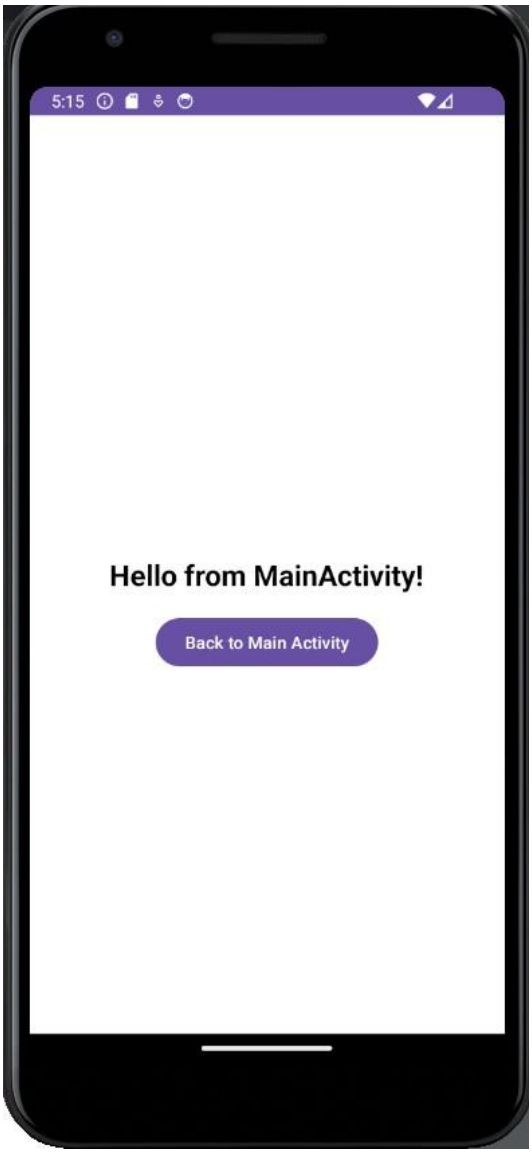


Activity 2

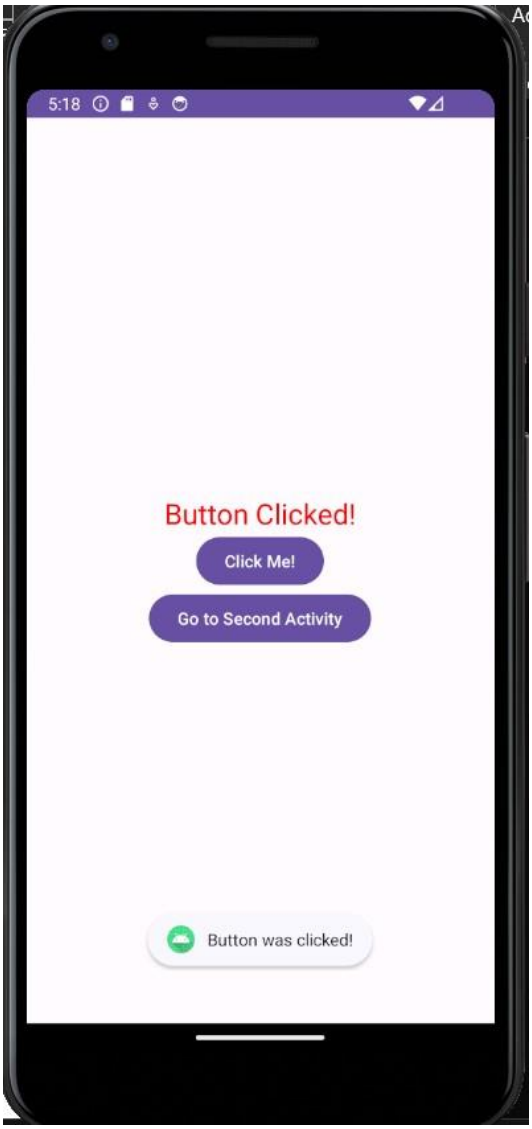

4:46
Hello, World!
Click Me!
Go to Second Activity
5:04
Welcome to the Second Activity!

Activity 3

Activity 4

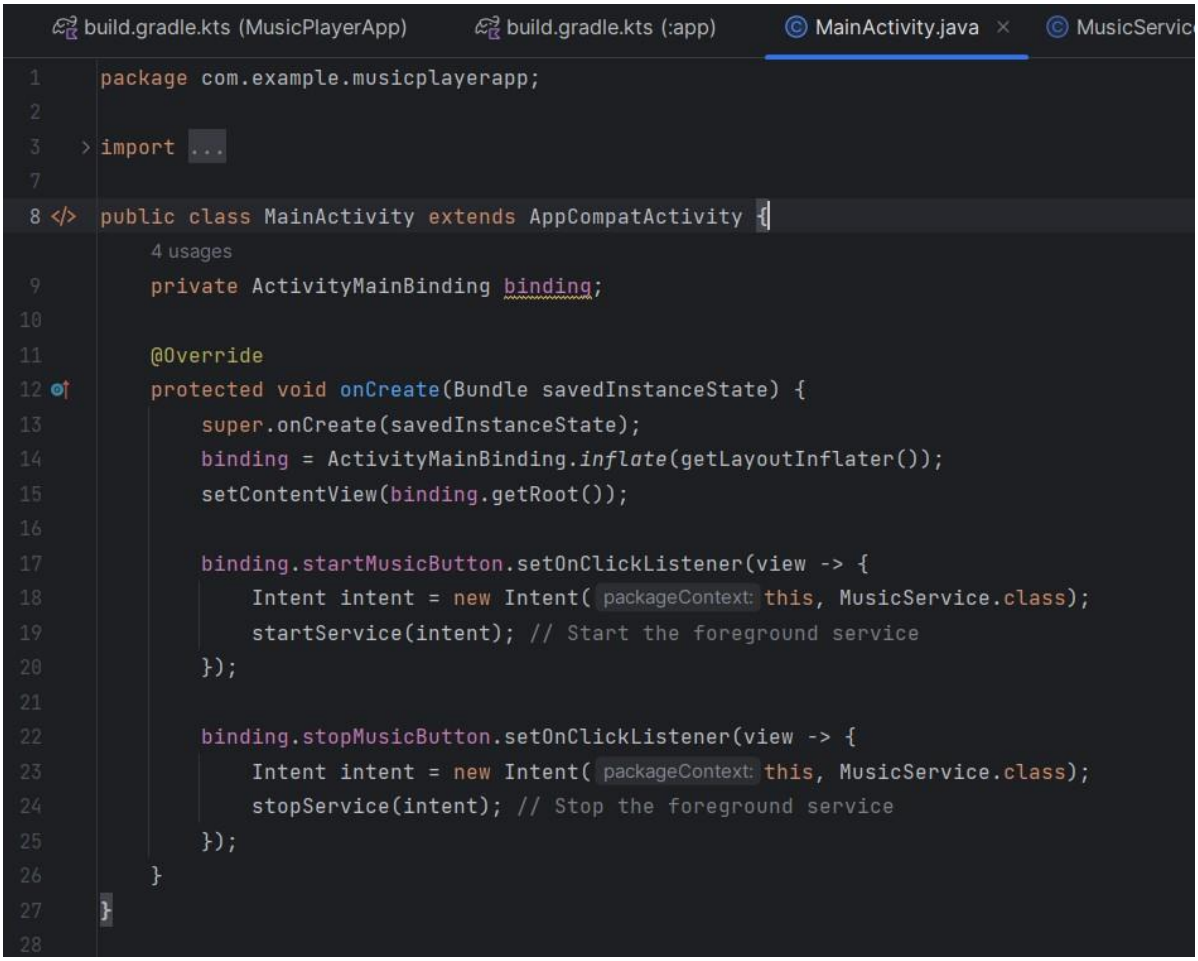


Activity 5


5:18
Button Clicked!
Click Me!
Go to Second Activity
Button was clicked!

Task 2

Java


build.gradle.kts (MusicPlayerApp)
build.gradle.kts (:app)
MainActivity.java
MusicServic
package com.example.musicplayerapp;
import ...
public class MainActivity extends AppCompatActivity {
4 usages
private ActivityMainBinding binding;
@Override
protected void onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState);
binding = ActivityMainBinding.inflate(getLayoutInflater());
setContentView(binding.getRoot());
binding.startMusicButton.setOnClickListener(view -> {
Intent intent = new Intent(packageContext: this, MusicService.class);
startService(intent); // Start the foreground service
});
binding.stopMusicButton.setOnClickListener(view -> {
Intent intent = new Intent(packageContext: this, MusicService.class);
stopService(intent); // Stop the foreground service
});
}
}

```java
public class MusicService extends Service {
    6 usages
    private MediaPlayer mediaPlayer;

    @Override
    public void onCreate() {
        super.onCreate();
        mediaPlayer = MediaPlayer.create( context: this, R.raw.song);
        mediaPlayer.setLooping(true);
    }

    @Override
    public int onStartCommand(Intent intent, int flags, int startId) {
        startForegroundServiceWithNotification();
        if (!mediaPlayer.isPlaying()) {
            mediaPlayer.start();
        }
        return START_STICKY;
    }

    1 usage
    @SuppressLint("ForegroundServiceType")
    private void startForegroundServiceWithNotification() {
        createNotificationChannel();
        Intent notificationIntent = new Intent( packageContext: this, MainActivity.class);
        PendingIntent pendingIntent = PendingIntent.getActivity(
                context: this,
                requestCode: 0,
                notificationIntent,
                flags: PendingIntent.FLAG_UPDATE_CURRENT | PendingIntent.FLAG_IMMUTABLE
        );
```

```java
    NotificationCompat.Builder notification = new NotificationCompat.Builder( context: this, channelId: "music_service_ch
            .setContentTitle("Music Player")
            .setContentText("Playing music...")
            .setSmallIcon(R.drawable.ic_music_note)
            .setContentIntent(pendingIntent);

    startForeground( id: 1, notification.build());
}

1 usage
private void createNotificationChannel() {
    if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.O) {
        NotificationChannel serviceChannel = new NotificationChannel(
                id: "music_service_channel",
                name: "Music Service Channel",
                NotificationManager.IMPORTANCE_HIGH
        );
        NotificationManager manager = getSystemService(NotificationManager.class);
        if (manager != null) {
            manager.createNotificationChannel(serviceChannel);
        }
    }
}
```

```java
    @Override
    public void onDestroy() {
        mediaPlayer.stop();
        mediaPlayer.release();
        super.onDestroy();
    }

    @Override
    public IBinder onBind(Intent intent) { return null; }
}
```

Kotlin

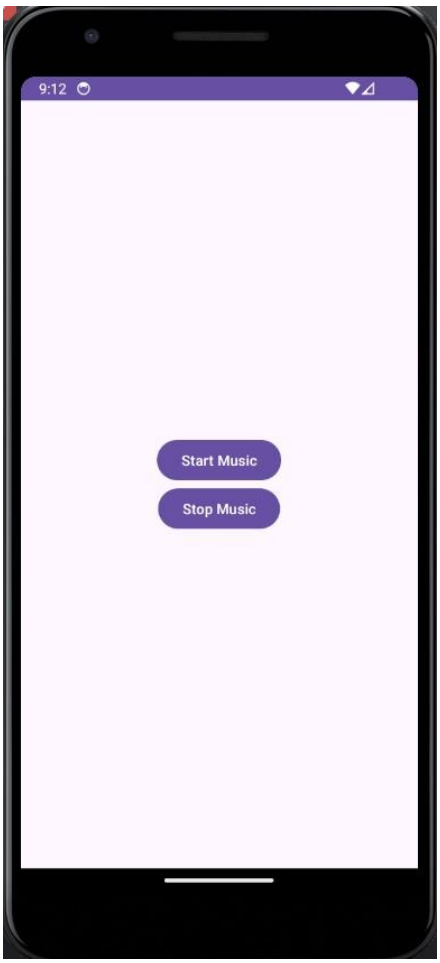
```kotlin
package com.example.musicplayerapp

import ...

class MainActivity : AppCompatActivity() {
    private lateinit var binding: ActivityMainBinding

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        binding = ActivityMainBinding.inflate(layoutInflater)
        setContentView(binding.root)

        binding.startMusicButton.setOnClickListener { it: View!
            val intent = Intent( packageContext: this, MusicService::class.java)
            startService(intent) // Start the foreground service
        }

        binding.stopMusicButton.setOnClickListener { it: View!
            val intent = Intent( packageContext: this, MusicService::class.java)
            stopService(intent) // Stop the foreground service
        }
    }
}
```

```kotlin
class MusicService : Service() {
    private lateinit var mediaPlayer: MediaPlayer

    override fun onCreate() {
        super.onCreate()
        mediaPlayer = MediaPlayer.create( context: this, R.raw.song)
        mediaPlayer.isLooping = true
    }

    override fun onStartCommand(intent: Intent?, flags: Int, startId: Int): Int {
        startForegroundServiceWithNotification()
        if (!mediaPlayer.isPlaying) {
            mediaPlayer.start()
        }
        return START_STICKY
    }

    @SuppressLint("ForegroundServiceType")
    private fun startForegroundServiceWithNotification() {
        createNotificationChannel()
        val notificationIntent = Intent( packageContext: this, MainActivity::class.java)
        val pendingIntent = PendingIntent.getActivity(
            context: this,
            requestCode: 0,
            notificationIntent,
            flags: PendingIntent.FLAG_UPDATE_CURRENT or PendingIntent.FLAG_IMMUTABLE // Add mutability flags
        )
```

```kotlin
    val notification = NotificationCompat.Builder( context: this,  channelId: "music_service_channel")
        .setContentTitle("Music Player")
        .setContentText("Playing music...")
        .setSmallIcon(R.drawable.ic_music_note)
        .build()

    startForeground( id: 1, notification)
}

private fun createNotificationChannel() {
    if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.O) {
        val serviceChannel = NotificationChannel(
            id: "music_service_channel",
            name: "Music Service Channel",
            NotificationManager.IMPORTANCE_HIGH
        )
        val manager = getSystemService(NotificationManager::class.java)
        manager?.createNotificationChannel(serviceChannel)
    }
}

override fun onDestroy() {
    mediaPlayer.stop()
    mediaPlayer.release()
    super.onDestroy()
}

override fun onBind(intent: Intent?): IBinder? {
    return null
}
```

XML

```xml
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:gravity="center"
    android:orientation="vertical">
    <Button
        android:id="@+id/startMusicButton"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Start Music" />
    <Button
        android:id="@+id/stopMusicButton"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Stop Music" />
</LinearLayout>
```



Task 3

Java

```java
public class MainActivity extends AppCompatActivity {
    3 usages
    private ActivityMainBinding binding;
    3 usages
    private MyCustomBroadcastReceiver myReceiver;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        binding = ActivityMainBinding.inflate(getLayoutInflater());
        setContentView(binding.getRoot());

        myReceiver = new MyCustomBroadcastReceiver();

        binding.sendBroadcastButton.setOnClickListener(view -> {
            // Create an intent with the custom action
            Intent intent = new Intent( action: "com.example.CUSTOM_ACTION");
            // Send the broadcast
            LocalBroadcastManager.getInstance( context: this).sendBroadcast(intent);
        });
    }

    @Override
    protected void onStart() {
        super.onStart();
        // Register the receiver with the intent filter
        LocalBroadcastManager.getInstance( context: this)
                .registerReceiver(myReceiver, new IntentFilter( action: "com.example.CUSTOM_ACTION"));
```

```java
    @Override
    protected void onStart() {
        super.onStart();
        // Register the receiver with the intent filter
        LocalBroadcastManager.getInstance( context: this)
                .registerReceiver(myReceiver, new IntentFilter( action: "com.example.CUSTOM_ACTION"));
    }

    @Override
    protected void onStop() {
        super.onStop();
        // Unregister the receiver to prevent memory leaks
        LocalBroadcastManager.getInstance( context: this).unregisterReceiver(myReceiver);
    }
}
```

```java
package com.example.custombroadcastreceiverapp;

import ...

public class MyCustomBroadcastReceiver extends BroadcastReceiver {
    1 usage
    private static final String TAG = "MyBroadcastReceiver";

    @Override
    public void onReceive(Context context, Intent intent) {
        // Check if the action matches the expected custom action
        if ("com.example.CUSTOM_ACTION".equals(intent.getAction())) {
            Log.d(TAG, msg: "Custom Broadcast Received!");
            Toast.makeText(context, text: "Custom Broadcast Received!", Toast.LENGTH_SHORT).show();
        }
    }
}
```

Kotlin

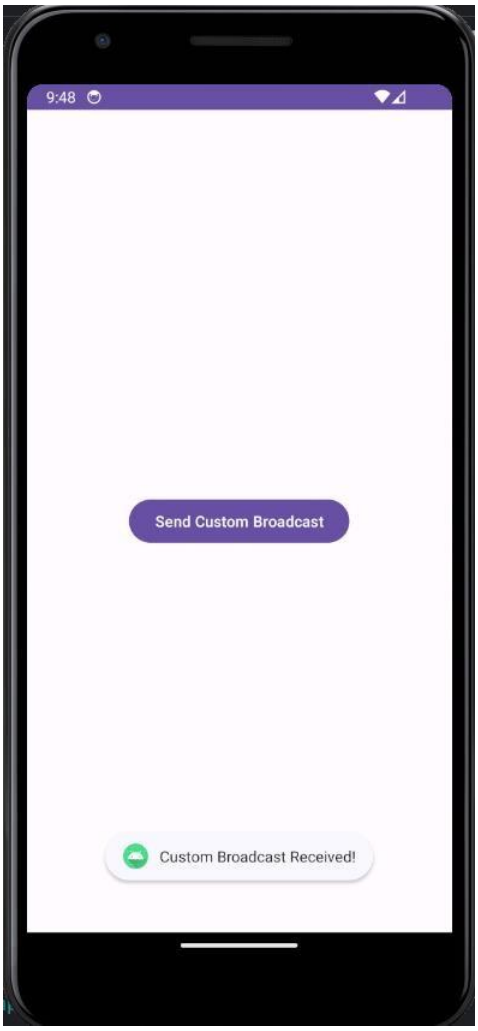
```kotlin
class MainActivity : AppCompatActivity() {
    private lateinit var binding: ActivityMainBinding
    private lateinit var myReceiver: MyCustomBroadcastReceiver

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        binding = ActivityMainBinding.inflate(layoutInflater)
        setContentView(binding.root)

        myReceiver = MyCustomBroadcastReceiver()

        binding.sendBroadcastButton.setOnClickListener { it: View!
            // Create an intent with the custom action
            val intent = Intent( action: "com.example.CUSTOM_ACTION")
            // Send the broadcast
            LocalBroadcastManager.getInstance( context: this).sendBroadcast(intent)
        }
    }

    override fun onStart() {
        super.onStart()
        // Register the receiver with the intent filter
        LocalBroadcastManager.getInstance( context: this)
            .registerReceiver(myReceiver, IntentFilter( action: "com.example.CUSTOM_ACTION"))
    }
```

```kotlin
    override fun onStop() {
        super.onStop()
        // Unregister the receiver to prevent memory leaks
        LocalBroadcastManager.getInstance( context: this).unregisterReceiver(myReceiver)
    }
}
```

```kotlin
import ...

class MyCustomBroadcastReceiver : BroadcastReceiver() {
    private val TAG = "MyBroadcastReceiver"

    override fun onReceive(context: Context, intent: Intent) {
        // Check if the action matches the expected custom action
        if (intent.action == "com.example.CUSTOM_ACTION") {
            Log.d(TAG, msg: "Custom Broadcast Received!")
            Toast.makeText(context, text: "Custom Broadcast Received!", Toast.LENGTH_SHORT).show()
        }
    }
}
```

XML

```xml
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <Button
        android:id="@+id/sendBroadcastButton"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Send Custom Broadcast"
        android:layout_centerInParent="true"/>
</RelativeLayout>

```




```
com...le.custombroadcastreceiverapp  D  app_time_stats: avg=280.06ms min=2.20ms max=2951.39ms count=11
com...le.custombroadcastreceiverapp  D  Custom Broadcast Received!
rter com...le.custombroadcastreceiverapp  D  Compat change id reported: 147798919; UID 10205; state: ENABLED
com...le.custombroadcastreceiverapp  D  Installing profile for com.example.custombroadcastreceiverapp
com...le.custombroadcastreceiverapp  D  app_time_stats: avg=977.60ms min=2.00ms max=45286.78ms count=47
com...le.custombroadcastreceiverapp  D  Custom Broadcast Received!
```

Answer the following question.

1. **In the provided implementation, why is the custom action string ("com.example.CUSTOM_ACTION") used? What would happen if this string were changed in one place but not another?**

   The custom action string "com.example.CUSTOM_ACTION" ensures that the broadcast is uniquely identified. If changed in only one place (e.g., in the broadcast but not in the receiver), the broadcast would fail, as the receiver wouldn't recognize it.

2. **What are the advantages of using LocalBroadcastManager over regular broadcasts?**

LocalBroadcastManager is used for sending broadcasts within an app, with several benefits over regular broadcasts:

- **Security**: Broadcasts remain within the app, reducing the risk of exposing sensitive data.
- **Efficiency**: Local broadcasts are more efficient because they don't cross process boundaries.
- **Less Overhead**: There's no need for permissions, simplifying broadcast communication within the app.

3. **Describe how you would modify the code to send a broadcast with additional data (using Intent.putExtra). Provide an example.**

To send a broadcast with additional data, use Intent.putExtra to attach data to the broadcast. For example, if we wanted to send a broadcast with a user's name, we would modify the code like this:

```
// Sending the broadcast with additional data val intent = Intent("com.example.CUSTOM_ACTION") intent.putExtra("username", "John Doe") // Adding extra data LocalBroadcastManager.getInstance(this).sendBroadcast(intent)
```

In MyCustomBroadcastReceiver, retrieve this data as follows:

```
override fun onReceive(context: Context, intent: Intent) { if (intent.action == "com.example.CUSTOM_ACTION") { val username = intent.getStringExtra("username") Toast.makeText(context, "Received for user: $username", Toast.LENGTH_SHORT).show() } }
```

4. **If you wanted to receive broadcasts from multiple activities, how would you modify the existing BroadcastReceiver setup?**

To receive broadcasts from multiple activities, you could:

- **Register the BroadcastReceiver in each activity**: Each activity would register the same BroadcastReceiver instance, allowing them all to receive the broadcast.
- **Use a centralized registration in a parent or base activity**: If your activities extend a base activity, you could register the BroadcastReceiver in that base activity so all child activities can receive the broadcast.
- **Broadcast Data Filtering**: Use different action strings or data filters to distinguish broadcasts intended for each specific activity.