School of Science & Engineering
Department of CSE
Canadian University of Bangladesh

Lecture-16: Android Programming Tips

Prof. Miftahur Rahman, Ph.D.

Semester: Summer 2024

Steps of building an Android app:

1. Set Up Android Studio: Download and install Android Studio, the official IDE for Android development.
2. Start a New Project: Select "New Project" and choose a template (e.g., Empty Activity).
3. Design the UI: Use XML or the visual editor to design the app layout in the `activity_main.xml` file.
4. Write Code: Implement logic in Java or Kotlin in `MainActivity.java`.
5. Test: Use an emulator or a connected device to run the app.
6. Build and Deploy: Click on "Build APK" to generate the app for installation.

Building an Android app in Android Studio:

1. Open Android Studio and start a new project. Select Empty Activity.
2. Design the UI: Go to `activity_main.xml` and add a `TextView`:

```
<TextView
    android:id="@+id/textView"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Hello, Android!"
    android:layout_gravity="center"/>
```

3. Code the Logic: In `MainActivity.java`, set up logic:

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

4. Run the App: Connect a device or use an emulator, then click Run.

To create and save the MonthAdapter.java file in your Android project, follow these steps:

Step 1: Locate the Java Source Directory
Open your Android project in Android Studio.

app > src > main > java > [your_package_name]

Replace [your_package_name] with the package name you chose when creating the project (e.g., com.example.yearlycalendarapp).

Step 2: Create the Java Class
Right-click on your package name (the folder under java) where your MainActivity.java is located.
Go to New > Java Class.

Step 3: Name the Class
In the dialog that appears, enter MonthAdapter as the class name.
Click OK.

Step 4: Paste the Code
Replace the default content in the newly created MonthAdapter.java file with the code provided earlier:

```
package com.example.yearlycalendarapp;
import android.content.Context;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.Button;
import android.widget.Toast;
import androidx.annotation.NonNull;
import androidx.recyclerview.widget.RecyclerView;

public class MonthAdapter extends RecyclerView.Adapter<MonthAdapter.MonthViewHolder> {
    private final String[] months;
    private final Context context;

    public MonthAdapter(Context context) {
        this.context = context;
        months = new String[]{
                "January", "February", "March", "April", "May", "June",
                "July", "August", "September", "October", "November", "December"
        };
    }
```

```
@NonNull
    @Override
    public MonthViewHolder onCreateViewHolder(@NonNull ViewGroup parent, int viewType) {
        View view = LayoutInflater.from(parent.getContext()).inflate(R.layout.item_month, parent,
false);
        return new MonthViewHolder(view);
    }

    @Override
    public void onBindViewHolder(@NonNull MonthViewHolder holder, int position) {
        holder.monthButton.setText(months[position]);
        holder.monthButton.setOnClickListener(v -> {
            Toast.makeText(context, "Selected Month: " + months[position],
Toast.LENGTH_SHORT).show();
        });
    }
```

```
@Override
    public int getItemCount() {
        return months.length;
    }

    static class MonthViewHolder extends RecyclerView.ViewHolder {
        Button monthButton;

        public MonthViewHolder(@NonNull View itemView) {
            super(itemView);
            monthButton = itemView.findViewById(R.id.monthButton);
        }
    }
}
```

Java for your Android project and don't have a build.gradle file, you might be working in a project structure that either doesn't follow the standard Gradle setup or you're using an older project format. Here’s what
you can do:

Step 1: Verify Project Structure
Check for build.gradle Files:
Open the app directory in your Android Studio project.
Look for a file named build.gradle. There should be one at the root of the app directory and possibly another one in the root of your project.
Step 2: Create a New Gradle Build File (If Missing)
If you find that the build.gradle file is indeed missing, you can create a new one:

Create a New build.gradle File:

Right-click on the app directory.
Select New > File and name it build.gradle.
Add Basic Configuration: Here’s a basic example of what you can include in your build.gradle file for a Java project:

Android TextView

In android **TextView** is a user interface control or widget which is used to display text to users on screen. It is similar to label in HTML. Basically it is a complete text editor but configured not to be edited, but we can edit it. Instead of editing it we can use EditText control for editing.

Check this android UI controls tutorial to know more about UI controls. **TextView** is subclass of **View** class which is base class for all the UI Controls in android.Since it is a sub class of **View** class we can use it inside a **View Group** or as a root element in android layout file.

Android EditText

In android **EditText** is a user interface control which is used to accept input from the user or to edit the text. **EditText** is nothing but a text field in HTML which accepts input. EditText is a sub class of TextView that configures itself to be editable.

Check this android UI controls tutorial to know more about UI controls.

EditText is subclass of **View** class which is base class for all the UI controls in android.Since it is a sub class of **View** class we can use it inside a **View Group** or as a root element in android layout file.

Defining EditText in Android

Defining **EditText** in android can be done in two ways, one is by declaring it inside a layout(XML) file and the second one is by creating an object of it in JAVA.

Defining EditText in XML(layout)

Example code snippet for defining EditText in 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:background="@color/white"
    android:id="@+id/textlaout"
    android:padding="20dp">

    <EditText
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        />
</LinearLayout>
```

Defining EditText in JAVA(programatically)

Example code snippet for defining EditText in JAVA.

```
package com.codesinsider.androidui;

import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.widget.LinearLayout;
import android.widget.TextView;

public class LayoutExample extends AppCompatActivity {

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

        EditText email = new EditText(this);
        //setting width and height of textview
        email.setLayoutParams(new LinearLayout.LayoutParams(50, 50));
        LinearLayout layout = findViewById(R.id.linear);
        layout.addView(email);
    }
}
```

Android EditText Attributes

Like all the UI controls in android EditText also has some attributes, so lets see the most commonly used attributes of EditText in detail.

id : The id is the most common attribute used to identify the View or ViewGroup created in xml file in java.

Example code snippet for setting an id to an EditText in XML.

```
<EditText
      android:id="@+id/email"
      android:layout_width="match_parent"
      android:layout_height="wrap_content"
      />
```

gravity : gravity attribute is used to position the text of the EditText to right, left, top, bottom, centre_vertical, center_horizontal etc.By default the gravity is start(left).

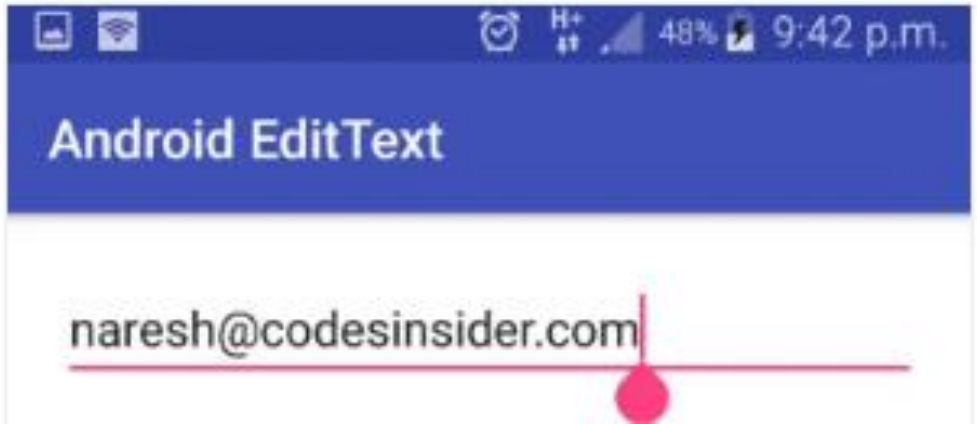
Example code snippet for setting the gravity of EditText to center in XML.

```
<EditText
    android:id="@+id/email"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:text="Codes insider"
    android:gravity="center"
    />
```

Example code snippet for setting gravity of EditText in JAVA.

```
EditText email = findViewById(R.id.email);
email.setGravity(Gravity.CENTER);
```

Output of the above code snippets.



layout_gravity : layout gravity is similar to gravity but positions the entire EditText layout instead of text.By default
the layout gravity is start(left).

Example code snippet for EditText with layout_gravity as center_horizontal in XML.

```
<EditText
    android:id="@+id/email"
    android:layout_width="200dp"
    android:layout_height="wrap_content"
    android:text="Codes insider"
    android:layout_gravity="center_horizontal"
    android:gravity="start"
    />
```

Below is the output of the above code snippet.If we observe the code the layout gravity is center_horizontal which is why
the layout is positioned center of screen and the gravity is set to start, for which the text is at the starting of the
EditText

text : Text attribute is used to set the text to EditText.

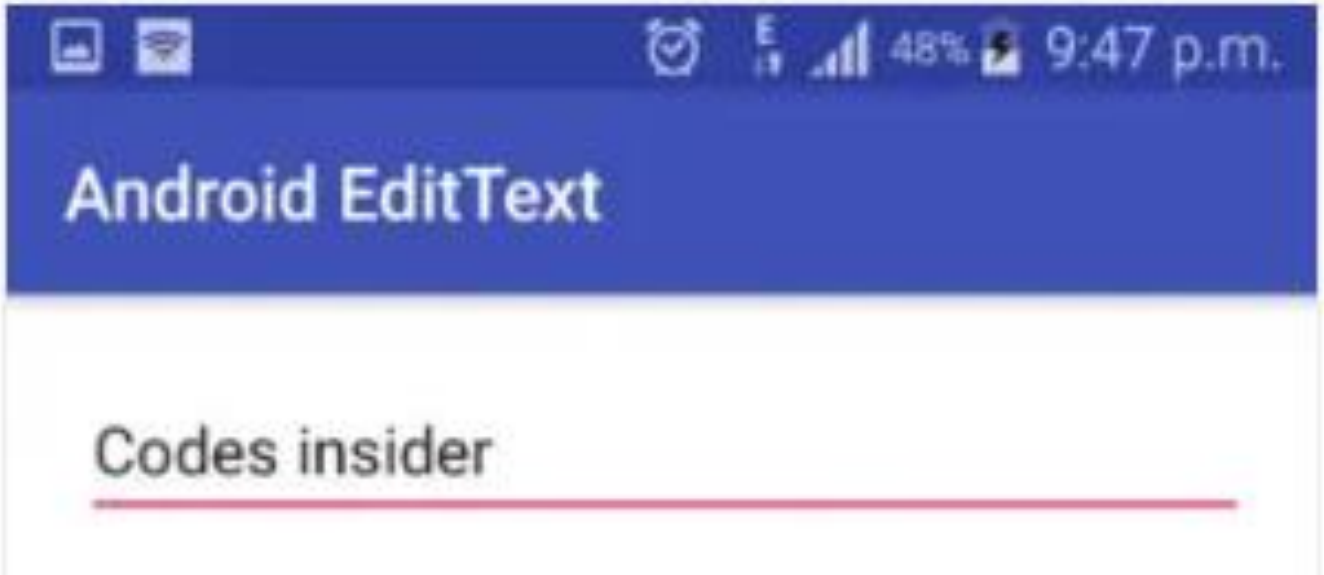
Example code snippet for setting text to a EditText in XML.

```
<EditText
        android:id="@+id/email"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Codes insider"
        />
```

Example code snippet for setting text to EditText in JAVA.

```
EditText email = findViewById(R.id.email);
email.setText("Email address");
```

Output of the above code snippets.



Note: The standard way of using strings, colors and dimentions in android was explained clearly in **using strings, colors, dimens in UI controls** section of this Android UI Controls tutorial.

hint : hint in android is similar to placeholder in html which is used to inform the user what type of data should be entered in the EditText.

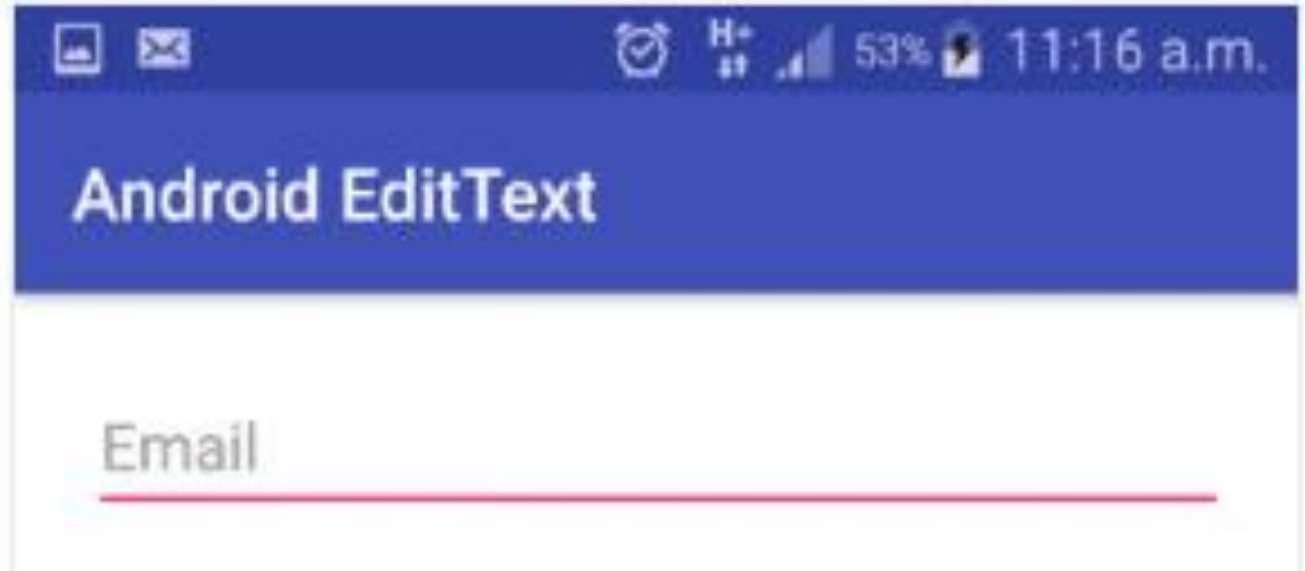
Example code Snippet for setting hint to EditText in XML

```
<EditText
    android:id="@+id/email"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:hint="Email"
    />
```

Example code snippet for setting hint to EdtText in JAVA.

```
EditText email = findViewById(R.id.email);
email.setHint("Email"); //setting hint
```

Output of the above code snippets.



textColorHint : textColorHint is used to set the color to hint text.

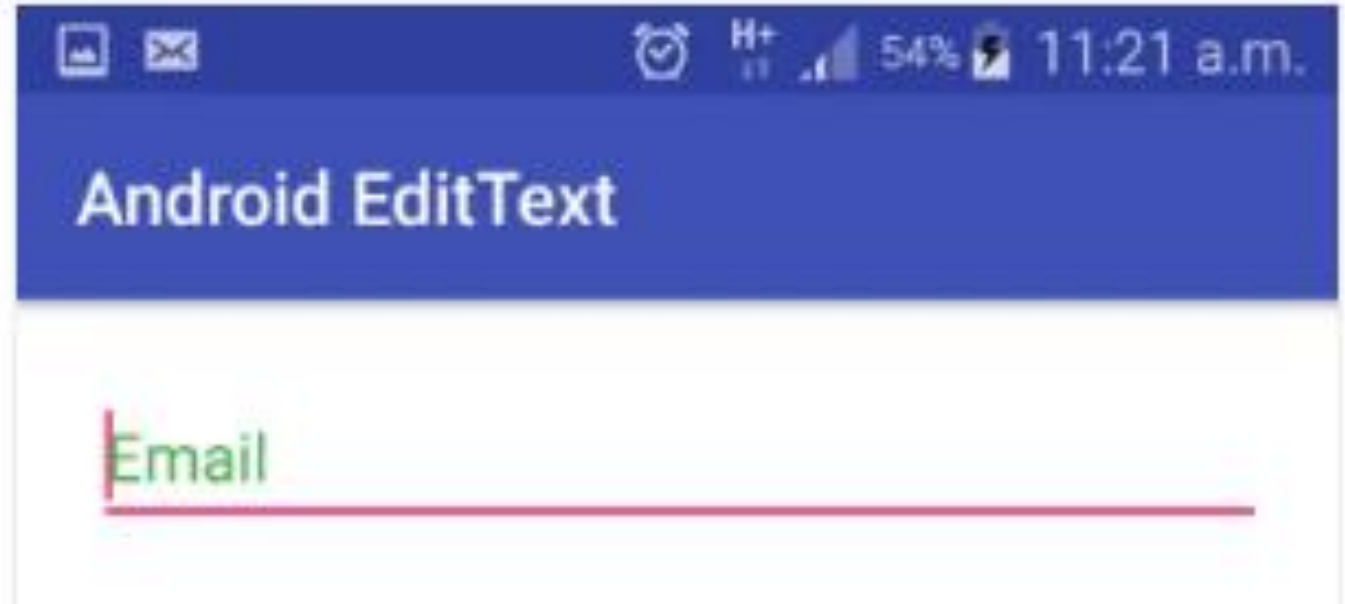
Example code snippet for setting textColorHint to EditText in XML.

```
<EditText
    android:id="@+id/email"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:hint="Email"
    android:textColorHint="@color/green"
    />
```

Example code snippet for setting textColorHint to EditText in JAVA.

```
EditText email = findViewById(R.id.email);
    email.setHint("Email");
    email.setHintTextColor(getResources().getColor(R.color.green)); //setting hint color
```

Output of the above code snippets.



inputType : Every EditText expects a certain type of text input, such as an email address, phone number, or just plain
text. So it’s important that you specify the input type for each EditText in your app so the system displays the
appropriate soft input method like if we set inputType to phone the android system displays number keypad when we
click on that EditText.

Though there are man input types available i’m just listing a few most commonly used input types below

date
phone
number
text
textEmailAddress
textPassword

Example code snippet for setting inputType for EditText in XML.

```
<EditText
    android:id="@+id/email"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:hint="Email"
    android:text="Codes Insider"
    android:inputType="phone"
    android:textColorHint="@color/green"
    />
```

textColor : textColor attribute is used to set the text color.The value for textColor can be “#argb”, “#rgb”, “#rrggbb”, or “#aarrggbb”.

Example code snippet for setting text color of an EditText in XML.

```
<EditText
    android:id="@+id/email"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:hint="Email"
    android:text="Codes insider"
    android:textColor="@color/green"
    />
```

Example code snippet for setting text color of an EditText in java

```
EditText email = findViewById(R.id.email);
email.setHint("Email");
email.setTextColor(getResources().getColor(R.color.green)); //setting text color
```

textSize : textSize attribute is used to set the text size of a EditText whose value can be either in sp(scale-independent pixels) or dp(density independent pixels) like for example 20sp or 20dp

Example code snippet for setting textSize of an EditText in XML.

```
<EditText
        android:id="@+id/email"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:hint="Email"
        android:text="Codes insider"
        android:textColor="@color/black"
        android:textSize="20sp"
        />
```

Example code snippet for setting textSize of an EditText in JAVA.

```
 EditText email = findViewById(R.id.email);
 email.setText("Codes insider");
 email.setTextSize(20);
```

textStyle : textStyle attribute is used to style the text of the EditText.There are three values for a textStyle attribute like normal, bold, italic.By default it is normal.If we want to use more than one value at a time we can use “|” operator. like bold|italic.

Example code snippet for setting text style of a EditText in XML.

```
<EditText
    android:id="@+id/email"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:hint="Email"
    android:text="Codes insider"
    android:textColorHint="@color/green"
    android:textColor="@color/black"
    android:textSize="20sp"
    android:textStyle="bold|italic"
    />
```

Android UI Controls

In android the components which lets the user to interact with the application on the device screen are called **UI controls** or components.These controls are used to design the user interface of an android application.Android provides different kinds of **UI controls** to develop advanced and beautiful applications.

In terms of android, user interface is made of objects of **View** and **View Group** classes.**View** is base class for all the UI controls or components like Text View, Edit Text, Check Box, Image View etc. All these UI controls are objects of **View** class.