**An-Najah National University** جامعة النجاح الوطنية

**Faculty of Engineering and IT** كلية الهندسة وتكنولوجيا المعلومات

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

­­­

**Computer Engineering Department**

**Course name - number: Operating Systems - 10636451**

**Report Grading Sheet**

|  |  |
| --- | --- |
| Instructor name: Dr. Sulaiman Abu Kharma | Assignment 2 |
| Academic year: 2022/2023 | Semester: Second |

|  |  |
| --- | --- |
| **Students** | |
| Amjad Kayed - 12028467 | Riham Katout - 12029366 |

Contents

[ Part1 – Code: without synchronization 3](#_Toc134748172)

[ Part1 – Result 5](#_Toc134748173)

[On windows11 OS 5](#_Toc134748174)

[On centOS8 6](#_Toc134748175)

[ Part2 – Code: with synchronization 7](#_Toc134748176)

[ Part2 – Result 10](#_Toc134748177)

[On centOS8 10](#_Toc134748178)

[On windows11 10](#_Toc134748179)

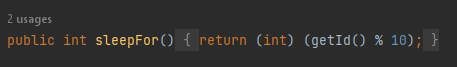
[ Compression between centOS & windows11 11](#_Toc134748180)

# Part1 – Code: without synchronization

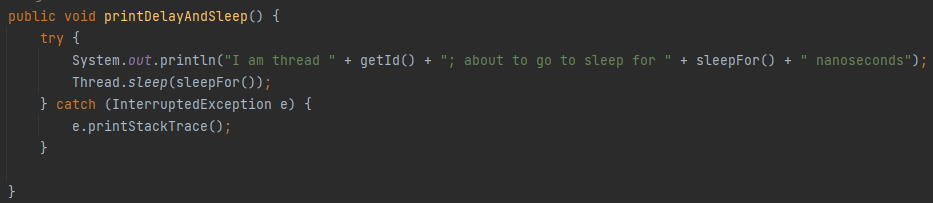
**In the first part, we have two classes:**

1. **“Part1Thread” class**, which represents the thread that will increase the value of the shared memory. It has these following methods:

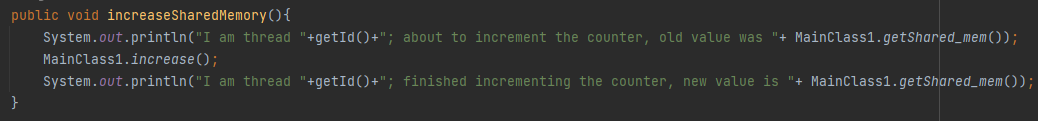
* sleepFor – returns the time the thread will sleep



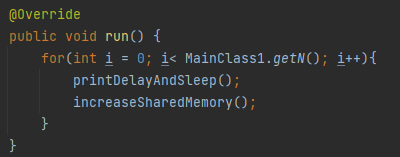
* printDelayAndSleep – prints the requested message then calls “sleep” method for the thread.



* increaseSharedMemory – to call increase method from the main so we can increase the value of shared\_mem



* run, this method will automatically be called when the thread starts



This is the main function in the class which will call “printDelayAndSleep” & “increaseSharedMemory” methods **N** times for each thread

1. **“MainClass1”** class, which contains:

* shared-mem, its methods (getter & setter), the variable N and its getter



* Main method



First, we created an array of threads with size **N**, then created **N** objects of our class Part1Thread, assigned each one to a pointer in our threads array and start it using “start” method.

The next step is iterating over threads array, and calling “join” method, so the parent thread won’t continue running until all the **N** threads finish their work.

Finally, we printed the expected value (= ; we have **N** threads each will call “increase” method **N** times), and the real value we have.

# Part1 – Result

## On windows11 OS

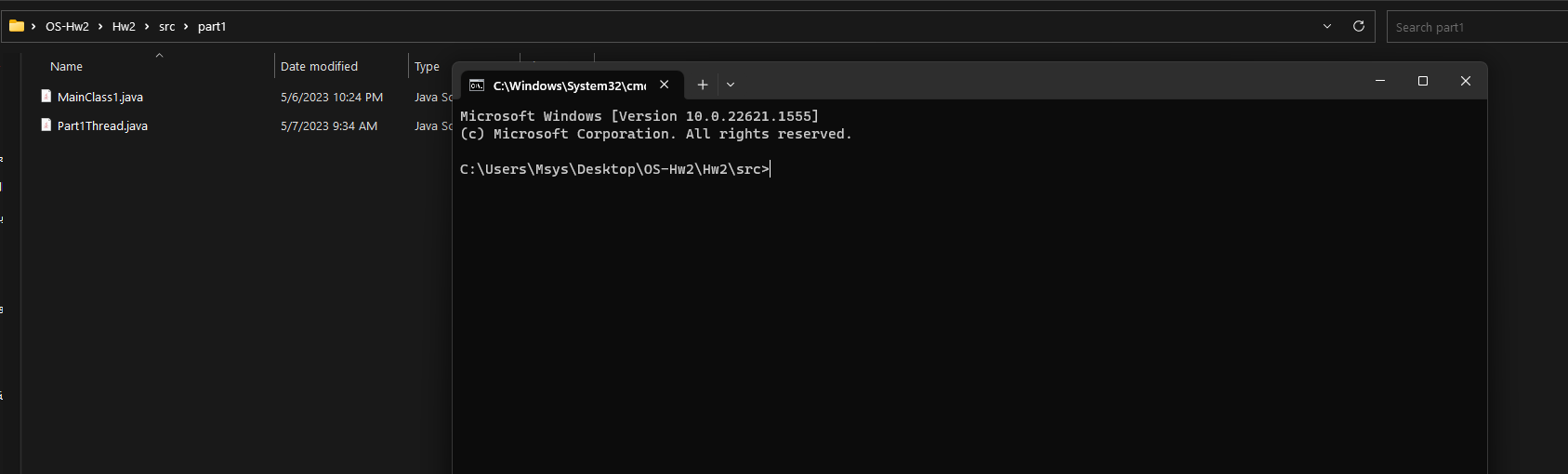


Figure 1: Part1 classes on windows before compiling.

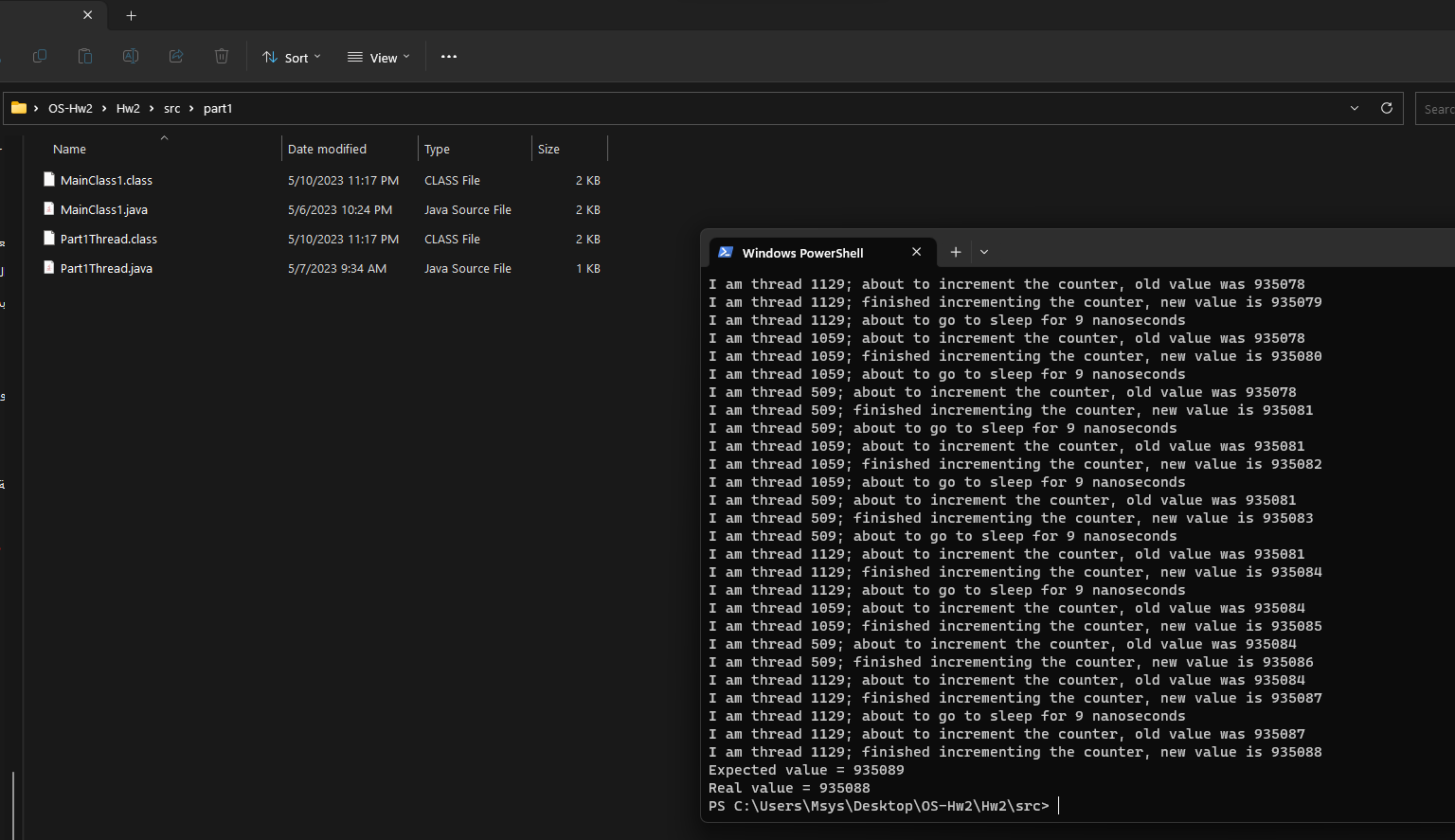


Figure 2: Part1 compiling and run result.

* ­as we can see, the real result is different than the expected, that is caused by multithreads calling ”increase” method at the same time.

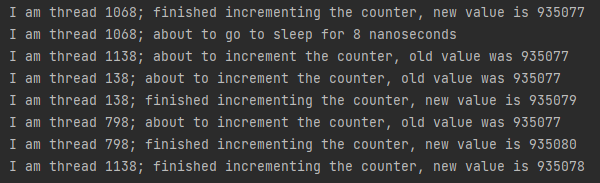


Figure 3: Error example

* we notice that there are 3 threads called “increase” method at the same time.

## On centOS8

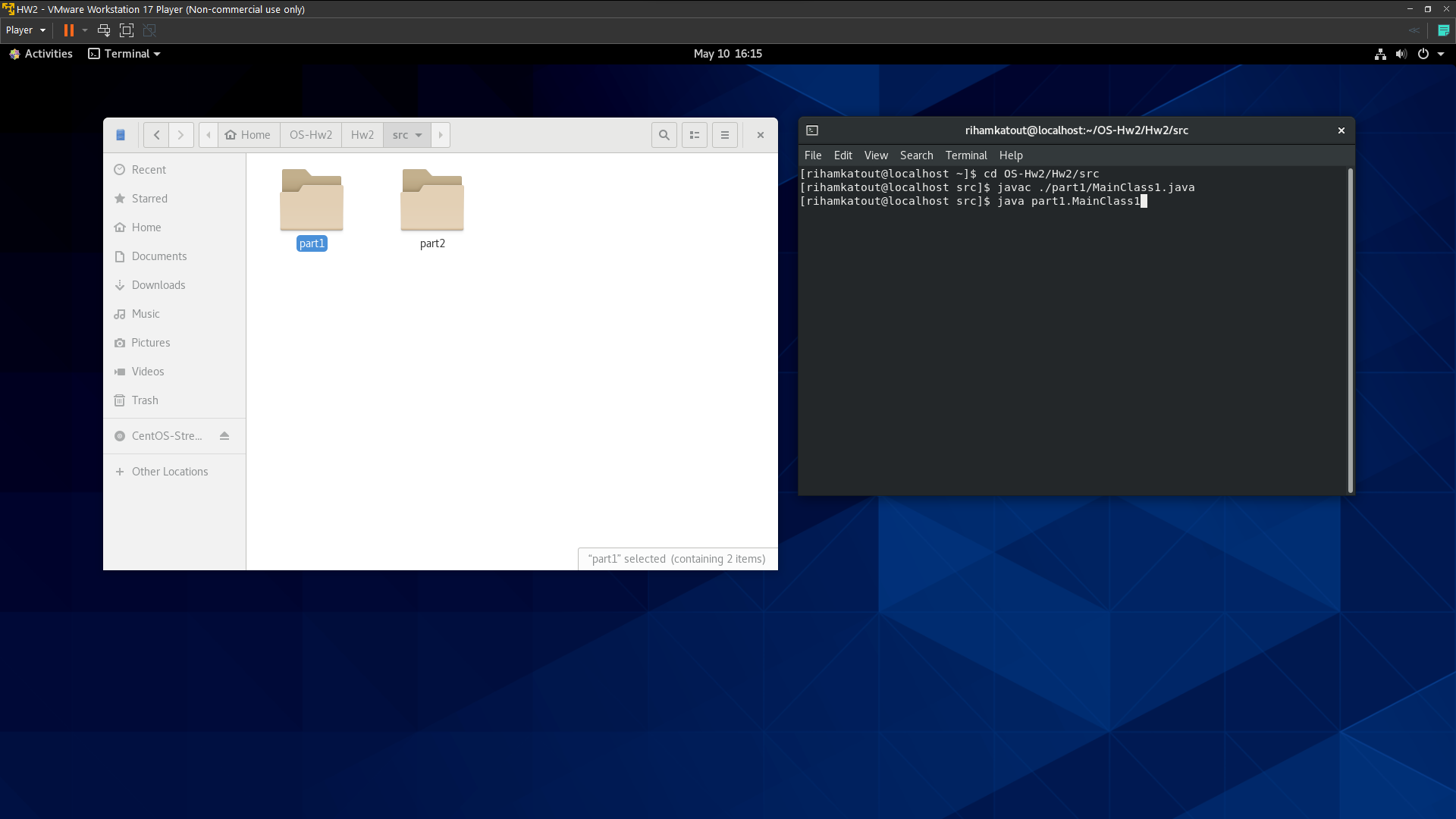


Figure 4: files of the project on CentOS

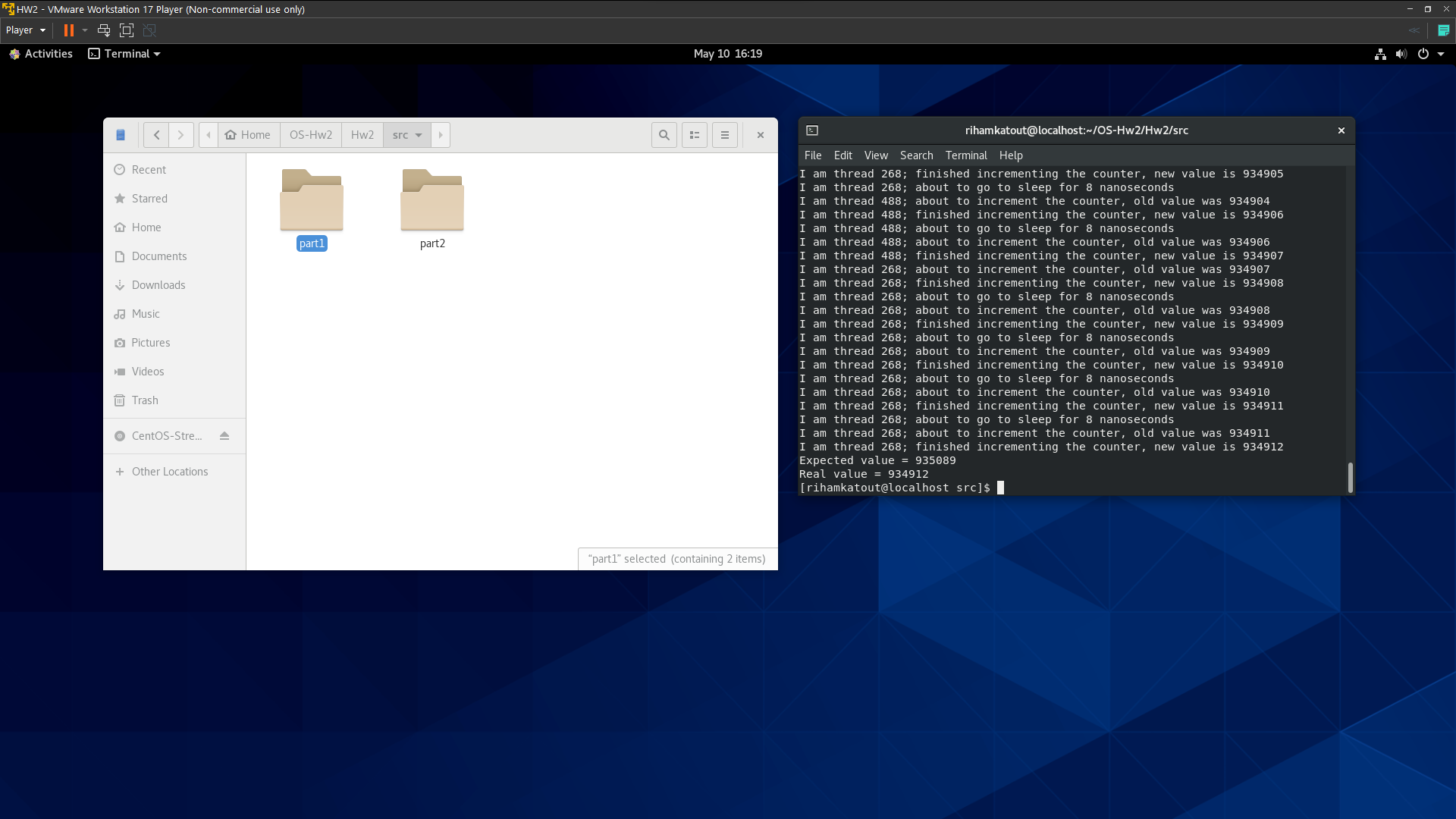


Figure 5: Compile and run result on CentOS

* Result is same to the one on windows

# Part2 – Code: with synchronization

We used the third method (static synchronization) according to our IDs (12029366 + 12028467) % 3 = 2

**In the second part, we have three classes:**

1. **“SharedMemory” class**, contains shared\_memory, its getter and setter.

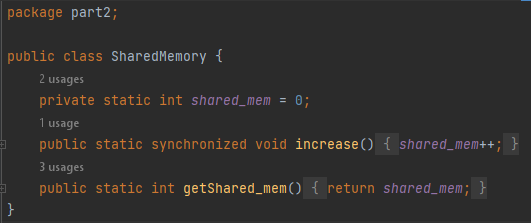
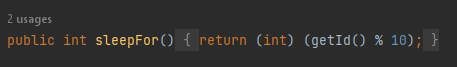
****

Figure 6: ShaedMemory Class contents

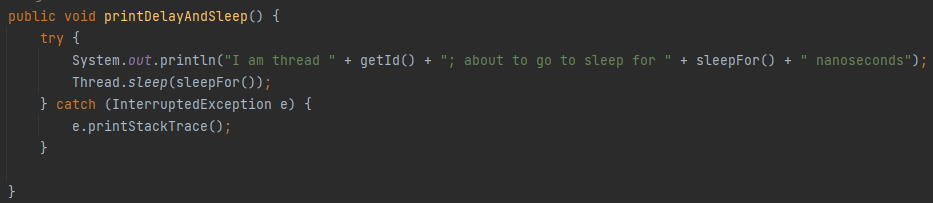
* increase method is static & synchronized, that means that it is running only by one thread at the same time, the other threads will be waiting in a queue, so we can guarantee that we will get the correct value of it.

1. **“Part2Thread” class**, which represents the thread that will increase the value of the shared memory. It has these following methods:

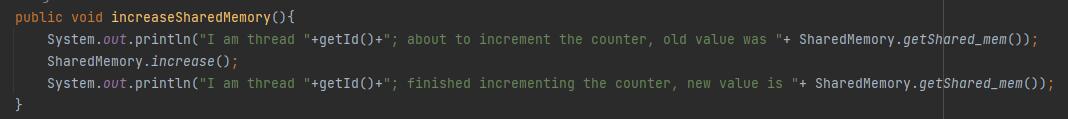
* sleepFor – returns the time the thread will sleep



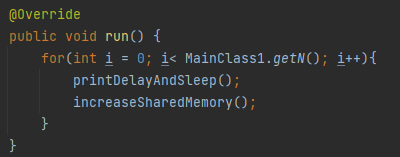
* printDelayAndSleep – prints the requested message then calls “sleep” method for the thread.



* increaseSharedMemory – to call increase method from SharedMemory class



* run, this method will automatically be called when the thread starts



This is the main function in the class which will call “printDelayAndSleep” & “increaseSharedMemory” methods **N** times for each thread

1. **“MainClass2”** class, which contains **N** and the main method only.

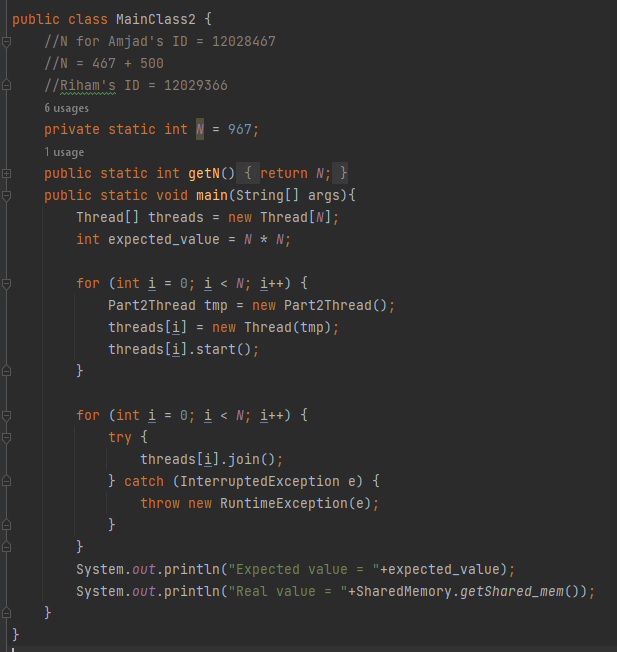


Figure 7: MainClass2 - part2

The process is same to part1, but instead of calling the value of shared\_mem from the main, we called it from SharedMemory class.

# Part2 – Result

## On centOS8

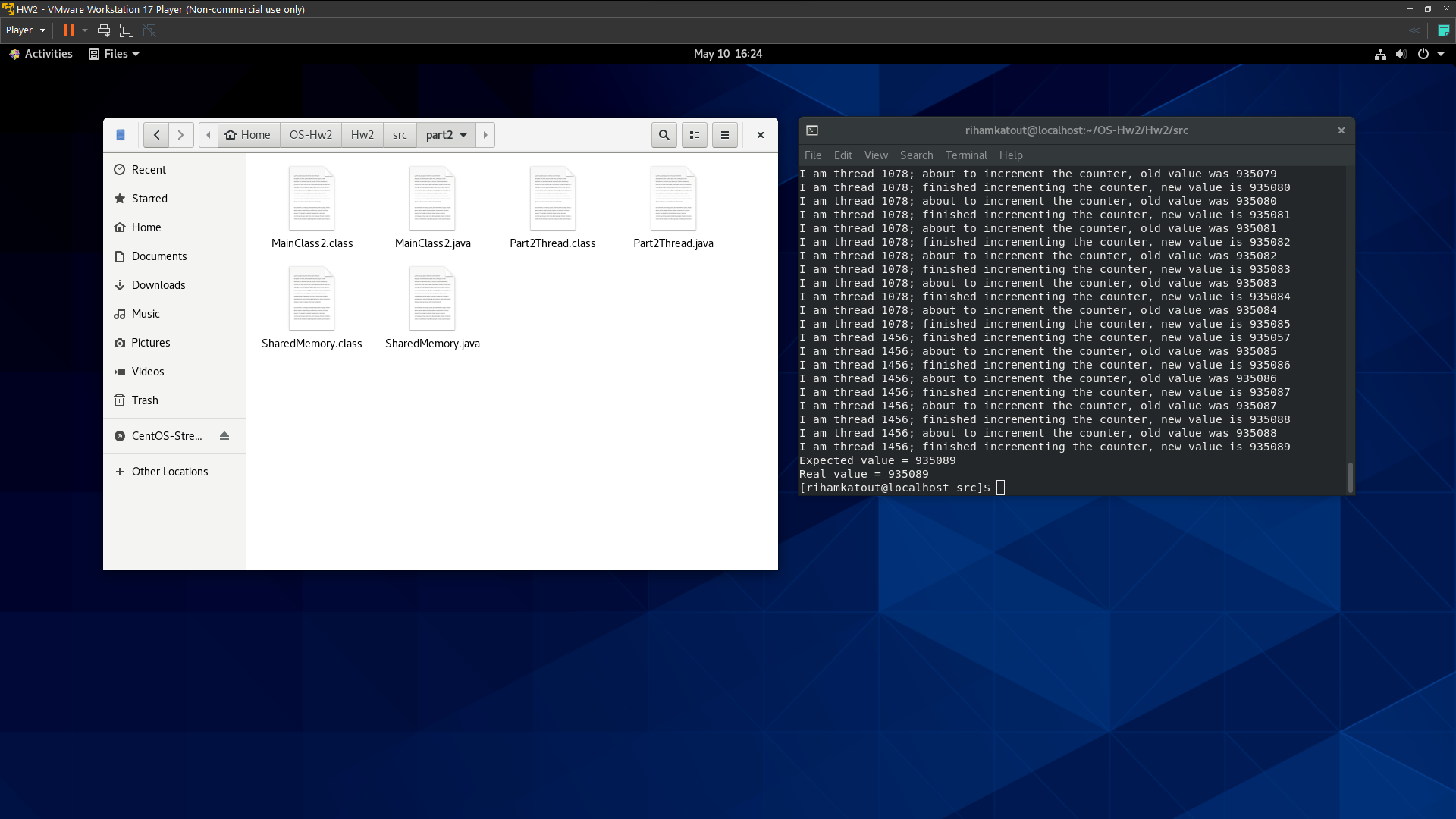


Figure 8: running part2 on centOS

## On windows11

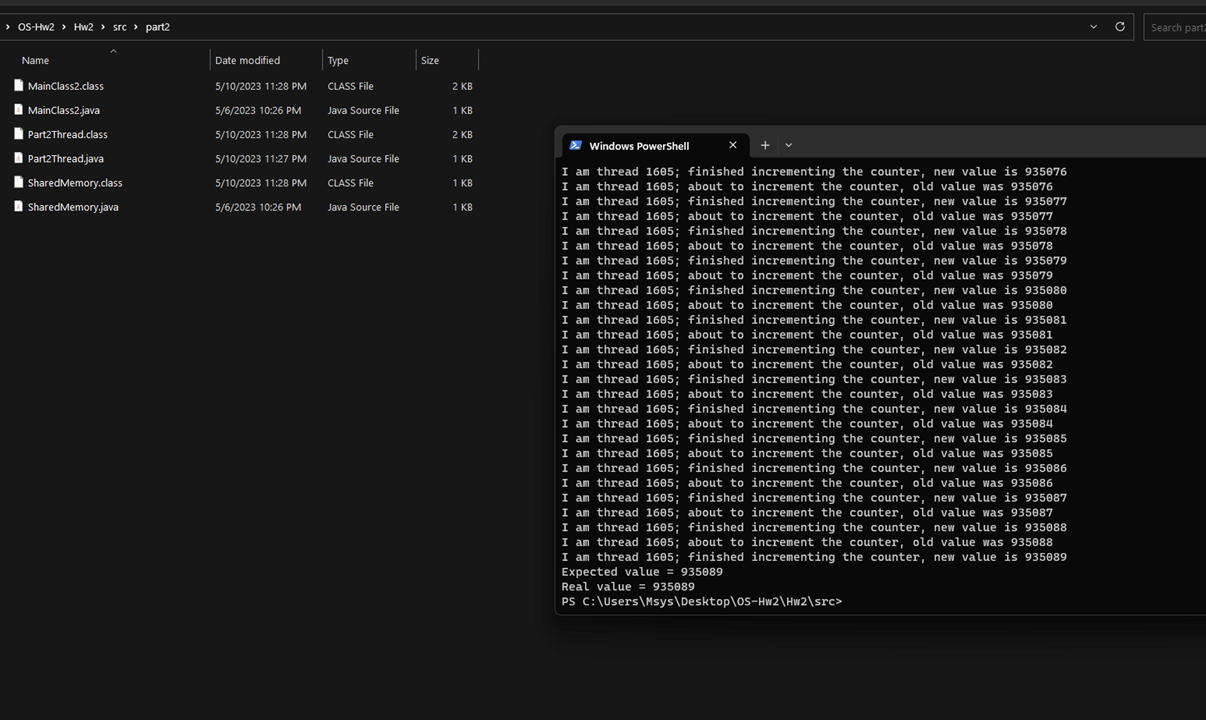


Figure 9: running it on windows

* We got the same value as we expected 😊

# Compression between centOS & windows11

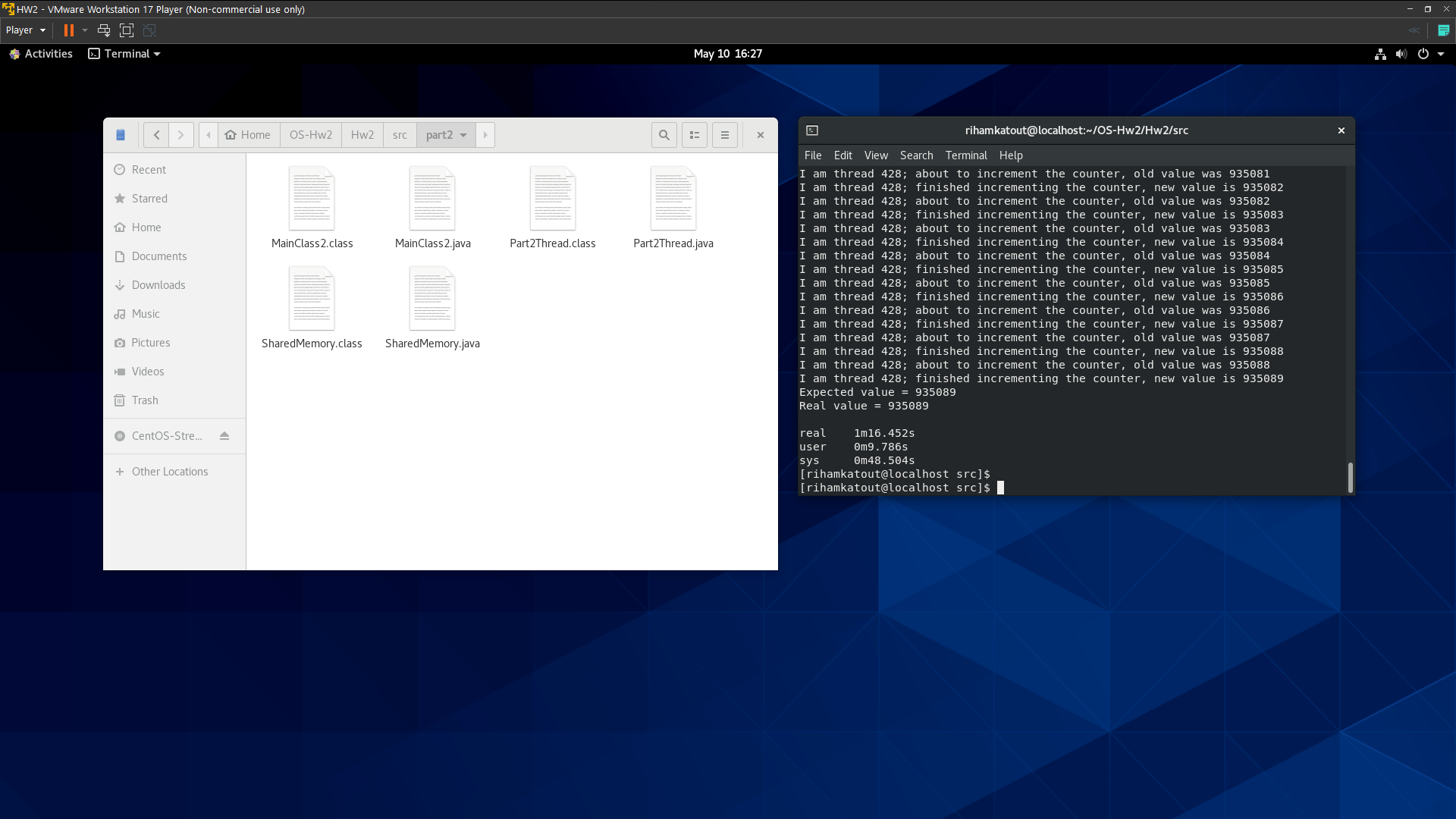


Figure 10: CentOS using time command

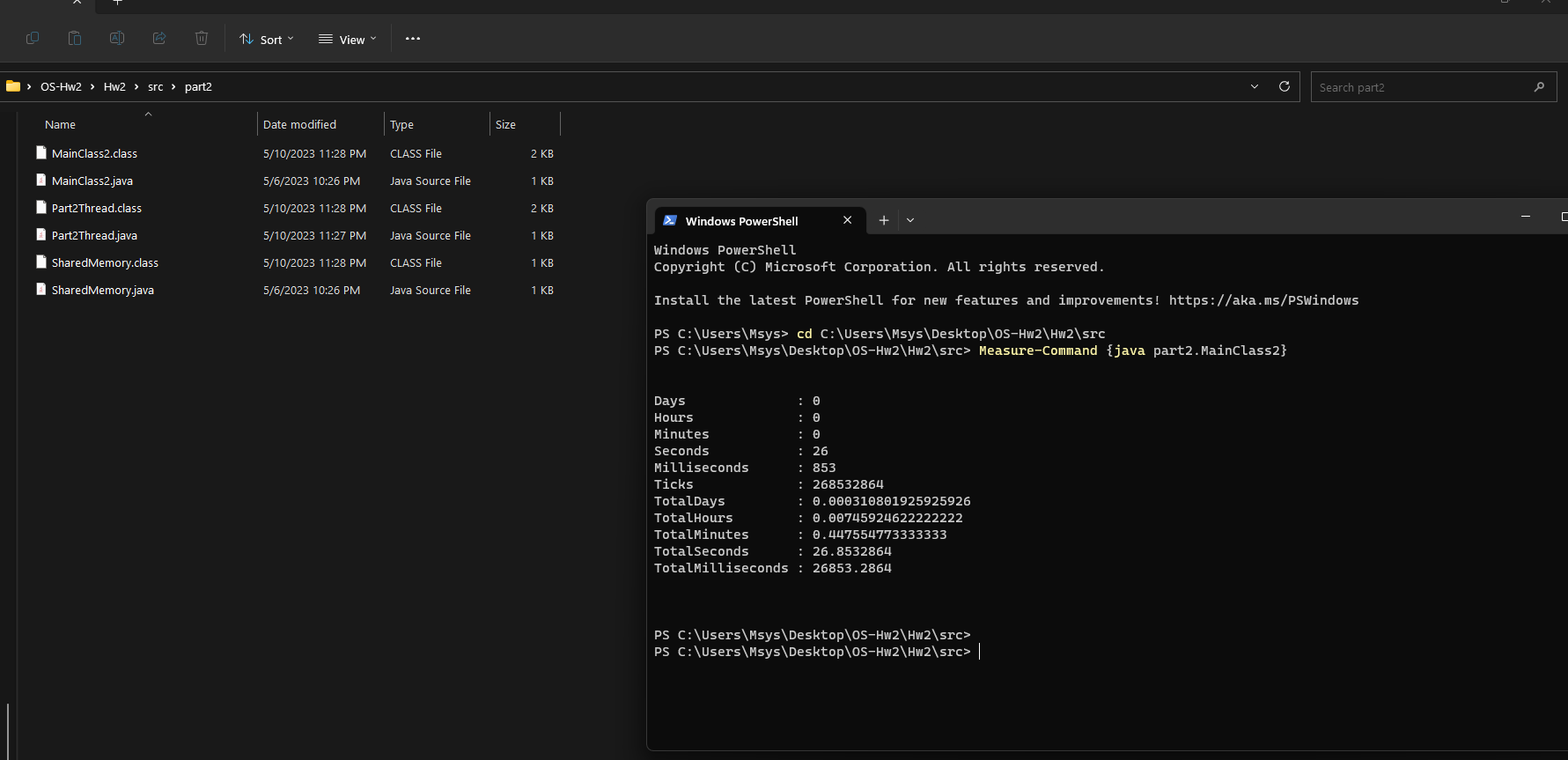


Figure 9: Windows using measure-command command

* It’s clear that Windows is approximately 3times faster than CentOS VM, this is due many reasons.
* **Why Windows is faster?**

1. **Hardware configuration:** the host machine has better compatibility, such as specific drivers or hardware features that enhance Windows performance.
2. **Virtualization software:** different virtualization platforms may have varying levels of performance optimization for different operating systems.
3. **Resource allocation:** the amount of CPU, memory, and other resources allocated to each VM affect its performance
4. **Software configuration:** the specific software packages and configurations on the CentOS VM may have performance implications. Suboptimal settings or configurations on CentOS could lead to reduced performance compared to a well-optimized Windows setup.