**ANDROID APPLICATION**

**ASSIGNMENT OF WEEK 2**

1. Discuss about life cycle of activity in student Registration process.

[The **student registration process** involves several stages, and the **activity lifecycle** plays a crucial role in managing these stages1](https://www.geeksforgeeks.org/activity-lifecycle-in-android-with-demo-app/). [The activity lifecycle in Android consists of four stages: **created**, **started**, **resumed**, and **destroyed**1](https://www.geeksforgeeks.org/activity-lifecycle-in-android-with-demo-app/). Each stage represents a different state of the activity.

[To understand the life cycle of an activity in the student registration process, let’s consider an example of a student registration system2](https://venngage.com/blog/activity-diagram/). [The following diagram illustrates the series of actions performed by the student, the student registration system, and the registration division to complete the student registration process](https://www.geeksforgeeks.org/activity-lifecycle-in-android-with-demo-app/)[2](https://venngage.com/blog/activity-diagram/):

graph LR

A[Student] --> B(Student Registration System)

B --> C{Registration Division}

C --> D[Complete Registration]

The life cycle of an activity can be summarized as follows:

1. **Created**: The activity is created but not yet visible to the user.
2. **Started**: The activity becomes visible to the user but not yet interactive.
3. **Resumed**: The activity is both visible and interactive. This is the state where the student interacts with the registration system to provide necessary information.
4. **Destroyed**: The activity is no longer needed and is removed from memory.

[Throughout the student registration process, various activities are created, started, resumed, and eventually destroyed as students progress through each step](https://www.geeksforgeeks.org/activity-lifecycle-in-android-with-demo-app/)[1](https://www.geeksforgeeks.org/activity-lifecycle-in-android-with-demo-app/). [These activities are managed by an activity stack, ensuring that previous activities remain below the current one](https://www.geeksforgeeks.org/activity-lifecycle-in-android-with-demo-app/)[1](https://www.geeksforgeeks.org/activity-lifecycle-in-android-with-demo-app/).

Please note that this explanation is based on general Android activity lifecycle principles and may vary depending on specific implementation details.

2. Create a android application for course registration using multiple activities.

Here’s a simple example of an Android application for course registration using multiple activities:

// MainActivity.java

public class MainActivity extends AppCompatActivity {

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_main);

// Start the registration process by launching the RegistrationActivity

Intent intent = new Intent(this, RegistrationActivity.class);

startActivity(intent);

}

}

// RegistrationActivity.java

public class RegistrationActivity extends AppCompatActivity {

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_registration);

// Perform the registration process here

// ...

// Once the registration is complete, launch the ConfirmationActivity

Intent intent = new Intent(this, ConfirmationActivity.class);

startActivity(intent);

}

}

// ConfirmationActivity.java

public class ConfirmationActivity extends AppCompatActivity {

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_confirmation);

// Display a confirmation message to the user

Toast.makeText(this, "Registration successful!", Toast.LENGTH\_SHORT).show();

}

}

In this example, we have three activities: MainActivity, RegistrationActivity, and ConfirmationActivity. The MainActivity is the entry point of the application and launches the RegistrationActivity when it starts. The RegistrationActivity performs the registration process, and once it’s complete, it launches the ConfirmationActivity. The ConfirmationActivity displays a toast message to confirm that the registration was successful.

Please note that this is a simplified example to demonstrate the concept of using multiple activities for course registration. In a real-world scenario, you would need to implement additional functionality and handle user input/validation.

3. Create an android application for simple basic calculator.

// MainActivity.java

public class MainActivity extends AppCompatActivity {

private EditText inputField;

private Button addButton;

private Button subtractButton;

private Button multiplyButton;

private Button divideButton;

private Button calculateButton;

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_main);

// Initialize UI elements

inputField = findViewById(R.id.input\_field);

addButton = findViewById(R.id.add\_button);

subtractButton = findViewById(R.id.subtract\_button);

multiplyButton = findViewById(R.id.multiply\_button);

divideButton = findViewById(R.id.divide\_button);

calculateButton = findViewById(R.id.calculate\_button);

// Set click listeners for the buttons

addButton.setOnClickListener(new View.OnClickListener() {

@Override

public void onClick(View v) {

performOperation("+");

}

});

subtractButton.setOnClickListener(new View.OnClickListener() {

@Override

public void onClick(View v) {

performOperation("-");

}

});

multiplyButton.setOnClickListener(new View.OnClickListener() {

@Override

public void onClick(View v) {

performOperation("\*");

}

});

divideButton.setOnClickListener(new View.OnClickListener() {

@Override

public void onClick(View v) {

performOperation("/");

}

});

calculateButton.setOnClickListener(new View.OnClickListener() {

@Override

public void onClick(View v) {

calculateResult();

}

});

}

private void performOperation(String operator) {

String input = inputField.getText().toString();

// Perform the operation based on the selected operator

// ...

// Update the input field with the result

// ...

}

private void calculateResult() {

String input = inputField.getText().toString();

// Calculate the result based on the input expression

// ...

// Display the result in the input field

// ...

}

}