Internals-03

Shakeel Hameed. S
1KN18CS097
CSE 'A' Sec
4th Sem,

OOC

| AWT | Swing |
|---|---|
| * Java AWT is an API to develop GUI applications in Java. | * Swing is a part of Java foundation classes and is used to create various applications. |
| * The components of Java AWT are heavy weighted. | * The components of Java Swing are light weighted. |
| * Java AWT has comparatively less functionality as compared to Swing | * Swing has more functionality as compared to AWT. |
| * MVC pattern is not supported by AWT | * MVC pattern is supported by Swing. |
| * The execution time of AWT is more than Swing. | * The execution time by Swing is less than AWT. |

JFrame :-

The javax.swing.JFrame class is a type of container which inherits the java.awt.Frame class. JFrame works

like the main window here where components like labels buttons, textfields are added to create a GUI.

Unlike Frame, JFrame has the option to hide or close the window with the help of setDefaultCloseOperation(int) method

JButton:

It is class that provides functionality of a button. A JButton is the Swing Compont of a button in AWT. It is used to provide an interface equivalent of a common button.

JButton has 3 constructors:

JButton(Icon ic)
JButton(String str)
JButton(String str, Icon ic)

(35) Key features of Swing are:

Platform independent: it is platform-independent, the Swing components that are used to build the system program are not platform specific. It can be used on any platform and anywhere.

customizable: Swing controls can be easily customized. It can be changed, the visual appearance of the Swing component application is independent of its internal representation.

manageable: Swing manages the programs, widgets and stores at specific location.

(9) Configurable: Swing components can be easily configurable. It can be changed.

Lightweighted: Swing components are lightweight, which helps in creating the UI lighter. The Swing components allows it to plug into the OS window interface framework that includes the mapping for screens or devices and other user interactions like keypress and mouse movements.

(10) The thread class is a thread of execution in a program. The java virtual machine allows an application to have multiple threads of execution running concurrently.

Creating thread:

Implementing Runnable: The easiest way to create a class that implements the Runnable interface. We can construct a thread on any object that implements Runnable. To implement Runnable, a class need only implement a single method called run, which is declared as

public void run()

run() establishes the entry point for another, concurrent thread of execution within your program. This thread will end when run() returns.

Thread defines several constructors.

Thread(Runnable threadOb, String threadName)

In this constructor, threadOb is an instance of a class that implements the Runnable interface.

This defines where execution of the thread will begin. The name of the new thread is specified by threadName. After the new thread is created, it will not start running until you call its start() method, which is declared within thread.

## isAlive()

The isAlive() method of thread class tests if the thread is Alive. A thread is considered alive when the start() method of thread class has been called and the thread is not yet dead.

Syntax:

```
public final boolean isAlive()
```

Eg:

```
public class JavaAliveExp extends Thread {
Public void run(){
    try {
        Thread.sleep(300);
        System.out.println(" is run() method is Alive()
        +Thread.currentThread().isAlive());
        }
    Catch (InterruptException ie) {
        }
}
public static void main (String[] args){
    JavaisAliveExp t1 = new JavaisAliveExp();
    System.out.println("before starting thread isAlive:"+t1.
                                                    isAlive());
```

```
t1.start();
System.out.println("after starting thread is alive :"+
        t1.isAlive());
    }
}
```

**Join():**

The join() method waits for a thread to die. in other words, it causes the currently running threads to stop executing until the thread it joins with complete its task.

**Syntax:**

```
public void join() throws InterruptedException.
public void join(long milliseconds) throws InterruptedExce
                                          -ption.
```

**Eg:**

```
class TestJoinMethod. extends Thread{
    public void run(){
        for(int i=1; i<5; i++){
            try{
                Thread.sleep(500);
            }
            catch(Exception e){
                {
                System.out.println(e);
                }
            System.out.println(i);
        }
    }
```

```
public static void main (String args[])
{
    TestJoinMethod t1 = new TestJoinMethod();
    TestJoinMethod t2 = new TestJoinMethod();
    TestJoinMethod t3 = new TestJoinMethod();
    t1.start();
    try{
        t1.join();
    }
    Catch(InterruptedException e)
    {
        System.out.println(e);
    }
    t2.start();
    t3.start();
}
}
```

(11b) Synchronization is a process of handling resources accessibility by multiple thread requests. The main Purpose of Synchronization is to avoid thread interference. At times when more than one thread try to access a shared resources, we need to Ensure that resources will be used by only one thread at a time. The process by which this is achieved is called Synchronization. The keyword synchronization in java creates a block of code referred to as critical section.

General Syntax:- synchronized (object)
{

}

We need Synchronization because, if we do not use Synchronization and let 2 or more threads access a shared resource at a time (same time), it will lead to distorted result.

Eg:- Class First{
Public void display (String msg){
System.out.println ("[" +msg);
{
Thread.sleep (1000);
}
Catch (InterruptedException e)
{
e.printStackTrace ();
}
System.out.println ("]");
}
}
class Second Extends Thread
{
String msg;
First fobj;
Second (First fp, String str)
{
fobj = fp;
msg = str;

```
start();
}
Public void run()
{
  Synchronized (fobj)
  {
   fobj.display (msg);
  }
 }
}

Public class MyThread
 {
  Public static void main (String[] args)
   {
    first fnew = new first();
    Second s1 = new Second (fnew, "welcome");
    Second s2 = new Second (fnew, "Home");
    Second s3 = new Second (fnew, "Programmer");
   }
}
```