JBK1014-Assignment : Thread

Thread creation by extending Thread class

Example 1:

```
class MultithreadingDemo extends Thread{
 public void run(){
  System.out.println("My thread is in running state.");
 }
 public static void main(String args[]){
   MultithreadingDemo obj=new MultithreadingDemo();
   obj.start();
 } }0
```

Example 2:

```
Class Count extends Thread {
  Count(){
   super("my extending thread");
   System.out.println("my thread created" + this);
   start();
  }
  public void run()  {
   try{
     for (int i=0 ;i<10;i++){
       System.out.println("Printing the count " + i);
       Thread.sleep(1000);
     }}
   catch(InterruptedException e)  {
     System.out.println("my thread interrupted");
   }
   System.out.println("My thread run is over" );
  } }
class ExtendingExample{
  public static void main(String args[]){
    Count cnt = new Count();
    try{
      while(cnt.isAlive()){
        System.out.println("Main thread will be alive till the child thread is
live");
        Thread.sleep(1500);
      } }
    catch(InterruptedException e){
      System.out.println("Main thread interrupted");
    }
    System.out.println("Main thread's run is over" );
  } }
```

Create a Simple Thread in Java

```
public class Thread_Ex1 extends Thread{
        String text = null;

        Thread_Ex1(String str) {
                text = str;
                start(); }
        public void run() {
                System.out.println(text);
        } }
class MainClass {
        public static void main(String args[]) {
                new Thread_Ex1("Thread Activity Started..");
        } }
```

Thread creation by implementing Runnable Interface

```
class MultithreadingDemo implements Runnable{
 public void run(){
  System.out.println("My thread is in running state.");
 }
 public static void main(String args[]){
   MultithreadingDemo obj=new MultithreadingDemo();
   Thread tobj =new Thread(obj);
   tobj.start();
} }
```

Example Program 2:

```
class Count implements Runnable{
  Thread mythread ;
  Count(){
    mythread = new Thread(this, "my runnable thread");
    System.out.println("my thread created" + mythread);
    mythread.start();
  }
  public void run(){
    try{
     for (int i=0 ;i<10;i++){
       System.out.println("Printing the count " + i);
       Thread.sleep(1000);
     } }
   catch(InterruptedException e) {
     System.out.println("my thread interrupted");
   }
   System.out.println("mythread run is over" );
  } }
class RunnableExample{
   public static void main(String args[]){
     Count cnt = new Count();
     try {
       while(cnt.mythread.isAlive()) {
         System.out.println("Main thread will be alive till the child thread is
live");
         Thread.sleep(1500);
       }
     }
     catch(InterruptedException e)
     {
       System.out.println("Main thread interrupted");
     }
     System.out.println("Main thread run is over" );
   }
}
```

Create a Runnable Thread in Java

```
public class Thread_Ex2 implements Runnable{

        String text = null;
        Thread thread;
        Thread_Ex2(String str) {
                text = str;
                thread = new Thread(this);
                thread.start();
        }  public void run() {
                System.out.println(text);
        } }
class MainClass {
        public static void main(String args[]) {
                new Thread_Ex2("Thread Activity Started..");
        } }
```

Without using join() Example:

```
class MyClass2 implements Runnable{

  public void run() {
        Thread t = Thread.currentThread();
     System.out.println("Thread started: "+t.getName());
     try {
        Thread.sleep(4000);
     } catch (InterruptedException ie) {
        ie.printStackTrace();
     }  System.out.println("Thread ended: "+t.getName());
  } }
public class JoinExample2 {
  public static void main(String[] args) {
    Thread th1 = new Thread(new MyClass2(), "th1");
    Thread th2 = new Thread(new MyClass2(), "th2");
    Thread th3 = new Thread(new MyClass2(), "th3");
    th1.start();
    th2.start();
    th3.start();
  }  }
```

same example with join()

```
class MyClass implements Runnable{
   public void run() {
           Thread t = Thread.currentThread();
      System.out.println("Thread started: "+t.getName());
      try {
         Thread.sleep(4000);
      } catch (InterruptedException ie) {
         ie.printStackTrace();
      }
      System.out.println("Thread ended: "+t.getName());
   } }
public class JoinExample {
  public static void main(String[] args) {
    Thread th1 = new Thread(new MyClass(), "th1");
    Thread th2 = new Thread(new MyClass(), "th2");
    Thread th3 = new Thread(new MyClass(), "th3");

    // Start first thread immediately
    th1.start();
      /* Start second thread(th2) once first thread(th1)
     * is dead
     */
    try {
       th1.join();
    } catch (InterruptedException ie) {
       ie.printStackTrace();
     }
    th2.start();

    /* Start third thread(th3) once second thread(th2)
     * is dead
     */
    try {
       th2.join();
    } catch (InterruptedException ie) {
        ie.printStackTrace();
     }
    th3.start();

    // Displaying a message once third thread is dead
    try {
       th3.join();
    } catch (InterruptedException ie) {
        ie.printStackTrace();
     }
    System.out.println("All three threads have finished execution");
  } }
```

Call sleep() using Thread in Java

```
import java.util.Date;

public class Thread_Ex4 {

        int wait = 0;

        Date dt;

        void ThreadActivity() {

                dt = new Date();
                System.out.println("Before Threading");
                System.out.println(dt.getHours() + " : "
                                + dt.getMinutes() + " : "
                                + dt.getSeconds());

                try {
                        while(wait <= 100) {
                                Thread.sleep(100);
                                wait ++;
                        }
                }
                catch (Exception e) {

                        System.out.println("Error : " + e.toString());
                }
                finally {

                        dt = new Date();
                        System.out.println("\nAfter Threading");
                        System.out.println(dt.getHours() + " : "
                                        + dt.getMinutes() + " : "
                                        + dt.getSeconds());
                }               }               }
```

```
class MainClass {

        public static void main(String args[]) {

                Thread_Ex4 obj = new Thread_Ex4();

                obj.ThreadActivity();
        }
}
```

example shows how to use setPriority() and getPriority() methods.

```
class MyThread extends Thread
{
  public MyThread(String name)
  {
    super(name);
  }


  public void run()
  {
    for(int i = 0; i < 1000; i++)
    {
      System.out.println("from "+getName());
    }
  }
}

public class ThreadsInJava
{
  public static void main(String[] args)
  {
    MyThread t1 = new MyThread("Thread 1");

    t1.setPriority(5);      //Setting the priority of Thread 1

    t1.start();

    MyThread t2 = new MyThread("Thread 2");

    t1.setPriority(7);      //Setting the priority of Thread 2
    t2.start();

    System.out.println(t1.getPriority());     //Output : 5

    System.out.println(t2.getPriority());     //Output : 7
  }
}
```

example for using wait() and notify() methods:

```
public class ThreadsInJava
{
  public static void main(String[] args)
  {
    System.out.println(Thread.MIN_PRIORITY);    //Output : 1

    System.out.println(Thread.NORM_PRIORITY);   //Output : 5

    System.out.println(Thread.MAX_PRIORITY);    //Output : 10
  }
}

class Shared
{
  synchronized void methodOne()
  {
    Thread t = Thread.currentThread();

    System.out.println(t.getName()+" is relasing the lock and going to
wait");

    try
    {
      wait();      //releases the lock of this object and waits
    }
    catch (InterruptedException e)
    {
      e.printStackTrace();
    }



 System.out.println(t.getName()+" got the object lock back and can
continue with it's execution");
  }
  synchronized void methodTwo()
  {
    Thread t = Thread.currentThread();

    try
    {
      Thread.sleep(5000);
    }
    catch (InterruptedException e)
    {
      e.printStackTrace();
    }

    notify();    //wakes up one thread randomly which is waiting for lock of
this object

    System.out.println("A thread which is waiting for lock of this object is
notified by "+t.getName());
  }
}

public class ThreadsInJava
{
  public static void main(String[] args) {
    final Shared s = new Shared();

    Thread t1 = new Thread(){
      public void run()  {
         s.methodOne();   //t1 calling methodOne() of 's' object
      }
    };
     Thread t2 = new Thread()  {
             public void run(){
         s.methodTwo();   //t2 calling methodTwo() of 's' object
      }
    };
    t1.start();

    t2.start();
  } }
```