# Multithreading

```
/*
 * 1. Write a Java program to perform a runnable interface, take two threads t1 and 12 and fetch the names of the thread
 * using getName() method.
 * Expected output:
 * Thread names are following:
 * Thread A
 * Thread B
 */

public class problem1{
    public static void main(String[] args) {
        Thread t1 = new Thread(new impThread(), "Thread A"); // creating thread t1 by passing impThread object and name of thread as "
        Thread t2 = new Thread(new impThread(), "Thread B"); // creating thread t2 by passing impThread object and name of thread as "

        System.out.println("Thread names are following:");
        System.out.println(t1.getName()); // printing name of thread t1
        System.out.println(t2.getName()); // printing name of thread t2
    }
}

class impThread implements Runnable{ // impThread class implements Runnable interface
    @Override
    public void run(){ // run method
        System.out.println("Running ");
    }
}
```

```
/*
 * 2. You need to run this following simple program with at least three threads (name those threads as A,B,C....),
 * and find the output.
 * i.   Initialize a variable sum = 0
 * ii.  Run a for loop for 10 times. i.e: for(int x = 0; x < 10; x++). Add the value x to the value sum in each iteration
 * iii. Print the thread name, and the value as follows: "Thread: A - value : 45" in each iteration
 * iv.  Print the final value of sum after the loop is completed.
 */
import java.lang.Thread;
public class problem2 {
    public static void main(String[] args){
        Thread t1 = new Thread(new iThread(), "Thread A");
        Thread t2 = new Thread(new iThread(), "Thread B");
        Thread t3 = new Thread(new iThread(), "Thread C");
        //Thread t4 = new Thread(new iThread(), "Thread D");

        t1.start();
        t2.start();
        t3.start();
        //t4.start();
    }
}

class iThread implements Runnable{
    int sum;
    @Override
    public void run(){
        sum = 0;
        for (int i = 0; i < 10; i++) {
            sum += i;
            System.out.println(Thread.currentThread().getName() + " - value : " + sum); // printing thread name and value of sum
        }
        System.out.println(Thread.currentThread().getName() + " - Final value of sum : " + sum); // printing final value of sum
    }
}
```

```
/*
 * 3. Write a program to print "Good morning" and "Welcome" continuously on the screen in Java using threads.
 */

public class problem3 {
    public static void main(String[] args) {
        Thread t1 = new Thread(new iThread("Good Morning"));
        Thread t2 = new Thread(new iThread("Welcome"));
        t1.start();
        t2.start();
        try {
            Thread.sleep(50000);
        } catch (InterruptedException e) {
            //e.printStackTrace();
        }
    }
}

class iThread implements Runnable{
    String greetingMsg;
    iThread (String greetingMsg){
        this.greetingMsg = greetingMsg;
    }
    @Override
    public void run(){
        while (true){
            System.out.println(greetingMsg);
        }
    }
}
```

```
/*
 * 4. Write a program to print "Good morning" and "Welcome" continuously on the screen in Java using threads.
 *
 * Add a step method in the welcome thread of question 3 to delay its execution for 200ms.
 */

public class problem4 {
    public static void main(String[] args) {
        Thread t1 = new Thread(new iThread("Good Morning"), "Thread A");
        Thread t2 = new Thread(new iThread("Welcome"), "Thread B");

        t1.start();
        t2.start();
    }
}

class iThread implements Runnable{
    String greetingMsg;
    iThread (String greetingMsg){
        this.greetingMsg = greetingMsg;
    }
    @Override
    public void run(){
        while (true){
            System.out.println(greetingMsg);
            if (greetingMsg.equals("Welcome")){
                step(200);
            }
        }
    }

    public void step(int milliSec){
        try {
            Thread.sleep(milliSec);
        } catch (InterruptedException e){
            System.out.println(e.getMessage());
            //e.printStackTrace();
        }
    }
}
```

```
/*
 * 5. Which integer between 1 and 10000 has the largest number of divisors, and how many divisors does it have?
 * Write a program to find the answers and print out the results. It is possible that several integers in
 * this range have the same, maximum number of divisors. Your program only has to print out one of them.
 * Now write a program that uses multiple threads to solve the same problem, but for the range 1 to 100000.
 * By using threads, your program will take less time to do the computation when it is run on a multiprocessor
 * computer. At the end of the program, output the elapsed time, the integer that has the largest number of
 * divisors, and the number of divisors that it has. For this exercise, you should simply divide up the problem
 * into parts and create one thread to do each part.
 */

public class divisorThread implements Runnable{ // implements Runnable interface
    static int largestDivisor = 0;
    static int numberWithLargestDivisor = 0;
    int start;
    int end;

    public divisorThread(int start, int end){
        this.start = start;
        this.end = end;
    }

    @Override
    public void run(){
        divisor(start, end); // calling divisor method
    }

    public static void main(String[] args) {
        Thread t1 = new Thread(new divisorThread(1, 2500), "Thread A"); //
        Thread t2 = new Thread(new divisorThread(2501, 5000), "Thread B");
        Thread t3 = new Thread(new divisorThread(5001, 7500), "Thread C");
        Thread t4 = new Thread(new divisorThread(7501, 10000), "Thread D");

        long startTime = System.currentTimeMillis();

        t1.start();
        t2.start();
        t3.start();
        t4.start();

        try {
            t1.join();
            t2.join();
            t3.join();
            t4.join();
        } catch (InterruptedException e) {
            e.printStackTrace();
        }

        long endTime = System.currentTimeMillis();
        long elapsedTime = endTime - startTime;

        System.out.println("The number with the largest number of divisors is " + numberWithLargestDivisor + " with " + largestDivisor
        System.out.println("The elapsed time is " + elapsedTime + " milliseconds.");
    }

    public static void divisor(int start, int end){
        for (int i = start; i <= end; i++){
            int divisorCount = 0;
            for (int j = 1; j <= i; j++){
                if (i % j == 0){
                    divisorCount++;
                }
            }
            if (divisorCount > largestDivisor){
                largestDivisor = divisorCount;
                numberWithLargestDivisor = i;
            }
        }
    }
}
```