

Answer for Homework

Android Bundle

- Android Bundle is used to pass data between activities.
- The values that are to be passed are mapped to String keys which are later used in the next activity to retrieve the values.

Example 01

Create an application to pass data from one activity to another by using Bundle Object.

- Create a new project.
- Create two Activities.
- Insert two EditText and a Button to the first activity.
- Insert two TextViews to the second activity. Constraint them and set IDs.

The image shows two side-by-side activity layouts. The left layout is for the first activity and contains two text input fields. The first field is labeled 'Name' and the second is labeled 'Address'. Below these fields is a button labeled 'SUBMIT'. The right layout is for the second activity and contains two TextViews. The top TextView is labeled 'TextView' and the bottom one is also labeled 'TextView' and has a blue border around it.

- Create references for them as follows in the Main Activity.

```
public class MainActivity extends AppCompatActivity {

    EditText name;
    EditText address;
    Button login;
    Bundle bundle;
    Intent check;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        login= (Button) findViewById(R.id.btnLog);
        name= (EditText) findViewById(R.id.editOne);
        address= (EditText) findViewById(R.id.editTwo);
        check = new Intent( packageContext: this, Activity2.class);
        bundle= new Bundle();
    }
}
```

Bundle Object

Intent Object

- In here you have to create an **Intent Object** as we did in the Extras method.
- As well as you have to create a **bundle Object**.
- Then for the **OnClick** method of the Button insert the following code.

```
login.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        bundle.putString("NAME", name.getText().toString());
        bundle.putString("ADDRESS", address.getText().toString());
        check.putExtras(bundle);
        startActivity(check);
    }
});
```

- In here since both address and name are String vales you can use **putString()** method to pass the data to Bundle object that you have created. There are several methods such as putInt(), putBoolean(), puLong(), putChar() etc...

```
putString(String key, String value)
```

```
bundle.putString("NAME", name.getText().toString());
```

- Here we are getting the data which is entered to the **EditText** By the user.

- By using **putExtras()** method you can pass the Bundle Object to the **Intent Object** that you have created.
- In the Java file of the second activity, insert two references for the **TextViews** as follows.

```
public class Activity2 extends AppCompatActivity {
    TextView name;
    TextView address;

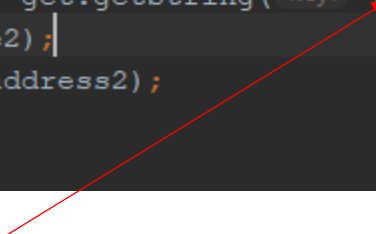
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_2);
        name= (TextView) findViewById(R.id.msg1);
        address= (TextView) findViewById(R.id.msg2);
    }
}
```

- Create an Intent Object and assign **getIntent()** method to it.
- The Intent object of the Main activity can be retrieved by using this **getIntent()** method.
- Then create a Bundle Object and by using **getExtras()** method you can fetch the data that have been stored in the Bundle object of the Main Activity.

```
Intent content= getIntent();
Bundle get = content.getExtras();
```

- Then you can create two String references, and by using **getString()** method of the Bundle Object you can access the data that have been passed from the Main activity.

```
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_2);
    name= (TextView) findViewById(R.id.msg1);
    address= (TextView) findViewById(R.id.msg2);
    Intent content= getIntent();
    Bundle get = content.getExtras();
    String name2= get.getString( key: "NAME");
    String address2 = get.getString( key: "ADDRESS");
    name.setText(name2);
    address.setText(address2);
}
```



- In here make sure to use the **Keys** that you entered in the Main activity.

Activity Life Cycle

In general, activity lifecycle has seven callback methods:

onCreate() – Called when the activity is first created

onStart() – Called just after it's creation or by restart method after onStop(). Here Activity start becoming visible to user

onResume() – Called when Activity is visible to user and user can interact with it

onPause() – Called when Activity content is not visible because user resume previous activity

onStop() – Called when activity is not visible to user because some other activity takes place of it

onRestart() – Called when user comes on screen or resume the activity which was stopped

onDestroy – Called when Activity is not in background

Example 02

- Create a new project.
- Initialize a static String variable with the name of the underlying class using getSimpleName() method. In our case MAIN_ACTIVITY_TAG is the name of the String variable which store class name HomeActivity.

```
public class MainActivity extends AppCompatActivity {  
    |  
    private static final String MAIN_ACTIVITY_TAG = MainActivity.class.getSimpleName();  
}
```

- Create a new method which will print message in Logcat.

```
private void showLog(String text){  
    |  
    Log.d(MAIN_ACTIVITY_TAG, text);  
    |  
}
```

- Now we will override all activity lifecycle method in Android and use showLog() method which we creating for printing message in Logcat.

```

@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);

    showLog( text: "Activity Created");
}

@Override

protected void onRestart() {

    super.onRestart();//call to restart after onStop

    showLog( text: "Activity restarted");
}

@Override

protected void onStart() {

    super.onStart();//soon be visible

    showLog( text: "Activity started");
}

@Override

protected void onResume() {

    super.onResume();//visible

    showLog( text: "Activity resumed");
}
}

```

```

@Override

protected void onPause() {

    super.onPause();//invisible

    showLog( text: "Activity paused");
}

@Override

protected void onStop() {

    super.onStop();

    showLog( text: "Activity stopped");
}

@Override

protected void onDestroy() {

    super.onDestroy();

    showLog( text: "Activity is being destroyed");
}
}

```

- When you will run the above program you will notice a blank white screen will open up in Emulator. *You might be wondering where is default Hello world screen. Actually we have removed it by overriding onCreate() method.*
- Now go to Logcat present inside Android Monitor: Scroll up and you will notice three methods which were called: Activity Created, Activity started and Activity resumed.

```

2021-02-18 13:42:08.386 7264-7264/com.example.activitylifecycle D/MainActivity: Activity Created
2021-02-18 13:42:08.485 7264-7264/com.example.activitylifecycle D/MainActivity: Activity started
2021-02-18 13:42:08.791 7264-7264/com.example.activitylifecycle D/MainActivity: Activity resumed

```

- first onCreate() method was called when activity was created.
- second onStart() method was called when activity start becoming visible to user.
- Finally onResume() method was called when activity is visible to user and user can interact with it.

- Now press the back button on the Emulator and exit the App:

```
2021-02-18 13:44:48.459 7264-7264/com.example.activitylifecycle D/MainActivity: Activity paused
2021-02-18 13:44:51.028 7264-7313/com.example.activitylifecycle D/EGL_emulation: eglMakeCurrent: 0xa363f560: ver 2 0 (tinfo 0xa19116c0)
2021-02-18 13:44:51.275 7264-7264/com.example.activitylifecycle D/MainActivity: Activity stopped
2021-02-18 13:44:51.276 7264-7264/com.example.activitylifecycle D/MainActivity: Activity is being destroyed
```

- onPause() method was called when user resume previous activity
- onStop() method was called when activity is not visible to user
- Last onDestroy() method was called when Activity is not in background

You can try different status of the app and examine how the callback methods are working.

Toast in Android

- A toast provides simple feedback about an operation in a small popup.
- It only fills the amount of space required for the message and the current activity remains visible and interactive. Toasts automatically disappear after a timeout.

Toast class

Toast class is used to show notification for a particular interval of time. After some time it disappears. It doesn't block the user interaction.

Constants of Toast class

There are only 2 constants of Toast class which are given below.

Constant	Description
public static final int LENGTH_LONG	displays view for the long duration of time.
public static final int LENGTH_SHORT	displays view for the short duration of time.

Example 03

- Create a new project.
- Insert a Button. Constraint it. Assign an ID.
- Then create a reference for it.

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

- Then you have to create a method to show the toast message for the Onclick() event of the button.

In XML:

```
<Button  
    android:id="@+id/btnClick"  
    android:layout_width="wrap_content"  
    android:layout_height="wrap_content"  
    android:layout_marginStart="8dp"  
    android:layout_marginTop="8dp"  
    android:layout_marginEnd="8dp"  
    android:layout_marginBottom="8dp"  
    android:text="Click"  
    app:layout_constraintBottom_toBottomOf="parent"  
    app:layout_constraintEnd_toEndOf="parent"  
    app:layout_constraintHorizontal_bias="0.498"  
    app:layout_constraintStart_toStartOf="parent"  
    app:layout_constraintTop_toTopOf="parent"  
    app:layout_constraintVertical_bias="0.637"  
    android:onClick="showMsg"  
>
```

In Main Activity:

- Inside the method that you have created create a **Toast** reference and by using **Toast.makeText() method** you can enter the message that you want to show.
- There you have to pass three parameters.
 1. Application Context - **this** – is used to specify to take the context of the same activity.
 2. The message that you want to show.
 3. The duration of the message
- By using show() method you can display the message.

```
public void showMsg(View view) {  
    Toast msg= Toast.makeText( context: this, text: "The button was clicked", Toast.LENGTH_LONG);  
    msg.show();  
}
```

Android - Shared Preferences

- Android provides many ways of storing data of an application. One of this way is called Shared Preferences. Shared Preferences allow you to save and retrieve data in the form of **key,value** pair.
- In order to use shared preferences, you have to call a method **getSharedPreferences()** that returns a **SharedPreferences** instance pointing to the file that contains the values of preferences.

Example 03

- Create a new project.
- Insert two **EditText** and two **Buttons**. Constraint them and assign IDs.
- Create references for them.

```
public class MainActivity extends AppCompatActivity {  
    EditText name;  
    EditText town;  
    Button read;  
    Button write;  
  
    @Override  
    protected void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        setContentView(R.layout.activity_main);  
        name= (EditText) findViewById(R.id.Name);  
        town= (EditText) findViewById(R.id.Town);  
        read= (Button) findViewById(R.id.read);  
        write= (Button) findViewById(R.id.write);  
    }  
}
```


- In here **Write** Button is used to store the data entered by the user and **Read** Button is used to show the data which has been stored.

Write Button

- For the **onClick()** event of the Write button first create a reference of **SharedPreferences** and you have to call a method **getSharedPreferences()** that returns a SharedPreferences instance pointing to the file that contains the values of preferences.

```
SharedPreferences data1= getSharedPreferences( name: "details",MODE_PRIVATE);
```

The key

The Mode

Examples of Modes:

	MODE_APPEND This will append the new preferences with the already existing preferences
	MODE_PRIVATE By setting this mode, the file can only be accessed using calling application
	MODE_WORLD_READABLE This mode allow other application to read the preferences
	MODE_WORLD_WRITEABLE This mode allow other application to write the preferences

- You can save something in the **sharedpreferences** by using **SharedPreferences.Editor** class.
- You will call the edit method of **SharedPreference** instance and will receive it in an editor object.

```
write.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        SharedPreferences data1= getSharedPreferences( name: "details",MODE_PRIVATE);
        SharedPreferences.Editor editor=data1.edit();
        editor.putString("NAME", name.getText().toString());
        editor.putString("TOWN", town.getText().toString());
        editor.commit();
        Toast.makeText(getApplicationContext(), text: "Data stored",Toast.LENGTH_SHORT).show();
    }
});
```

- Apart from the **putString** method, there are methods available in the editor class you can search and find about them.
- In here we have used a **Toast** to see whether data has been stored to the shared preference correctly.

Read Button

- For the **onClick()** event of the Read button too first create a reference of **SharedPreferences** and call **getSharedPreferences()** method as in the previous one. In here make sure to use the same name as the previous shared preference (**name:"details"**).
- In here to retrieve the data that has been stored in the **SharedPreference** you can use **getString()** method. And store them in String variables.
- Make sure to use the same keys that used in **putString()** method.

```
SharedPreferences data2= getSharedPreferences( name: "details",MODE_PRIVATE);
String name=data2.getString( key: "NAME", defValue: "name not stored");
String town=data2.getString( key: "TOWN", defValue: "town not stored");
```

- Then create a Toast to show the data that has been entered by user.

```
read.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        SharedPreferences data2= getSharedPreferences( name: "details",MODE_PRIVATE);
        String name=data2.getString( key: "NAME", defValue: "name not stored");
        String town=data2.getString( key: "TOWN", defValue: "town not stored");
        Toast.makeText(getApplicationContext(), text: "Your name : "+name+" City: "+town,Toast.LENGTH_LONG).show();
    }
});
```

- You can enter data for the two fields (Name, City) and click **Write** Button to store the data in the Shared Preference.
- After that click **Read** Button, you can see the data that you have entered in a Toast message.
- Next you can close/back the app and start it again.
- Just click **Read** Button.
- Then you will be able to see the data that have been entered by you is still showing in your toast message.
- Hence, you can understand by using shared preferences if the app is closed or paused you can access the data that have been entered previously. They are not destroyed.

Exercise 01 – Bundle Object

Create the following application.

- The app should contain three activities.

Main Activity

- The Main activity should contain two EditText as **User Name** and **Password**, one button as **Log In**.
- If the user name is “**user**” and password is “**root**” the user should be directed to the second activity.

Second Activity

- Should contain four EditText as Name, Index Number, Department and Faculty and one button as OK.
- When the user enters data for the 4 fields and clicks OK button he should be directed to the third activity.

Third Activity

- That should display the details which are entered by the user in **four Text Fields**.

Exercise 02 – Toast

Create an application to take First Name and Last Name from the user and when user click “**Show**” **Button** it should display **Full Name** (First Name + Last Name) in a Toast message.

Exercise 03 – Shared Preferences

Create the following application

- The app should contain four activities (Main, Sign Up, Sign In, Last).

The Main Activity

- Should contain two Buttons as **Sign In** and **Sign Up**.
- When the user clicks **Sign Up** Button he should be directed to Sign Up Activity.

Sign Up Activity

- Should contain four EditText as Name, City, User Name and Password and two buttons as Sign Up and Back.
- Those entered details should be stored in a **Shared Preference**.
- When the user clicks Sign up Button a toast message named “**You have signed up successfully**” should be displayed.
- And when he clicks the Back Button he should be directed to the home page (Main Activity).
- Then the user should be able to log In to the application by clicking **Sign In** Button.

Sign In Activity

- Should contain two **EditText** to take User Name and Password.
- In here you have to match the **user entered User Name and Password** with the **User Name and Password that the user entered during Sign Up process**. (you have stored those in a SharedPreferences).
- If the User Name and Password are correct user should be directed to the Last Activity.

Last Activity

- Should contain two TextView and they should display the Name and the City entered by the user during the Sign Up process. (which is stored in the shared preference).