# EXPERIMENT 5 – THREAD MANAGEMENT

##### OBJECTIVES

* Using Java Threads
* Thread Synchronization

**TIME REQUIRED** : 3 hrs

**PROGRAMMING LANGUAGE** : C/C++/Java

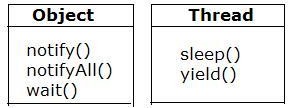
**SOFTWARE REQUIRED** : Ubuntu/Fedora, gcc/gc, Windows, Dev, NetBeans **HARDWARE REQUIRED** : Core i5 in Computer Labs **MULTITHREADING:**

Multithreading refers to two or more tasks executing concurrently within a single program. A thread is an independent path of execution within a program. Many threads can run concurrently within a program.

##### CONTEXT SWITCH USING THREADS:

Switching among threads is much cheaper than to switch processes as threads share the same process area.

##### JAVA THREADS:

Every thread in Java is created and controlled by the **java.lang.Thread class**. A Java program can have many threads, and these threads can run concurrently, either asynchronously or synchronously. Following shows the methods in Object and Thread Classes of Java.

##### THREAD CREATION BY IMPLEMENTING THE RUNNABLE INTERFACE:

public interface Runnable { void run();

}

One way to create a thread in java is to implement the Runnable Interface and then instantiate an object of the class. We need to override the run() method into our class which is the only method that needs to be implemented. The run() method contains the logic of the thread.

* A class implements the Runnable interface, providing the run() method that will be executed by the thread. An object of this class is a Runnable object.
* An object of Thread class is created by passing a Runnable object as argument to the Thread constructor. The Thread object now has a Runnable object that implements the run() method.
* The start() method is invoked on the Thread object created in the previous step. The start() method returns immediately after a thread has been spawned.

The thread ends when the run() method ends, either by normal completion or by throwing an uncaught exception.

##### THREAD CREATION BY EXTENDING THE THREAD CLASS:

The procedure for creating threads based on extending the Thread is as follows:

* A class extending the Thread class overrides the run() method from the Thread class to define the code executed by the thread.
* This subclass may call a Thread constructor explicitly in its constructors to initialize the thread, using the super() call.
* The start() method inherited from the Thread class is invoked on the object of the class to make the thread eligible for running.

##### TASK 5.1:

Type in the following code. Execute and show outcome:

/\*\*

* @(#)ThreadExample.java
* @author Engr. Umar Mahmud

\* @version 1.00 2014/4/25

\*/

public class ThreadExample extends Thread {

//To create producer and consumer as threads public ThreadExample(String threadName){

//Constuctor super(threadName);

//Call to constructor of Thread class

}

public void run(){

System.out.println("I am a thread and my name is "+Thread.currentThread().getName());

}

public static void main(String[] args) {

ThreadExample t = new ThreadExample("Thread 01");

//Create a thread object

t.start(); //execute run() method

}

}

##### EXERCISE 5.1 [3]

Create another thread object and execute it. Show code and outcome here:

##### TASK 5.2:

Execute the following code for producer and consumer. Execute and show outcome:

/\*\*

* @(#)ProducerConsumerUsingThreads.java
* ProducerConsumerUsingThreads application
* @author Engr. Umar Mahmud

\* @version 1.00 2013/12/20

\*/

public class ProducerConsumerUsingThreads extends Thread { private int x = 0; //Shared variable

private int maxLimit = 5;//Maximum Limit to simulate a buffer public ProducerConsumerUsingThreads(String threadName) {

//Constuctor

super(threadName); //Call to constructor of Thread class

}

public void run() {

for (int i = 0; i < 10; i++) { System.out.println(Thread.currentThread().getName()); if(Thread.currentThread().getName().startsWith

("Producer")) { //In a producer if (x < maxLimit) {

x++;

} else {

System.out.println("\nProducer blocked"); yield(); //Give way to another thread

}

}

else if (Thread.currentThread().getName().startsWith ("Consumer")) { //In a Consumer

if (x > 0) {

x--;

} else {

System.out.println("\nConsumer blocked"); yield(); //Give way to another thread

}

} else { //default case System.out.println("Some Error");

}

}

}

public static void main(String[] args) { ProducerConsumerUsingThreads p = new

ProducerConsumerUsingThreads("Producer"); ProducerConsumerUsingThreads c = new

ProducerConsumerUsingThreads("Consumer"); p.start();

c.start();

}

}

Re-execute the code and identify if the output trace is same as before.

Again re-execute the code and identify if the output trace is same as before.

What is your conclusion?

##### EXCERSISE 5.2 [7]

Create two threads one to read from a file and the other to write into another file. **Show the output and code**

for a given text file.

##### RESOURCES

<http://www.youtube.com/watch?v=KxTRsvgqoVQ> <http://www.youtube.com/watch?v=D0u2USIonds> <http://www.youtube.com/watch?v=CKFq-WTZZc8> <http://docs.oracle.com/javase/6/docs/api/java/lang/Thread.html>[MUST VISIT] <http://www.javabeginner.com/learn-java/java-threads-tutorial>