# **Programming Assignment Unit 3**

Computer Science, University of the People

CS 1103-01 Programming 2 - AY2024-T4

Instructor, Rupa Sarda

May 1, 2024

For this assignment I was required to demonstrate a simple application that has two threads running simultaneously in the background, the first I s a clock thread that outputs the time to the console constantly, and the second a “heavy” non-blocking background thread that works on its load while not interfering with the other thread. I chose to use the Fibonacci calculation as my load thread since it is easy to simulate a load using the recursive function at higher numbers.

Code:

public class Main {

    public static void main(String[] args) {

        Clock clock = new Clock();

        Fibonachi fibonachi = new Fibonachi();

        Thread clockThread = new Thread(clock);

        clockThread.setPriority(Thread.MAX\_PRIORITY);

        clockThread.start();

        Thread fibonacciTask = new Thread(fibonachi);

        fibonacciTask.setPriority(Thread.MIN\_PRIORITY);

        fibonacciTask.start();

    }

}

import java.text.SimpleDateFormat;

import java.util.Date;

public class Clock implements Runnable {

    private final SimpleDateFormat formatter = new SimpleDateFormat("HH:mm:ss dd-MM-yyyy");

    public void run() {

        while (true) {

            Date now = new Date();

            System.out.println(formatter.format(now));

            try {

                Thread.sleep(1000); // Sleep for 1 second before updating the time again

            } catch (InterruptedException e) {

                System.out.println("Clock thread interrupted.");

                return;

            }

        }

    }

}

public class Fibonachi implements Runnable {

    @Override

    public void run() {

        // I chose to run from 40 to 100 to show the time taken for heavy computation

        // since my pc started to lag at 40, if your pc is faster you can increase the

        // range

        for (int i = 40; i <= 100; i++) {

            long fibNumber = fibonacci(i);

            System.out.println("Fibonacci(" + i + ") = " + fibNumber);

        }

    }

    // A simple recursive method to calculate the Fibonacci number

    private static long fibonacci(int n) {

        if (n <= 1) {

            return n;

        } else {

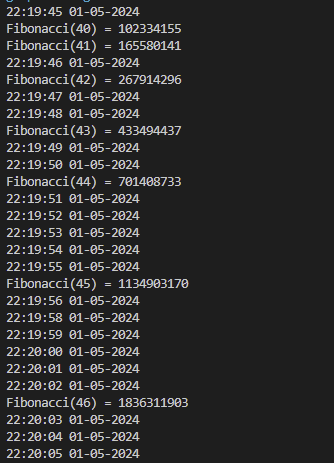
            return fibonacci(n - 1) + fibonacci(n - 2);

        }

    }

}

Screenshot:



## References

* Learning Guides  
  <https://my.uopeople.edu/course/view.php?id=7671>
* Eck, D. J. (2022). Introduction to programming using java version 9, JavaFX edition. Licensed under CC 4.0.  
  <https://math.hws.edu/javanotes/>
* Morelli, R. & Wade, R. (n.d.). Exceptions - When things go wrong. LibreTexts. Licensed under CC 4.0.  
  <https://eng.libretexts.org/Bookshelves/Computer_Science/Programming_Languages/Java_Java_Java_-_Object-Oriented_Programming_(Morelli_and_Walde)/00%3A_Front_Matter>
* Fibonacci sequence - <https://en.wikipedia.org/wiki/Fibonacci_sequence>

Source:

