AP Computer Science A

Assignment – The Blockchain Assignment 3: Preliminaries for Multithreading

---

Create a new repl. Write responses to the following as comments in the code.

1. Write a method that solves the following problem:

   ```
   find the integer in the range 1 to 10000 that has the largest number of divisors
   ```

2. Use the `Date` class in the `java.util` package to time how long it takes to execute youa method.
3. Change the "search range" of your method to 1 to 100000. Time the execution of this new version.

*A **thread** is a path followed by the computer through a sequence of code. **Multithreading** is concurrent execution of two or more threads.*

4. Suppose that I ***partitioned*** *the* problem from prompt 1 of this assignment into many, smaller subproblems:

   ```
   find the integer in the range 1 to 2000 that has the largest number of divisors
   find the integer in the range 2001 to 4000 that has the largest number of divisors
   find the integer in the range 4001 to 6000 that has the largest number of divisors
   find the integer in the range 6001 to 8000 that has the largest number of divisors
   find the integer in the range 8001 to 10000 that has the largest number of divisors
   ```

   and defined each subproblem to be a different thread. Why might a multithreaded, concurrent solution yield a faster runtime than your original solution?

5. Add the following code to your repl.

```
public class Main
{
  public static void main (String args[])
  {
    ThreadDemo t1 = new ThreadDemo ("miss");
    ThreadDemo t2 = new ThreadDemo ("issippi");
    t1.start();
    t2.start();
  }
}

public class ThreadDemo extends Thread
{
   private Thread t;
   private String outputString;

   ThreadDemo (String s)
   {
      outputString = s;
   }

   //override
   public void run ()
   {
      for (int i = 0; i < 10; i++)
      {
         System.out.println (outputString);
         try {Thread.sleep (1);}
         catch (Exception e) {}
      }
   }

   //override
   public void start ()
   {
      if (t == null) {
         t = new Thread (this, outputString);
         t.start ();
      }
   }
}
```

6. Focus your attention on method `run` and ignore the `try/catch` block for a moment. What does this method do?
7. Run this new code a few times—enough times to generate "different" output. From just looking at the output, how can you tell that there are two distinct, concurrent threads of execution? Explain.
8. Explain the operation of the method `start` that is overridden in `ThreadDemo`.
9. What "burning questions" do you have at this point in your thinking about these concepts?

In the next lesson we will learn how to create threads by extending the `Thread` class; overriding the `start` and `run` methods of that class.

In subsequent lessons, we will code a multithreaded, concurrent solution to the coding problem presented at the start of this assignment.

Constructor

Constructor and Description
`Date``()` Allocates a `Date` object and initializes it so that it represents the time at which it was allocated, measured to the nearest millisecond.

Selected Method Summary

Modifier and Type	Method and Description
`int`	**getMinutes**`()` Returns the number of the minutes represented by this `Date` object.
`int`	**getHours**`()` Returns the number of the hours represented by this `Date` object.
`int`	**getSeconds**`()` Returns the number of the seconds represented by this `Date` object.
`long`	**getTime**`()` Returns the number of milliseconds since January 1, 1970, 00:00:00 GMT represented by this `Date` object.