



Dc Project

Project realised by *The Big 4*

Timisoara
May, 2024

Students:

Manole Paul,
Voineasa Ana-Maria,
Horoba Andreea,
Stanca Madalina

Content:

1 Introduction

1.1 Context

1.2 Motivation

2 State of the art

3 Design and implementation

4 Usage

5 Results + Output

6 Conclusions

7 Bibliography

Chapter 1

Introduction: 1.1Context

Our benchmark tests both the CPU and the RAM.

The CPU is tested using a Fibonacci's Sequence related algorithm. The user is asked to input the number of elements wanted and the program will display the time and the sequence.

The RAM component is tested by having the length of an array and the number of cycles.

The benchmark also has incorporated a feature called INFO, which upon interaction displays general information about the device the program is ran on.

1.2Motivation

Our interface design was inspired by the Fibonacci Golden Ratio, which is approximately 1:0.618 and is considered aesthetically pleasing.

This project combines two existing benchmarks: CPU and RAM, with an added feature inspired by the Fibonacci sequence. We'll delve deeper into our inspiration in the next chapter.

Chapter 2

State of the art

The Y-Crunche

Our first inspiration was the Y-Cruncher. Y-Cruncher is a multi-threaded Pi benchmark that leverages your computer's processing power to calculate Pi and other constants with high precision, reaching trillions of digits.

This benchmark is notable for being the first of its kind to be multi-threaded and scalable to multi-core systems. Since its launch in 2009, it has become a popular benchmarking and stress-testing tool. Despite initial skepticism, Y-Cruncher has set numerous records and is widely used. It holds the standard for the most digits of Pi ever computed, achieving 13.3 trillion digits in October 2014.

Advantage-High Precision Calculations: It can compute Pi and other constants to trillions of digits, providing an excellent measure of computational precision.

Stress Testing: It is useful for testing memory configurations and pushing the limits of your hardware components.

Multi-Core Utilization: Being multi-threaded, it effectively utilizes multi-core systems, making it a robust tool for modern processors

Disadvantages

Lack of Benchmark Comparison: It does not provide a comparison between your score/time and an optimal score/time, making it difficult to gauge performance.

High Memory Demand: Due to the memory-intensive nature of calculating Pi and other constants, Y-Cruncher requires substantial memory bandwidth to perform effectively.

Resource Intensive: The heavy computational load can be taxing on the system, which may not be suitable for all environments.

MemTest86

MemTest86 is a widely used memory testing software designed to check the integrity of your computer's RAM. It can detect and diagnose a variety of memory-related issues, from faulty RAM modules to incorrect configurations. MemTest86 runs independently of the operating system, booting from a USB drive or CD, which allows it to thoroughly test memory without interference from other software.

Advantages

Comprehensive Testing: MemTest86 performs a wide range of tests on RAM, identifying errors that could lead to system instability or crashes.

Bootable and OS-Independent: Since it runs independently of the operating system, it can test memory more thoroughly and avoid conflicts that might occur during normal operation.

Disadvantages

Time-Consuming: Thorough memory testing can take several hours, especially with larger RAM capacities, which can be inconvenient for users needing a quick diagnosis.

No Real-Time Usage Insights: While it tests memory effectively, it does not provide insights into memory performance under typical usage conditions.

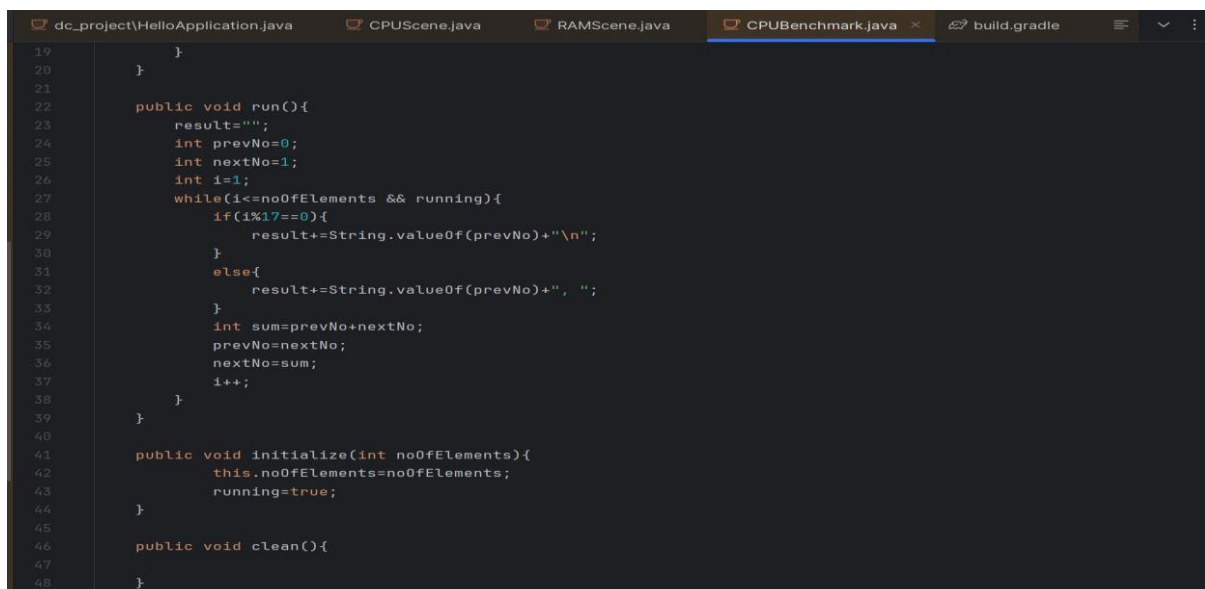
Chapter 3

Design and implementation

The benchmarks are testing the CPU and the RAM.

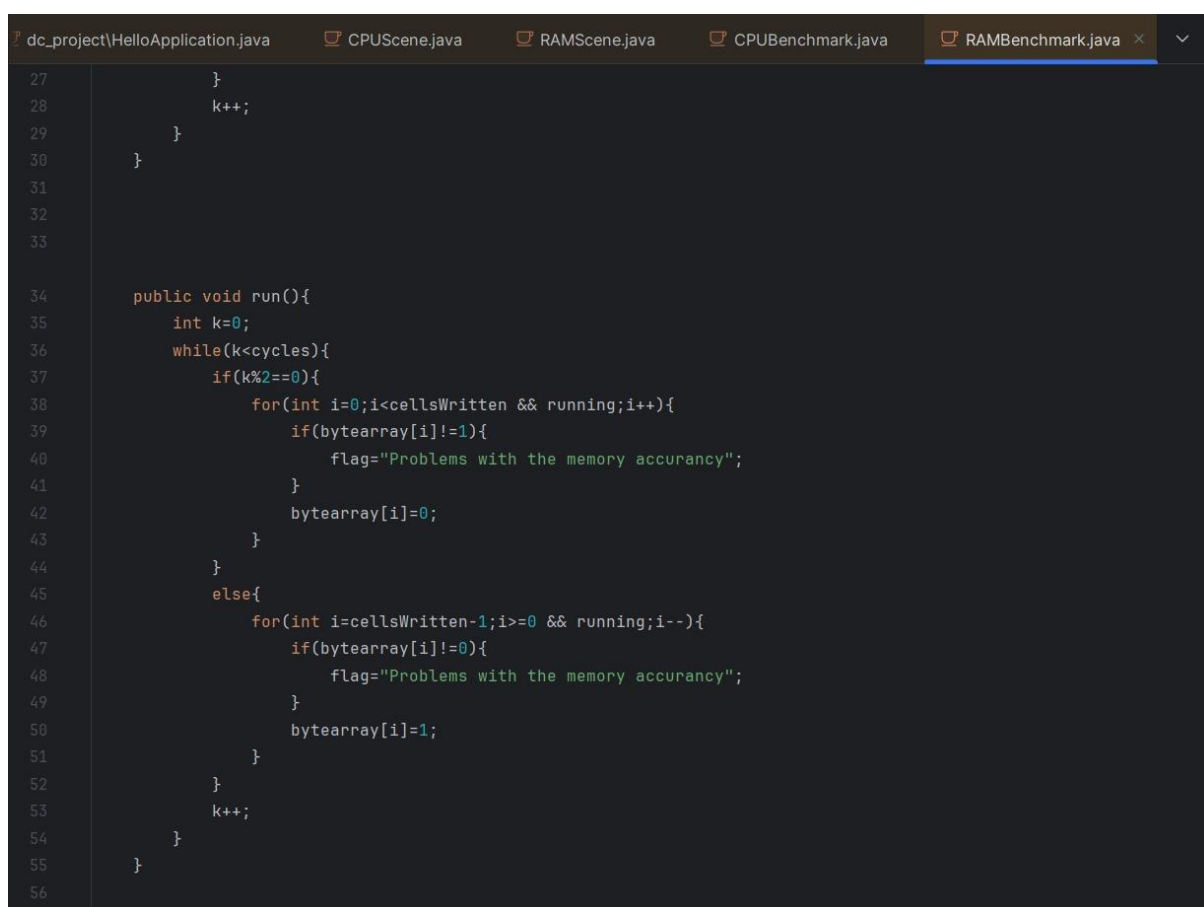
CPU: The CPU is the brain of the computer, responsible for executing instructions from programs. It performs the basic arithmetic, logic, control, and input/output (I/O) operations specified by the instructions in the application. We implemented the Fibonacci algorithm .

Here is a piece of the run function for the Fibonacci:



```
19     }
20 }
21
22 public void run(){
23     result="";
24     int prevNo=0;
25     int nextNo=1;
26     int i=1;
27     while(i<=noOfElements && running){
28         if(i%17==0){
29             result+=String.valueOf(prevNo)+"\n";
30         }
31         else{
32             result+=String.valueOf(prevNo)+", ";
33         }
34         int sum=prevNo+nextNo;
35         prevNo=nextNo;
36         nextNo=sum;
37         i++;
38     }
39 }
40
41 public void initialize(int noOfElements){
42     this.noOfElements=noOfElements;
43     running=true;
44 }
45
46 public void clean(){
47
48 }
```

RAM: RAM is the computer's short-term memory, used to store data that the CPU needs quick access to while executing an application. It is much faster than long-term storage devices like hard drives or SSDs. Here is a piece of the run function :

A screenshot of an IDE window showing a Java file named RAMBenchmark.java. The window has several tabs at the top: dc_project\HelloApplication.java, CPUScene.java, RAMScene.java, CPUBenchmark.java, and RAMBenchmark.java (which is the active tab). The code is displayed in a dark-themed editor with line numbers on the left. The code shows a run() method that contains a while loop for cycles, with nested for loops and if statements for memory accuracy checks. The code is as follows:

```
27         }
28         k++;
29     }
30 }
31
32
33
34 public void run(){
35     int k=0;
36     while(k<cycles){
37         if(k%2==0){
38             for(int i=0;i<cellsWritten && running;i++){
39                 if(bytearray[i]!=1){
40                     flag="Problems with the memory accuracy";
41                 }
42                 bytearray[i]=0;
43             }
44         }
45         else{
46             for(int i=cellsWritten-1;i>=0 && running;i--){
47                 if(bytearray[i]!=0){
48                     flag="Problems with the memory accuracy";
49                 }
50                 bytearray[i]=1;
51             }
52         }
53         k++;
54     }
55 }
56
```


Chapter 4

Usage

Using the benchmark is quite simple. It basically can be used with any operating systems. It can be easy to navigate through the app just by clicking the desired option. We used graphics inspired by the Cars Movie.

Home Page:

In the home page we have 3 options:

- CPU
- RAM
- INFO

CPU Page:

First, you are required to enter the number of elements of the Fibonacci sequence and how many seconds you are willing to wait.

To start the actual proccess just press START.

If the input doesn't match the required parameters it will take you to a "Wrong input page" having the option to go back and start again.

RAM Page:

For the RAM benchmark, you will be asked, „How many times do you want to read/write data?“ You can enter any number, but keep in mind that a higher number will take more time.

You will also need to enter the maximum number of seconds you are willing to wait and the size of the array. To start the benchmark, press „START“ . There is also an option to return to the main page. If the input is incorrect, an error message will appear.

INFO Page:

The last page displays information about the system such as the Operating System, the RAM, the processor name, the frequency, the number of the physical CPU...

It doesn't need any given input. It just offers additional information. You also have the „Go back“ option which takes you to the Home Page.

Chapter 5 Results-Output

CPU: After entering the number of elements for Fibonacci sequence, the execution time and the actual Fibonacci sequence will be shown.

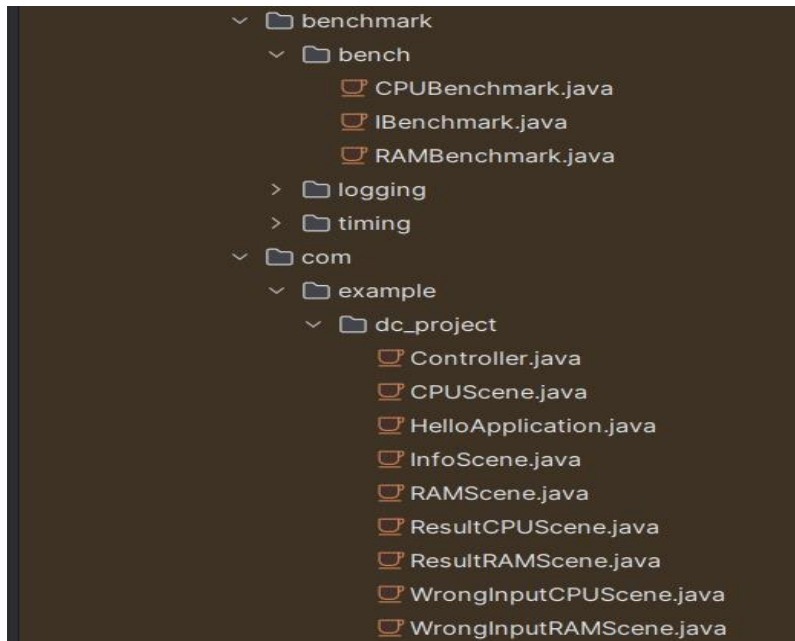
In this page you have the option to go back and also the option to see your CPU score by pressing the score button.

RAM: After entering the number, the size of the array and the number of seconds, the I/O throughput is displayed as also a message about the status of your proccess.

Just like the CPU page, you have the possibility to go back or see your RAM score.

RESULTS:

Here is the structure of the app in IntelliJ:



Also, a piece from the HelloApp class for the app:

```
1 package org.example.java_fx;
2
3 import javafx.application.Application;
4 import javafx.fxml.FXMLLoader;
5 import javafx.scene.Scene;
6 import javafx.stage.Stage;
7
8 import java.io.IOException;
9
10 public class HelloApplication extends Application {
11     @Override
12     public void start(Stage stage) throws IOException {
13         FXMLLoader fxmlLoader = new FXMLLoader(HelloApplication.class.getResource("hello-view.fxml"));
14         Scene scene = new Scene(fxmlLoader.load(), 320, 240);
15         stage.setTitle("Hello!");
16         stage.setScene(scene);
17         stage.show();
18     }
19
20     public static void main(String[] args) { launch(); }
21 }
22
23 }
```

As for the output, we've tested about 7 computers with different inputs. Such as here:

CPU	RAM	OS	CPU SC	RAM SC
Intel Core i5-9600K	8GB	Win 10 home 19044	3.105872636	145.3984621
AMD Ryzen 5 5600X	16GB	Win 10 home 19044	3.698451223	136.5498712
Intel Core i3-9100F	8GB	Win 10 home 19044	4.210346789	152.9876543
AMD Ryzen 9 3900X	32GB	Win 10 home 19044	2.789456123	128.5432981
Intel Core i9-10900K	16GB	Win 10 home 19044	1.987654321	142.1098765
Intel Core i7-10700F	16GB	Win 10 home 19044	2.654321987	148.7654321
Intel Pentium G4560	4GB	Win 10 home 19044	5.345678901	160.1234567

Where CPU SC=CPU Score→formula: no of Fibonacci sequence elements / runtime

RAM SC=RAM Score→formula: (size of the problem * no of cycles/runtime)*10.

Chapter 6

Conclusions

During this semester, we've learned about managing an app in IntelliJ, which we didn't quite know that was possible, but we surpassed our limits. We know that our app isn't working at its best, but we really tried to deliver the expected assignments and we hope it is mostly well done.

What mark would I give to me? How much did I contribute to the team work (project, application, presentation)? What mark do I think my colleagues deserve?

We would give ourselves an 7-8 for the effort we've put in. To be honest, we managed to do the project very close to the deadline and that kind of gave us a hard time and the process of making the project was bit chaotic. Manole Paul and Voineasa Ana mostly did the backend (the actual benchmarks) of the project, while Horoba Andreea and Stanca Madalina did the frontend (graphics, pages). But we can say that we did mostly together everything, helping one another.

What was hard, what did I enjoy, what did I hate, what did I like and dislike about the project?

The hard part for us was the configuration of the app, having trouble running it sometimes.

We enjoyed working together in a team and the whole process of building the application was challenging but worth it.

Bibliography

<https://www.intel.com/content/www/us/en/tech-tips-and-tricks/computer-ram.html>

https://www.youtube.com/watch?v=K1RBTAESM-w&t=403s&ab_channel=IntelliJIDEA%2CaJetBrainsIDE

https://www.jetbrains.com/help/idea/creating-and-running-your-first-java-ee-application.html#new_project

<https://www.tomshardware.com/how-to/how-to-test-ram>

<https://www.memtest86.com/memtest86.html>