Labsheet 2 – Friday 3rd September 2021
Building an Interactive Application
Software Development for Portable Devices

Information about HD TA:

AASHITA DUTTA <h20201030130@hyderabad.bits-pilani.ac.in>

Breakout room link: https://meet.google.com/erq-uxqi-egv

Information about FD TA:

ARUNEEMA DESHMUKH <f20180568@hyderabad.bits-pilani.ac.in>

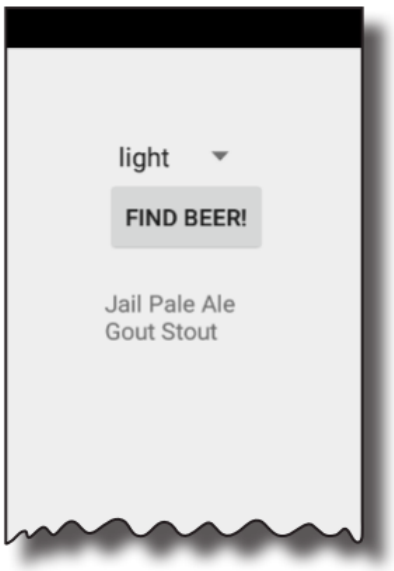
Breakout room link: https://meet.google.com/pvo-osiz-cuh

In this practical you learn how to make apps that are interactive. The apps will perform actions as a response to user input .

1. Create a default activity and layout
2. Update Layout
    a. Adding components with the design editor
    b. Using string resources
3. Connect activity by
    a. Making the button call a method
    b. Adding an onClickFindBeer() method to the activity
4. Write the Logic

App Description:

We'll be creating a Beer Advisor app. In the app, users can select the types of beer they enjoy from a drop down, click a button, and get back a list of tasty beers to try out.



Task 1: Create a new project

1. Just like the app made in the previous lab, select **Empty Activity** as Project Template. After clicking next you can name this activity (FindBeerActivity). [Incase you do not get this option, you can rename the activity later, follow Renaming an Activity]
2. Give a suitable App name (BeerAdvisor) and Package/Domain name.

Task 2: Add the components to the Layout - 15 min

For the "FindBeerActivity" the accompanying layout is “activity_find_beer.xml” present in **app/src/main/res/layout folder**. We will be adding the GUI components in this file.
By default there will be a TextView that displays Hello World! on the screen inside a ConstraintLayout, delete that.
First we will use a **Linear Layout.**

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
   xmlns:android="http://schemas.android.com/apk/res/android"
   xmlns:tools="http://schemas.android.com/tools"
   android:layout_width="match_parent"
   android:layout_height="match_parent"
   android:padding="16dp"
   android:orientation="vertical"
   tools:context=".YourActivityName">

</LinearLayout>
```

1. There are two ways of adding GUI components to the layout: via XML or using the design editor. Try adding a **Button** via the design editor. It can be found under the **Palette Tab,** to add simply drag and drop or right click and **Add to Design.** If you switch to the code editor, you’ll see that adding the button via the design editor has added some lines of code to the file
2. Now add a **Spinner** and a **TextView** either way**.** Try to rearrange the components such that the spinner comes first, then the button and then the textview.

Questions:
Who is the parent when we assign layout height and width to “match_parent”? What is meant by “wrap_content”?
How is android:id different from android:text?
Explore the other available layouts. Refer to the link here and more practise on linear and relative layouts -
https://www.cs.dartmouth.edu/~sergey/cs65/cheatsheets/Layout.pdf

Try it yourself: Use the **Relative Layout** and observe the output.

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
   xmlns:tools="http://schemas.android.com/tools"
   android:layout_width="match_parent"
   android:layout_height="match_parent"
   android:paddingBottom="16dp"
   android:paddingLeft="16dp"
   android:paddingRight="16dp"
   android:paddingTop="16dp"
   tools:context=".MainActivity" >
   <Spinner
       android:id="@+id/color"
       android:layout_width="wrap_content"
       android:layout_height="wrap_content"
       android:layout_alignParentTop="true"
```

```
        android:layout_centerHorizontal="true"
        android:layout_marginTop="37dp" />
    <Button
        android:id="@+id/find_beer"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignLeft="@+id/color"
        android:layout_below="@+id/color"
        android:text="Button" />
    <TextView
        android:id="@+id/brands"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignLeft="@+id/find_beer"
        android:layout_below="@+id/find_beer"
        android:layout_marginTop="18dp"/>

</RelativeLayout>
```

Try it yourself: Explore the other GUI components. See the changes in the code when you Add/Move a component by using the split editor. Also try altering the different attribute values and see the component change.
Here are some examples of view components that you can use.

- **TextView:** To add some text in your application.
- **EditText:** This is used when you want to take some input from the users.
- **ImageView:** To add some image in the application.
- **ProgressBar:** To show the progress to something. For example, the loading screen.
- **Button:** Buttons are used to trigger some action on the click of the button. It can be starting a new activity or something else.
- **ImageButton:** It is used to make a clickable image.
- **CheckBox:** CheckBox is used to select some options out of many available options.
- **DatePicker:** To select some particular date.

Task 3: Use string resources rather than hardcoding the text

At the moment, the GUI components use hardcoded string values for their text properties. it's a good practice to change these to use the strings resource file strings.xml instead. **[Why?]**

1. Open the **app/src/main/res/values/strings.xml file**.
2. Add a new resource called "find_beer" with a value of "Find Beer!" and a new resource named "brands" but don't enter anything for the value.
   ```
   <string name="find_beer">Find Beer!</string>
   <string name="brands"></string>
   ```
3. Now to use this resource in the layout
   For Button- `android:text="@string/find_beer"`

For TextView- `android:text="@string/brands"`

Try it yourself: Run the app at this point. Does it look the way you expected it to?

Questions

1. **Why is strings.xml useful?**
2. **What are the types of string resources available?**
3. **What is the significance of @+id in xml file?**

Task 4: Add values to the spinner

A spinner provides a drop-down list of values. It allows you to choose a single value from a set of values.
We can give the spinner a list of values the same way that we set the text on the button or any other view, by using a resource. We add an array of String values, and get the spinner to reference it.

```
<string-array name="beer_colors">
    <item>light</item>
    <item>amber</item>
    <item>brown</item>
    <item>dark</item>
</string-array>
```

And in the Layout for spinner `android:entries="@array/beer_colors"`

Try it yourself: Run the app again.

Task 5: Add method to call in the activity when the button is clicked. - 15 min

When an user clicks on the button, we want the app to perform some action. To get a button to call a method in the activity when it's clicked, we need to change the **layout file** activity_find_beer.xml.

- We'll specify which method in the activity will get called when the button is clicked.

All you need to do is add an android:onClick attribute to the element (Button), and give it the name of the method you want to call
Eg: `android:onClick="onClickFindBeer"`

We need to change the **activity file** FindBeerActivity.java too.

- We need to write the method that gets called.

We added an onClick attribute to the button in our layout and gave it a value of onClickFindBeer. We need to add this method to our activity so it will be called when the button gets clicked. This will enable the activity to respond when the user touches a button in the user interface.
The method needs to take the following form:

```
public void onClickFindBeer(View view) {}
```

The View parameter refers to the GUI component that triggers the method (in this case, the button)

```
package com.example.beeradvisor;

import androidx.appcompat.app.AppCompatActivity;

import android.os.Bundle;
```

```
import android.view.View; // import this

public class FindBeerActivity extends AppCompatActivity {

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

   //Call when the user clicks the button
   public void onClickFindBeer(View view){
   }

}
```

Questions

1. **Why is the View class object passed as an argument to onClickFindBeer function?**
2. **What are event listeners? What are the different types of event listeners?**

Task 6: Write the Logic

We need to get our app to display a selection of different beers [a list] that match the beer type the user has selected.

1. We need to get a reference to both the spinner and text view GUI components in the layout. This will allow us to retrieve the value of the chosen beer type from the spinner, and display text in the text view. We can get a handle for the GUI components using a method called findViewById(). This method provides you with a Java version of the GUI component. This means that you can get and set properties in the GUI component using the Java class methods.

   ```
   TextView brands = (TextView) findViewById(R.id.brands);
   ```

   When this line of code gets called, it creates a TextView object called brands. You are then able to call methods on this TextView object. Eg brands.setText("Gottle of geer");
   Similarly,

   ```
   Spinner color = (Spinner) findViewById(R.id.color);
   ```

   To retrieve the currently selected item in the spinner [the item is a generic Java Object], and convert it to a String:

   ```
   String.valueof(color.getSelectedItem())
   ```

   Try it yourself: Retrieve the value from the spinner and display it on the TextView. Run the app to see if it works. You will need to add import android.widget.Spinner; and import android.widget.TextView;

2. Let's write a Java class called BeerExpert. It should have a method, getBrands(), that takes a preferred beer color (as a String), and return a List of recommended beers.
   To do so, Right-Click on com.example.beeradvisor[this will be your package name] package in the **app/src/main/java folder,** and go to File→New...→Java Class. Name it as BeerExpert. A new class will be created in the package.

Try it yourself: If you are familiar with Java write the method called getBrands(). It should take one String parameter called color and return a List <String>. If the color is “amber” the list should have "Jack Amber" and “Red Moose”. For any other color return "Jail Pale Ale" and "Gout Stout".

```
package com.example.beeradvisor;

import java.util.ArrayList;
import java.util.List;

public class BeerExpert {
    List<String> getBrands(String color) {
        List<String> brands = new ArrayList<>();
        if (color.equals("amber")) {
            brands.add("Jack Amber");
            brands.add("Red Moose");
        } else {
            brands.add("Jail Pale Ale");
            brands.add("Gout Stout");
        }
        return brands;
    }
}
```

3. To use this class in FindBeerActivity.java include the next line inside FindBeerActivity

```
private BeerExpert expert = new BeerExpert();
```

Try it yourself: Get the recommendations from BeerExpert based on a color and store then in a List. Use a StringBuilder to append each word and a new line with it. Display the result in the TextView

```
//Call when the button gets clicked
public void onClickFindBeer(View view) {
    //Get a reference to the TextView
    TextView brands = (TextView) findViewById(R.id.brands);

    //Get a reference to the Spinner
    Spinner color = (Spinner) findViewById(R.id.color);

    //Get the selected item in the Spinner
    String beerType = String.valueOf(color.getSelectedItem());

    //Get recommendations from the BeerExpert class
    List<String> brandsList = expert.getBrands(beerType);
    StringBuilder brandsFormatted = new StringBuilder();
    for (String brand : brandsList) {
        brandsFormatted.append(brand).append('\n');
    }

    //Display the beers
    brands.setText(brandsFormatted);
}
```

Questions

1. **Why we need to type-cast TextView or Spinner?**
2. **How activity find reference to GUI components like TextView or spinner?**
3. **Why we need custom Java class?**

References:
https://github.com/dogriffiths/HeadFirstAndroid2ndEdition/tree/master/chapter02/BeerAdviser

Additional Exercises (To be completed as home assignment)

Design a simple calculator application having:

1. numeric buttons(0-9), operator buttons(addition(+), multiplication(*), subtraction(-), division(/)), clear button and equal(=) button, decimal(.) button.
2. One EditText widget.
3. Connect layout with the activity class.
4. Write logic behind each operator button in activity class.
5. Display the results of calculations.