Name: **John Bernard A. Garcia** Date: **22/08/2025**
Section: **BSIT - 3B** Subject: **IT Professional Elective 1**

Assignment #1

Research about the Activity lifecycle, get information, code on how to use it [onCreate] [onStart] , [onResume] , [onPause] , [onStop] , [onDestroy].

What is the Activity lifecycle?

As a user navigates through, out of, and back to your app, the Activity instances in your app transition through different states in their lifecycle. The Activity class provides a number of callbacks that let the activity know when a state changes or that the system is creating, stopping, or resuming an activity or destroying the process the activity resides in

To navigate transitions between stages of the activity lifecycle, the Activity class provides a core set of six callbacks: onCreate(), onStart(), onResume(), onPause(), onStop(), and onDestroy(). The system invokes each of these callbacks as the activity enters a new state.

[onCreate]

- You must implement this callback, which fires when the system first creates the activity. On activity creation, the activity enters the Created state. In the onCreate() method, perform basic application startup logic that happens 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. This method receives the parameter savedInstanceState, which is a Bundle object containing the activity's previously saved state. If the activity has never existed before, the value of the Bundle object is null.

  If you have a lifecycle-aware component that is hooked up to the lifecycle of your activity, it receives the ON_CREATE event. The method annotated with @OnLifecycleEvent is called so your lifecycle-aware component can perform any setup code it needs for the created state.

Sample Code:

```
package com.example.lifecycleexample;

import android.os.Bundle;
import android.util.Log;
import androidx.appcompat.app.AppCompatActivity;

public class MainActivity extends AppCompatActivity {

    private static final String TAG = "LifecycleExample";

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main); // sets the layout for the activity

        // Code to run when the activity is created
        Log.d(TAG, "onCreate called!");

        // Example: initialize UI components, variables, etc.
    }
}
```

[onStart]

- When the activity enters the Started state, the system invokes onStart(). This call makes the activity visible to the user as the app prepares for the activity to enter the foreground and become interactive. For example, this method is where the code that maintains the UI is initialized.

  When the activity moves to the Started state, any lifecycle-aware component tied to the activity's lifecycle receives the ON_START event.

  The onStart() method completes quickly and, as with the Created state, the activity does not remain in the Started state. Once this callback finishes, the activity enters the *Resumed* state and the system invokes the onResume() method.

Sample Code:

```
package com.example.lifecycleexample;

import android.os.Bundle;
import android.util.Log;
import androidx.appcompat.app.AppCompatActivity;

 public class MainActivity extends AppCompatActivity {

 private static final String TAG = "LifecycleExample";

 @Override
 protected void onStart() {
  super.onStart();
  // Code to run when the activity becomes visible to the user
  Log.d(TAG, "onStart called!");

  // You can start animations, refresh UI, or resume resources here
 }
}
```

[onResume]

- When the activity enters the Resumed state, it comes to the foreground, and the system invokes the onResume() callback. This is the state in which the app interacts with the user. The app stays in this state until something happens to take focus away from the app, such as the device receiving a phone call, the user navigating to another activity, or the device screen turning off. And also this is where the lifecycle components can enable any functionality that needs to run while the component is visible and in the foreground, such as starting a camera preview.

  Sample Code:

```
package com.example.lifecycleexample;

import android.os.Bundle;
import android.util.Log;
import androidx.appcompat.app.AppCompatActivity;

public class MainActivity extends AppCompatActivity {

private static final String TAG = "LifecycleExample";

@Override
protected void onResume() {
super.onResume();
// Code to run when the activity has become interactive
```

```
        Log.d(TAG, "onResume called!");

        // You can restart animations, refresh data, or resume tasks here
    }
}
```

[onPause]

- When an activity moves to the Paused state, any lifecycle-aware component tied to the activity's lifecycle receives the ON_PAUSE event. This is where the lifecycle components can stop any functionality that does not need to run while the component is not in the foreground, such as stopping a camera preview.

  Use the onPause() method to pause or adjust operations that can't continue, or might continue in moderation, while the Activity is in the Paused state, and that you expect to resume shortly.

  You can also use the onPause() method to release system resources, handles to sensors (like GPS), or any resources that affect battery life while your activity is paused and the user does not need them.

  Sample Code:

```
package com.example.lifecycleexample;

import android.os.Bundle;
import android.util.Log;
import androidx.appcompat.app.AppCompatActivity;

public class MainActivity extends AppCompatActivity {

private static final String TAG = "LifecycleExample";

@Override
protected void onPause() {
super.onPause();
// Code to run when the activity is about to go into the background
 Log.d(TAG, "onPause called!");

 // You can pause animations, save data, or release resources here
    }
}
```

[onStop]

- When your activity is no longer visible to the user, it enters the *Stopped* state, and the system invokes the onStop() callback. This can occur when a newly launched activity covers the entire screen. The system also calls onStop() when the activity finishes running and is about to be terminated. And this is where the lifecycle components can stop any functionality that does not need to run while the component is not visible on the screen.

  In the onStop() method, release or adjust resources that are not needed while the app is not visible to the user. For example, your app might pause animations or switch from fine-grained to coarse-grained location updates. Using onStop() instead of onPause() means that UI-related work continues, even when the user is viewing your activity in multi-window mode.

  Sample Code:

```
package com.example.lifecycleexample;

import android.os.Bundle;
import android.util.Log;
import androidx.appcompat.app.AppCompatActivity;

public class MainActivity extends AppCompatActivity {

    private static final String TAG = "LifecycleExample";

    @Override
    protected void onStop() {
        super.onStop();
        // Code to run when the activity is no longer visible to the user
        Log.d(TAG, "onStop called!");

        // You can release resources, stop animations, or save data here
    }
}
```

[onDestroy]

- When the activity moves to the destroyed state, any lifecycle-aware component tied to the activity's lifecycle receives the ON_DESTROY event. This is where the lifecycle components can clean up anything they need to before the Activity is destroyed.

Sample Code:

```
package com.example.lifecycleexample;

import android.os.Bundle;
import android.util.Log;
import androidx.appcompat.app.AppCompatActivity;

public class MainActivity extends AppCompatActivity {

private static final String TAG = "LifecycleExample";

@Override
protected void onDestroy() {
super.onDestroy();
// Code to run when the activity is about to be destroyed
Log.d(TAG, "onDestroy called!");

// You can clean up resources, stop threads, or close database connections here
  }
}
```