Read Me:-

i. Before going through below exercises please visit the link given below, where you can experience the coding standard that each and every developer should follow.

ii. This Code Conventions for the Java Programming Language document contains the standard conventions that Sun follow and recommend that we should follow. It covers filenames, file organization, indentation, comments, declarations, statements, white space, naming conventions, programming practices and includes a code example.

iii. LINK - http://www.oracle.com/technetwork/java/codeconv-138413.html

Multithreading Lab Exercise Day-2

Duration: 2 Hours

1. Controlling the main thread…

a. Write a program to create a class 'CurrentThreadDemo' which
implements main method as per the following structure.

```
public static void main(String args[]) {
//Create a object of java.lang.Thread class using the 'currentThread()'
method of java.lang.Thread class.
//Print the above thread in the console with message as it is the current
thread.
//give a new name to this thread using setName() method of
java.lang.Thread() class.
//Print the name of the thread in the console with message as it is
the name of the thread after name change.
try {
for (intn = 5; n > 0; n--) {
//print the value of 'n' in the console.
//make the thread sleep for 1000 milisecs
}
} catch (InterruptedException e) {
//Print the exception message in the console as main thread
is inturupted.
}
}
```

Expected out put :

```
Current thread: Thread[main,5,main]
After name change: Thread[My Thread,5,main]
5 4 3 2 1
```

2. Change Thread Priority…

Write a program to create a class ‘CounterThread’ which extends
java.lang.Thread class having an instance variable ‘name’ of String type.
The class ‘CounterThread’ implements a parameterized constructor to
initialize the above declared Instance variable name.
The class ‘CounterThread‘ also override the run() method
of java.lang.Thread class as following.

```
public void run() {
intcount = 0;//Declare a local variable to
while (true) {
try
{
sleep(100);
} catch (InterruptedException e) {
}
if (count == 50)//if the count value is 50 initialize it to
zerocount = 0;
//print the value of name and count the console.
}
}
```

Write a program to create a class ‘CounterThreadDemo’ which
implements main method as per the following structure.

```
public static void main(String[] args) {
//Create a object ‘thread1’ of ‘CounterThread’ type name it as thread1 by
passing this name to it’s constructor.
//Set the priority of above thread ‘thread1’ as 10 by using
setPriority() method .
//Now Create a object ‘thread2’ of ‘CounterThread’ type name it as
thread2 by passing this name to it’s constructor.
//Set the priority of above thread ‘thread2’ as 1 by using
setPriority() method .
//Now start thread2() method using start() method.
//Now start thread1() method using start() method.
}
```

3. Minimum and Maximum Priority Threads…

Write a program to create a class ‘MyRunnable’ which implements
Runnable Interface having a instance variable ‘name’ of String type.This class
‘MyRunnable’ implements a parameterized constructor to initialize the above
instance variable ‘name’.This class ‘MyRunnable’ override the run() method to

print the value of instance variable ‘name’,make this inside a infinite loop.
Write a program to create a class ‘TestClass’ which implements
the default constructor as per the following structure

```
public TestClass() {
//Create a reference ‘runner’ of Runnable interface using the
parameterized constructor of ‘MyRunnable’ class passing ‘First’ as parameter.
//Create an object ‘t’ of Thread type by passing the above Runnable
reference as parameter to Thread class constructror.
//Set minimum priority to the above created thread ‘t’ by passing
Thread.MIN_PRIORITY as parameter to the method setPriority() method
//Now start the thread ‘t’ using start() method.
//Assign new object of ‘MyRunnable’ class to the reference ‘runner’
using the parameterized constructor of MyRunnable class passing ‘second’ as
parameter.
//Reassign value to object ‘t’ passing ‘runner’ as parameter to Thread class
constructor.
//set The Priority as maximum to the thread object ‘t’ by passing
Thread.MAX_PRIORITY as parameter to the setPrority() method.
//Now Start the above thread ‘t’ using start() method.
}
```

Now implement the main () method in the ‘TestClass’ ,Inside which create
an object of TestClass.

4. Demonstrate thread priorities…

Write a program to create a class ‘Clicker’ Which implements
Runnable interface having two instance variables ‘click’ ,‘t’ of int and Thread
type respectively.Also declare a private instance variable ‘running’ of boolean
type and make it volatile also initialize it as true. The class ‘Clicker’
implements a paramegterized constructor as following

```
public clicker(intp) {
//Initialize ‘t’ by creating a thread of this class.
//set the priority to thread ‘t’ as ‘p’ using setPriority() method .
}
```

The class ‘Clicker’ also override run(),start() and stop() methods as per
the following

```
public void run() {
while (running) {
click++;//Increment the variable click.
}
}
public void stop() {
```

//set the variable ‘running’ as false.
}
public void start() {
//Start the thread ‘t’;
}
Write a program to create a class ‘HighLowPriority’ which implements
main method as per the following structure.
public static void main(String args[]) {
//Set the priority of the current thread as maximum using methods like
currentThread(),setPriority() etc..passThread.MAXPRIORITY as parameter
to setPriority() method.
//Create a object ‘hi’ of Clicker type and set its priority more than that of
the normal priority by passing Thread.NORM_PRIORITY+2 as parameter to it’s
constructor.
//Create a object ‘lo’ of Clicker type and set its priority less than that of the
normal priority by passing Thread.NORM_PRIORITY-2 as parameter to it’s
constructor.
//Now start both the thread hi and lo using start()
method. try {
// Make the current thread sleep for 10000 miliseconds.
} catch (InterruptedException e) {
//Print appropriate message in the console as main thread interrupted.
}
//Now stop both the thread hi and lo using stop() method .
// Wait for child threads to terminate.
try {
hi.t.join();
lo.t.join();
} catch (InterruptedException e) {
System.out.println("InterruptedException caught");
}
System.out.println("Low-priority thread: " + lo.click);
System.out.println("High-priority thread: " + hi.click);
}
Expected output :
Low-priority thread: -995096799(this value going to vary with each run)
High-priority thread: -950947528(this value going to vary with each run)

5. Using join() to wait for threads to finish…

Write a program to create a class ‘NewThread’ which implements Runnable interface and having two instance variables ‘name’, ‘t’ of String and Thread type respectively. The class ‘NewThread’ implements a parameterized constructor as following

```
NewThread(String threadname) {
//Initialize the instance variable name with parameter threadname.
// Initialize the instance variable ‘t’ creating a thread object passing
appropriate parameter(this,name) to it’s parameterized constructor.
// Print the thread ‘t’ in the console with some appropriate message.
// Start the thread ‘t’ with start() method.
}
```

The class ‘NewThread’ override the run() method as following ..

```
public void run() {
try {
for (inti = 5; i > 0; i--){
System.out.println(name + ": " +i);
//Make that thread sleep for 1000 miliseconds.
  }
} catch (InterruptedException e) {
//Print appropriate message in the console to handle this exception.
}
System.out.println(name + " exiting.");
}
```

Write a program to create a class ‘DemoJoin’ which implements main method as following.

```
public static void main(String args[]) {
//Create three objects ‘ob1’,’ob2’,’ob2’of NewThread class passing
name(one,two,three) as parameters to their constructors.
//Use isAlive() to check out Whether these threads are alive or not.
System.out.println("Thread One is alive: " + ob1.t.isAlive());
System.out.println("Thread Two is alive: " + ob2.t.isAlive());
System.out.println("Thread Three is alive: " + ob3.t.isAlive());
try {
System.out.println("Waiting for threads to finish.");
ob1.t.join();
ob2.t.join();
ob3.t.join();
} catch (InterruptedException e) {
//Display appropriate message in the console that thread is interrupted.
}
//Once again check that whether these threads are alive or not.
System.out.println("Thread One is alive: " + ob1.t.isAlive());
System.out.println("Thread Two is alive: " + ob2.t.isAlive());
System.out.println("Thread Three is alive: " + ob3.t.isAlive());
System.out.println("Main thread exiting.");
}
```

6. Demonstrate ThreadGroup…

Write a program to create a class ‘NewThread’ which extends class Thread and having a instance variable ‘suspendFlag’ of boolean type. Class ‘NewThread’ also implements a parameterized constructor as following. NewThread(String threadname, ThreadGrouptgOb) {

```
super(tgOb,threadname);
suspendFlag= false; start();
}
```

The class ‘NewThread’ also implements run method as following public void run() {

```
try {
for (inti = 5; i > 0; i--){
System.out.println(getName() + ": " +i);
Thread.sleep(1000);
synchronized (this) {
while (suspendFlag)
{
  wait();
}
}
}
} catch (Exception e) {
System.out.println("Exception in " + getName());
}
}
```

The class ‘NewThread’ also implements two methods ‘mysuspend’ and ‘myresume’ as following

```
void mysuspend() {
// set the instance variable suspendFlag as true.
}
synchronized void myresume() {
//set the instance variable suspendFlag as
false. notify();
```

}
Write program to create a class ‘ThreadGroupDemo’ which implements the main method as following.

```
public static void main(String args[]) {
//Create a object groupA of ThreadGroup type passing “ GroupA”
String as a parameter to it’s constructor. As following
ThreadGroupgroupA = newThreadGroup("Group A");
//Simillarly create another object groupB of ThreadGroup type passing
“GroupB” String as a parameter to it’s constructor.
//Now create four objects of NewThread class as following ….
NewThread ob1 = newNewThread("One", groupA);
NewThread ob2 = newNewThread("Two", groupA);
NewThread ob3 = newNewThread("Three", groupB);
NewThread ob4 = newNewThread("Four", groupB);
groupA.list();
groupB.list();
Thread tga[] = new Thread[groupA.activeCount()];
groupA.enumerate(tga);
for (inti = 0; i <tga.length; i++){
((NewThread)tga[i]).mysuspend();
}
try{
 Thread.sleep(4000);
} catch (InterruptedException e) {
System.out.println("Main thread interrupted.");
}
System.out.println("Resuming Group A");
for (inti = 0; i <tga.length; i++){
  ((NewThread) tga[i]).myresume();
}
try{
ob1.join();
ob2.join();
ob3.join();
ob4.join();
} catch (Exception e) {
System.out.println("Exception in Main thread");
}
}
```

7. Create a class ‘MyDaemon’ which implements Runnable interface and having a instance variable ‘thrd’ of Thread class.The class ‘MyDaemon’ implements the default constructor as following

```
MyDaemon() {
//Initailise the instance variable ‘thrd’ by creating a thread of this
class. thrd.setDaemon(true);//Make this Thread a Daemon thread.
thrd.start();//Start the thread.
}
```

This class MyDaemon override isDaemon() and run() method as following.

```
public
booleanisDaemon(){
return
thrd.isDaemon();
}
public void run() {
try {
while(true) {
System.out.print(".");
Thread.sleep(100);
}
}
catch(Exception exc){
  System.out.println("MyDaemon interrupted.");
 }
}
```

Write a program to create a class ‘TestDaemon’ which implements main method as following

```
public static void main(String args[]) throws
Exception{//Create a object ‘dt’ of MyDaemon
class. if(dt.isDaemon())
//print appropriate message in the console that ‘dt’ is a daemon
thread.
//make the current thread sleep for 1000 miliseconds.
//Print appropriate message in the console that main thread is ending.
}
```

All the Best