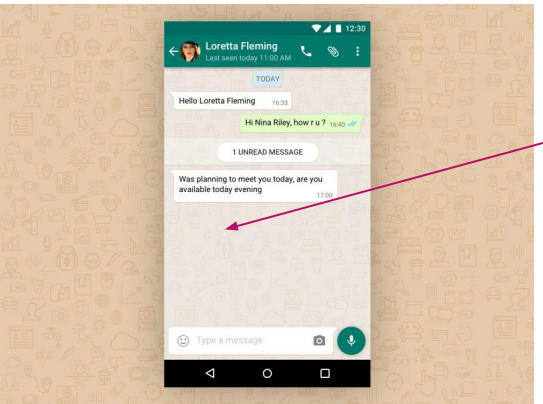
Introduction to Android Studio

ANDROID STUDIO IS THE OFFICIAL INTEGRATED DEVELOPMENT ENVIRONMENT (IDE) FOR ANDROID APP DEVELOPMENT.

Introduction to Activity

- An activity represents a single screen with a user interface i.e. Activity performs actions on the screen.
- An activity provides the window in which the app draws its UI.



This chat within WhatsApp is a single activity.

Parts of Activity

- An Activity consists of 2 parts-

1.) A layout file which defines the visual structure for a user interface, such as the UI for an activity or app widget.

2.)A Java class which can execute commands and generate outputs based on the interactions in the UI.

Basic Configuration Files

- **AndroidManifest.xml-**The manifest presents essential information about the application to the Android system, information the system must have before it can run any of the application's code. It names the package,describes the components of the application, and the permission required to interact with APIs and other components.

Basic Configuration Files

- **Gradle-**Gradle can be described a structured building mechanism where it provides a developer the tools and flexibility to manage the resources of a project

Intent Filter

- An **intent filter** is an expression in an app's manifest file that specifies the type of **intents** that the component would like to receive.
- Intent filter can be used to specify the activity which is first launched when the app is opened i.e. the main activity.

Building Layouts

- The Layouts for Activities are built using **XML**, because it is a lightweight language which is easy to implement.
- The whole concept of **Android UI** is defined using the hierarchy of View and ViewGroup objects. A ViewGroup is an invisible container that organizes child views.
- These child views are other widgets which are used to make the different parts of UI. ViewGroups can contain other viewgroups.

Layouts

- Various different types of layouts, which are subclasses of ViewGroup class, are used for positioning the various child views.
- A typical layout defines the visual structure for an Android user interface.

Types of Layouts

- **Linear Layout**- All the elements are displayed in a linear fashion, either horizontally or vertically.
- **Relative Layout-** Used to position our view components /views based on the relative or sibling component's position.

Constraint Layout

Various Views in Android

- A View occupies a rectangular area on the screen and is responsible for drawing and event handling.
- Commonly used Views are :
- EditText
- ImageView
- TextView
- Button

Widgets

- **TextView**- This is used to display text on the application screen.
- **EditText**-Used to get input from User for various purposes.
- **ImageView**-Used to display Image on the application screen.
- **Button**-A user interface element the user can tap or click to perform an action.

XML Attributes

- Setting a View's size to wrap content will force it to expand only far enough to contain the values it contains.
- Setting it to fill parent will force it to expand to take up as much space as is available within the layout element it's been placed in.
- The sizes are sometimes set in dp(density independent pixels) px(pixels), and sp(Scale-independent pixels).

XML Basics

- Every view object defined using XML should have a **starting as well as a closing tag.**
- The width and height of each view object has to be defined.
- Every view object has an id associated with it through which it can interact with other view objects as well as the methods defined in the java class of the layout.

XML Basics

- A simple TextView XML code looks as follows:
- **<TextView**

  android:layout_width="wrap_content" – Used to set width

  android:layout_height="wrap_content"-Used to set height

  android:text="Hello world"- Define text to be displayed

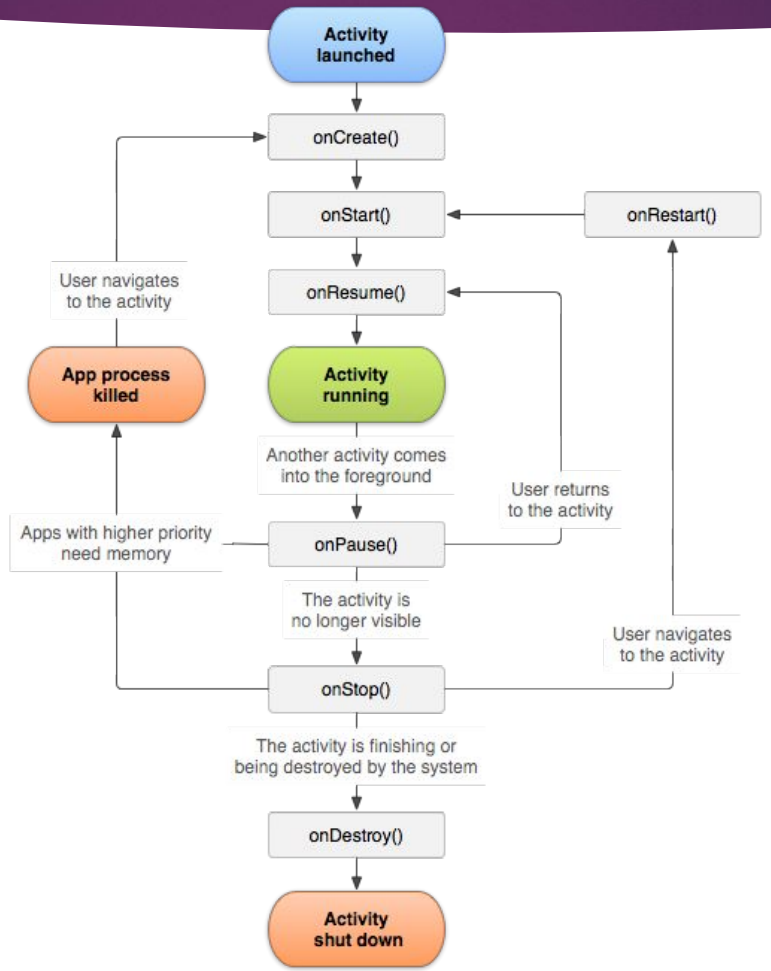
  android:id= "@id/textview" – Used to associate ID with the view.

  />

Linking layout with Java class

- After creating the layout of the activity, we need to create a java class associated with that activity as well.
- This is done by creating new class by extending it from a predefined class in the android framework called **Activity/AppCompatActivity.**
- Then we **override** the **OnCreate** method of the Activity class.

Activity Lifecycle



Activity Lifecycle

- **OnCreate**- Called when the activity is first created. In the **onCreate()** method, you perform basic application startup logic that should happen only once for the entire life of the activity. For example, your implementation of onCreate() might bind data to lists, associate the activity with a ViewModel, and instantiate some class-scope variables.

Activity Lifecycle

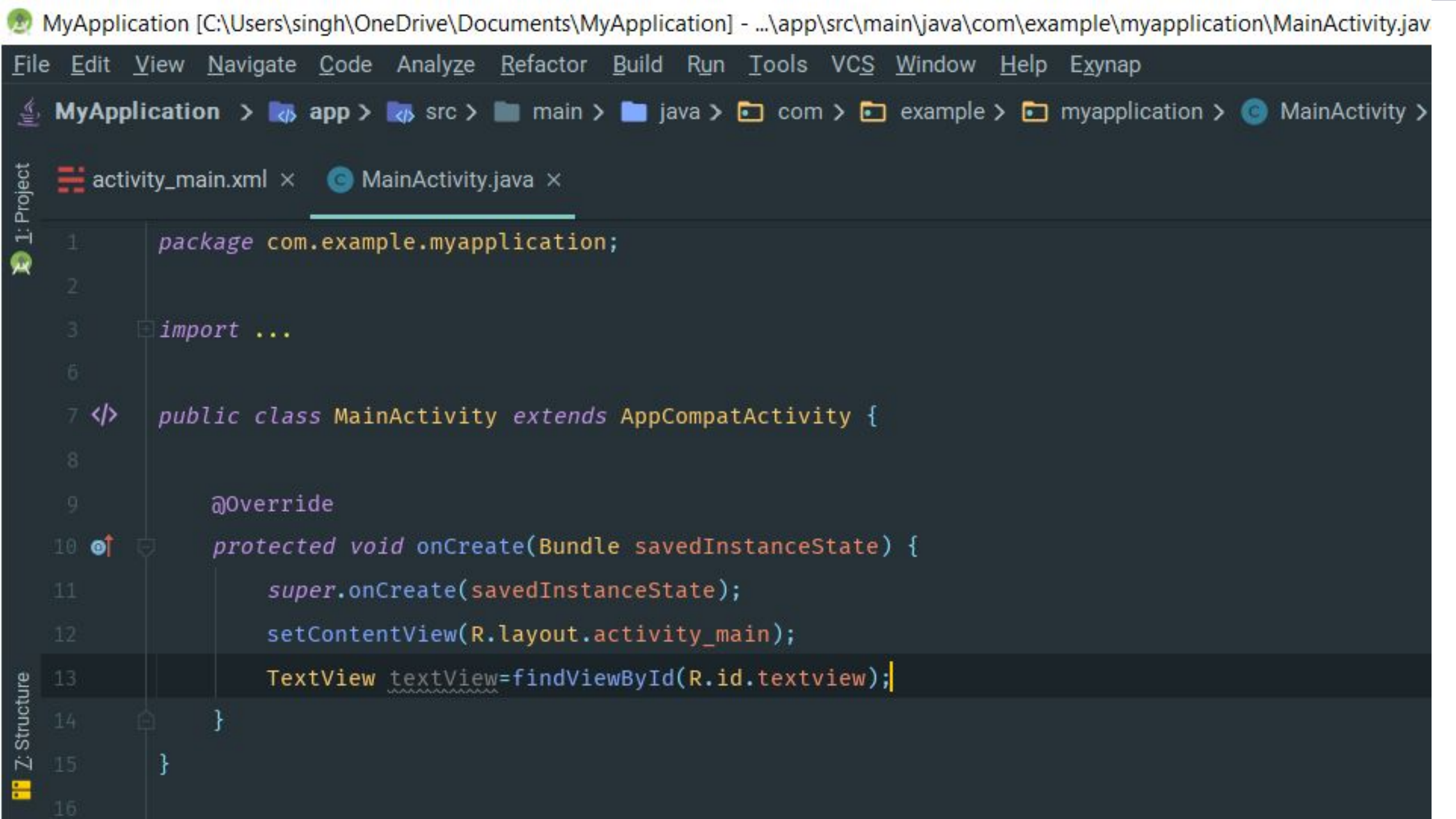
- **OnStart-** The **onStart()** call makes the activity visible to the user, as the app prepares for the activity to enter the foreground and become interactive .
- **OnResume()-** When the activity enters the Resumed state, it comes to the foreground, and then the system invokes the onResume() callback. This is the state in which the app interacts with the user.

Activity Lifecycle

- **onPause()-** The system calls this method as the first indication that the user is leaving your activity (though it does not always mean the activity is being destroyed); it indicates that the activity is no longer in the foreground.
- **onStop()-**When your activity is no longer visible to the user, it has entered the Stopped state, and the system invokes the onStop() callback.
- **onDestroy()-**onDestroy() is called before the activity is destroyed.

OnCreate

- The OnCreate Bundle is used to initialize our activity.
- Here, **setContentView** is called with a layout resource(R.layout.layoutname) defining the UI.
- Also the **findViewById** method is called to retrieve the widgets from the UI that we need to interact with programmatically.


MyApplication [C:\Users\singh\OneDrive\Documents\MyApplication] - ...\app\src\main\java\com\example\myapplication\MainActivity.jav
File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help Exynap
MyApplication > app > src > main > java > com > example > myapplication > MainActivity >
activity_main.xml
MainActivity.java
1: Project
Z: Structure
1 package com.example.myapplication;
2
3 import ...
6
7 public class MainActivity extends AppCompatActivity {
8
9 @Override
10 protected void onCreate(Bundle savedInstanceState) {
11 super.onCreate(savedInstanceState);
12 setContentView(R.layout.activity_main);
13 TextView textView=findViewById(R.id.textview);
14 }
15 }
16