6.1 Creating Thread

Java is a multi-threaded programming language which means we can develop multi-threaded program using Java.

A multi-threaded program contains two or more parts that can run concurrently and each part can handle a different task at the same time making optimal use of the available resources especially when your computer has multiple CPUs.

By definition, multitasking is when multiple processes share common processing resources such as a CPU.

Multi-threading extends the idea of multitasking into applications where you can subdivide specific operations within a single application into individual threads.

Each of the threads can run in parallel. The OS divides processing time not only among different applications, but also among each thread within an application.

Multi-threading enables you to write in a way where multiple activities can proceed concurrently in the same program.

6.1.1 Extending the Thread class

The First way to create a thread is to create a new class that extends Thread class using the following two simple steps.

This approach provides more flexibility in handling multiple threads created using available methods in Thread class.

Step 1

You will need to override run( ) method available in Thread class. This method provides an entry point for the thread and you will put your

complete business logic inside this method. Following is a simple syntax of run() method:

```
public void run( )
```

Step 2

Once Thread object is created, you can start it by calling start( ) method, which executes a call to run( ) method.

Following is a simple syntax of start() method:

```
void start( );
```

Example:

```
class ThreadDemo extends Thread
{
private Thread t;
private String threadName;
ThreadDemo( String name)
{
threadName = name;
System.out.println("Creating " + threadName );
}
public void run()
{
System.out.println("Running " + threadName );
Try
 {
for(int i = 4; i > 0; i--)
{
System.out.println("Thread: " + threadName + ", " + i);
// Let the thread sleep for a while.
Thread.sleep(50);
```

```
}
}
catch (InterruptedException e)
{
System.out.println("Thread " + threadName + " interrupted.");
}
System.out.println("Thread " + threadName + " exiting.");
}
public void start ()
{
System.out.println("Starting " + threadName );
if (t == null)
{
t = new Thread (this, threadName);
t.start ();
}
}
}

public class TestThread
 {
public static void main(String args[])
 {
ThreadDemo T1 = new ThreadDemo( "Thread-1");
T1.start();
ThreadDemo T2 = new ThreadDemo( "Thread-2");
T2.start();
}
}
```

6.1.2 Implementing Runnable Interface

If your class is intended to be executed as a thread then you can achieve this by implementing a Runnable interface. You will need to follow three basic steps:

Step 1

As a first step, you need to implement a run() method provided by a Runnable interface. This method provides an entry point for the thread and you will put your complete business logic inside this method. Following is a simple syntax of the run() method:

```
public void run( )
```

Step 2

As a second step, you will instantiate a Thread object using the following constructor:

```
Thread(Runnable threadObj, String threadName);
```

Where, threadObj is an instance of a class that implements the Runnable interface and threadName is the name given to the new thread.

Step 3

Once a Thread object is created, you can start it by calling start( ) method, which executes a call to run( ) method. Following is a simple syntax of start() method:

```
void start( );
```

Example:

```
class RunnableDemo implements Runnable {
private Thread t;
```

```
private String threadName;
RunnableDemo( String name){
threadName = name;
System.out.println("Creating " + threadName );
}
public void run() {
System.out.println("Running " + threadName );
try {
for(int i = 4; i > 0; i--) {
System.out.println("Thread: " + threadName + ", " + i);
// Let the thread sleep for a while.
Thread.sleep(50);
}
}
catch (InterruptedException e) {
System.out.println("Thread " + threadName + " interrupted.");
}
System.out.println("Thread " + threadName + " exiting.");
}
public void start () {
System.out.println("Starting " + threadName );
if (t == null) {
t = new Thread (this, threadName);
t.start ();
}
}
}
public class TestThread
 {
```

```
public static void main(String args[])
{
RunnableDemo R1 = new RunnableDemo( "Thread-1");
R1.start();
RunnableDemo R2 = new RunnableDemo( "Thread-2");
R2.start();
}
}
```