Semester	T.E. Semester VI – Computer Engineering
Subject	Mobile Computing
Subject Professor In-charge	Prof. Sneha Annappanavar
Assisting Teachers	Prof. Sneha Annappanavar
Laboratory	M310A

Student Name	Deep Salunkhe
Roll Number	21102A0014
TE Division	A

Title:

Introduction to android development 2

Explanation:

Android development using Java is a popular approach for creating mobile applications for devices running the Android operating system. Java is the primary language used for Android app development, offering a robust and well-supported environment for building powerful, feature-rich applications. Below is a comprehensive introduction to Android development using Java:

1. **Understanding Android: Android is an open-source mobile operating system developed by Google. It powers billions of devices worldwide, including smartphones, tablets, smartwatches, and even smart TVs. Android provides developers with a rich set of tools and APIs to create innovative and engaging applications.**

2. **Setting Up Development Environment: Before you start developing Android applications, you need to set up your development environment. This includes installing Android Studio, the official integrated development environment (IDE) for Android development. Android Studio provides all the necessary tools for designing, coding, testing, and debugging Android apps.**

3. **Understanding the Android Architecture: Android applications are built using a layered architecture. At the core is the Linux kernel, which provides low-level hardware abstraction and basic system services. On top of the kernel, there are various layers such as the native libraries, the Android Runtime (ART), the application framework, and finally, the applications themselves.**

4. **Learning Java Programming: Since Java is the primary language for Android development, it's essential to have a good understanding of Java programming concepts. This includes object-oriented programming principles, data types, control structures, arrays, collections, and exception handling.**

5. **Understanding Android Components: Android applications are composed of various components that interact with each other to provide the desired functionality. The main components include Activities, Services, Broadcast Receivers, and Content Providers. Understanding how these components work together is crucial for building efficient and scalable Android applications.**

6. Creating User Interfaces with XML and Layouts: Android user interfaces (UIs) are created using XML layout files, which define the structure and appearance of the UI elements. Android provides a wide range of layout managers and UI widgets to design responsive and visually appealing user interfaces.

7. Handling User Input and Events: Android applications interact with users through various input methods such as touch gestures, keyboard input, and voice commands. Handling user input and events is an essential aspect of Android development, and it involves implementing event listeners and callbacks to respond to user actions.

8. Working with Resources: Android applications use various types of resources such as images, strings, colors, and dimensions. These resources are stored in the res directory of the project and can be accessed programmatically using resource identifiers.

9. Understanding Activities and Fragments: Activities are the building blocks of Android applications, representing individual screens or UI components. Fragments are reusable UI components that can be combined within activities to create flexible and modular UI designs.

10. Testing and Debugging: Testing is a crucial part of the Android development process. Android Studio provides tools for testing your applications on virtual devices or physical devices connected to your development machine. You can also use debugging tools to identify and fix errors in your code.

Implementation: -

```
package com.example.exp1;

import androidx.appcompat.app.AppCompatActivity;

import android.os.Bundle;
import android.graphics.Color;
import android.view.View;
import android.widget.Button;
import android.widget.TextView;

public class MainActivity extends AppCompatActivity
{
    int ch=1;
    float font=30;
    @Override
    protected void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        final TextView t= (TextView) findViewById(R.id.textView);
        Button b1= (Button) findViewById(R.id.button1);
        b1.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                t.setTextSize(font);
```

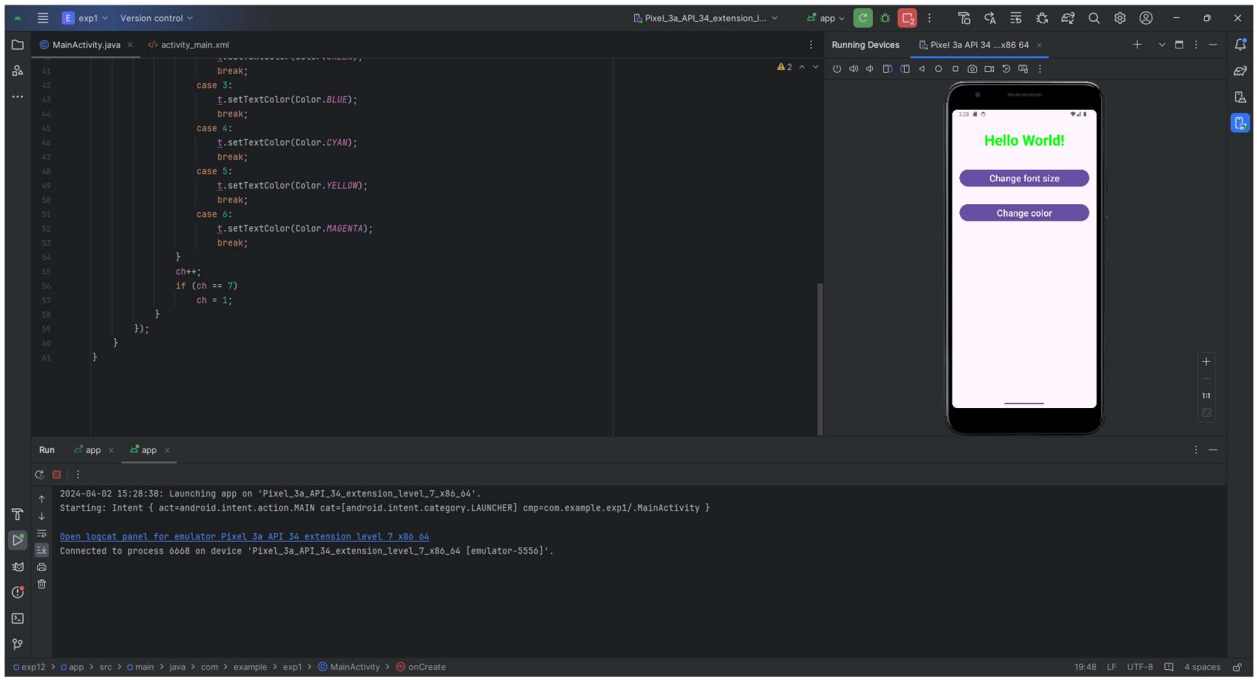
```
                font = font + 5;
                if (font == 50)
                    font = 30;
            }
        });
        Button b2= (Button) findViewById(R.id.button2);
        b2.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                switch (ch) {
                    case 1:
                        t.setTextColor(Color.RED);
                        break;
                    case 2:
                        t.setTextColor(Color.GREEN);
                        break;
                    case 3:
                        t.setTextColor(Color.BLUE);
                        break;
                    case 4:
                        t.setTextColor(Color.CYAN);
                        break;
                    case 5:
                        t.setTextColor(Color.YELLOW);
                        break;
                    case 6:
                        t.setTextColor(Color.MAGENTA);
                        break;
                }
                ch++;
                if (ch == 7)
                    ch = 1;
            }
        });
    }
}
```

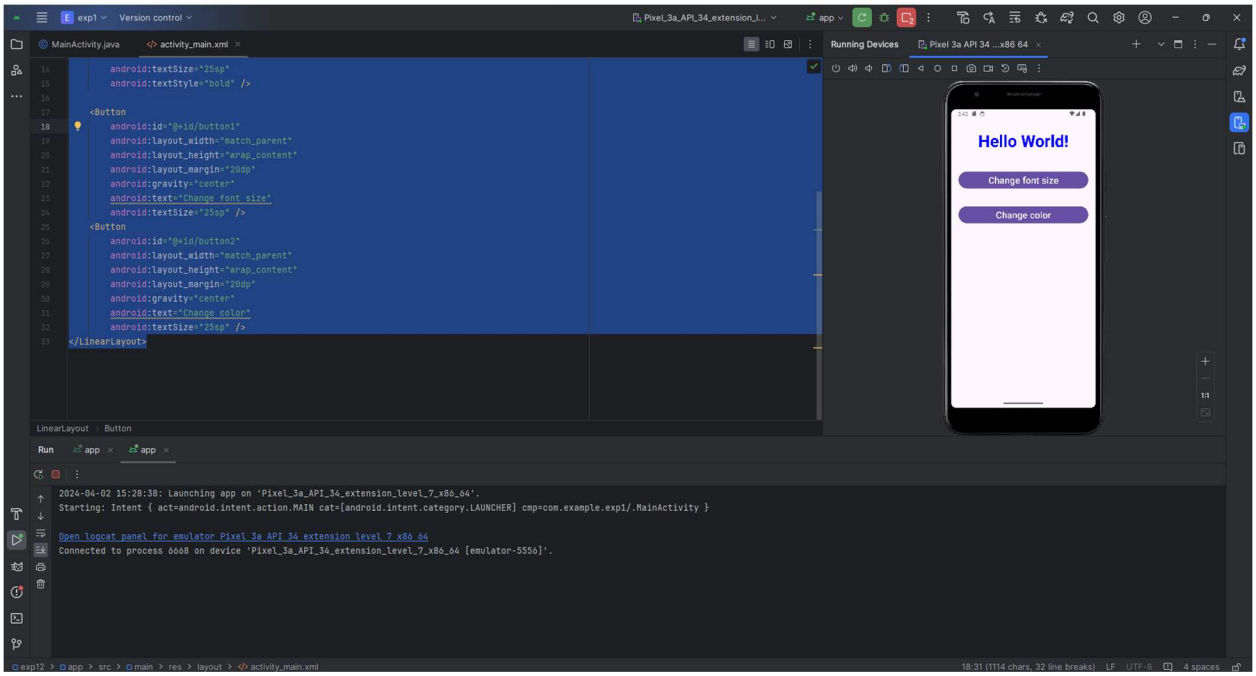
```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <TextView
        android:id="@+id/textView"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_margin="30dp"
        android:gravity="center"
        android:text="Hello World!"
        android:textSize="25sp"
        android:textStyle="bold" />
```

```
    <Button
        android:id="@+id/button1"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_margin="20dp"
        android:gravity="center"
        android:text="Change font size"
        android:textSize="25sp" />
    <Button
        android:id="@+id/button2"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_margin="20dp"
        android:gravity="center"
        android:text="Change color"
        android:textSize="25sp" />
</LinearLayout>
```

End Result:





Conclusion:

By mastering these fundamentals of Android development using Java, you'll be well-equipped to create a wide range of Android applications, from simple utility apps to complex, feature-rich applications. As you gain experience and proficiency, you can explore advanced topics such as performance optimization, security, and integration with other technologies and services.