Assignment - 2.

1. How many types of ways are there for creating threads? Explain each with an example.

A. There are two ways to create a thread:

1. By extending Thread class.
2. By implementing Runnable interface.

Thread class provide constructors and methods to create and perform operations on a thread. Thread class extends Object class and implements Runnable interface.

① Java Thread Example by extending Thread class.

FileName: Multi.java.

```
class Multi extends Thread {
  public void run(){
  System.out.println("thread is running...");
}
  Public Static void main( string args[]) {
    Multi t1 = new Multi();
      t1.start();
    } }
```

Output: Thread is running.

② Java Thread Example by implementing Runnable interface.

Filename: Multi3.java

```
class Multi3 implements Runnable {
    Public void run(){
```

```
    System.out.println("Thread is running.....");
    }
    public static void main(String args[]){
        Multi3 m1 = new Multi3();
        Thread t1 = new Thread(m1); //using the Constructor Thread(Runnable r)
        t1.start();
    } }
```

Output:- Thread is running.....

2. Delegation Event Model is essential in Event handling. Justify the Statement.

A. The Delegation Event model is defined to handle events in GUI programming languages. The GUI stands for Graphical User Interface, where a User graphically/visually interacts with the system.

The GUI programming is inherently event-driven; whenever a User initiates an activity such as a mouse activity, clicks, Scrolling, etc., each is known as an event that is mapped to a code to respond to functionality to the User. This is known as event handling.

Delegation Model defines a standard and compatible mechanism to generate and process events. In this model, a Source generates an event and forwards it to one or more listeners. The listener waits until it receives an event. Once it receives the event, it is processed by the listener and returns it. The UI elements are able to delegate the processing of an event to a Separate function.

The key advantage of the Delegation Event Model is that the application logic is Completely Separated from the interface logic

In the older model, an event was propagated up the containment until a component was handled. This ("method") needed components to receive events that were not processed, and it took lots of time. The Delegation Event model Overcame this issue.

Basically an Event Model is based on the following, three Components.

* Events , * Events Sources , * Events Listeners.

3. How do you handle the key events Using Java AWT. Explain.

A. changing the state of an Object is known as an event. for example click on button, dragging mouse etc. The java.awt.event package provides many event classes and Listener interfaces for event handling.

Steps to perform Event Handling:

Following steps are required to perform event handling:

1) Registration Methods:- For Registration the Component with the Listener, many classes provide the registration methods.

for Example:- ⇒ Button ⇒ public void addActionListener(ActionListener a){}

- TextField ⇒ Public void addActionListener(ActionListener a) { }
- TextArea ⇒ Public Void addActionListener (TextListener a) { }

2) Java Event Handling Code:-

We can put the event handling code into one of the following places:

1. Within class
2. Other class.
3. Anonymous class.

3) Java event handling by Outer class

4) Java event handling by anonymous class

4. Illustrate the ways of passing parameters to applets.

A. Parameter in Applet.

We can get any information from the HTML file as a parameter. For this purpose, Applet class provides a method named getParameter().

syntax:- Public String getParameter(String ParameterName)

Example:

```
import java.applet.Applet;
import java.awt.Graphics;
public class UserParam extends Applet {
  public void paint(Graphics g) {
    String str = getPassword("msg");
    g.drawString(str, 50, 50);
  }
}
```

My applet.html

```
<html>
<body>
<applet code = "UserParam.class" width = "300" height = "300">
<param name = "msg" value = "Welcome to applet">
</applet>
</body>
</html>
```

5. Compare and contrast any 4 AWT and swing components

A. AWT and Swing in Java.

AWT and swing are Used to develop window-based applications in Java. Awt is an abstract window toolkit that provides various components classes like Label, Button, TextField, etc., to show window components on the screen.

Swing is the part of JFC built on the top of AWT and written entirely in Java. The javax.swing API Provides all the components classes like JButton, JTextField etc. The components of swing are platform-independent i.e., Swing doesn't depend on the operating system to show the components.

Context	AWT	Swing
API Package	The AWT Component classes are provided by the java.awt package.	The swing component classes are provided by the javax.swing package.
Operating System	The Components Used in AWT are mainly dependent on the operating system.	The Components Used in Swing are not dependent on the Operating system. It is completely Scripted in Java.
Full-Form.	Java AWT stands for Abstract WindowToolkit	Java Swing is mainly referred to as Java Foundation classes (JFC).

Memory	Java AWT needs a higher amount of memory for the execution	Java Swing needs less memory space as compared to Java AWT.
Speed.	Java AWT is slower than Swing in terms of performance	Java Swing is faster than the AWT.
Functionality and Implementation	Java AWT many features that are completely developed by the developer. It serves as a thin layer of development on the top of the operating System.	Swing components provide the higher level inbuilt functions for the developer that facilitates the coder to write less code.