Tema de casă

- The Great Sorcerer saves the world -

January 12, 2020

Student: Anghel Florina-Cătălina

Group: CR3.1A

Year: 3

Section: CR

1 Problem statement

The Great Sorcerer owns multiple covens scattered throughout the world in hidden places (every year he tries again to save the world). He has demons in each coven creating potion ingredients, he has witches awaiting the potion ingredients to be made to create potions and he desperately needs his plan to work. With potions he can defeat all the undeads that will rise. Elements:

- Covens:
  - Random number of Covens (between 3 and 20) that create different potion ingredients
  - Each Coven creates an unlimited number of potion ingredients of different types [there are 10 different possible potion ingredients]
  - All covens are in matrix like form (random NxN, where N is between 100-500)
  - Each coven contains an array list with demons
  - There can be no more than N/2 number of demons in a coven
  - Covens every 10 seconds will announce its demons to stop for 1 second, scaring all demons and making them drop their ingredients
- Demons:
  - Demons are spawning randomly in each coven at a random place in the coven (random time between 500-1000 milliseconds)
  - When demons are spawned randomly they need to tell to the coven that they are there
  - No two demons can be on the same position
  - Each demon works independently of any other demon
  - Demons create potion ingredients by:
    - Moving in one direction (left, right, up, down)
    - When they reach a wall of the coven they move in any other direction than the wall's
      - Also, for 2 extra turns they will not be able to create any ingredient in this case
      - After 10 walls hits demons will be demoted with 100 levels on their social skills (see below what social skills means)
      - Once they move they also create an ingredient of a different random type from the list of possible ingredient types
      - If a demon is surrounded by other demons and cannot move it stops for a random time (between 10-50 milliseconds)
      - When a demon encounters another demon while trying to move, they both increase their social skills rating

· The maximum number of ingredients possible to be created at every turn by a demon is 10
· With every 100 levels of social skill ratings, the demons are promoted and thus able to create 1 extra ingredient on every move
· When the ingredient is created they update their coven to know what ingredient(s) they created
· After the demon creates an ingredient, it needs to rest for a short while (30 milliseconds)
· Demons will work like in... forever. As such they don't need to stop.

- Witch:
  - Witches await to receive ingredients from the coven.
  - They can always read from the coven the number of ingredients (and which ingredients), but they won't be allowed access when demons notify the coven about new ingredients
  - A maximum of 10 witches can read from a coven at a time (this is also the maximum number of witches the Grand Sorcerer has)
  - A random time must pass between two consecutive coven readings
  - When a witch visits the coven it will get available ingredients, put them in its magic bag and create potions out of them
    * There are 20 different possible potions, each requiring between 4-8 ingredients of specific types so that it is created
    * Each potion can be created in a random number of time (between 10-30 milliseconds)
    * In order for the witches to not get burden with ingredients, find a proper mechanism to read only as many ingredients as possible to create potions, while also considering to not have too many ingredients per coven (see fairness)
    * It is left for you to think about potential potions and their ingredient list
  - Witches will put all potions through a magic portal and send them to the Grand Sorcerer
  - Multiple witches can at the same time put the potions through the magic portal while the Grand Sorcerer reads by itself all potions sent through the magic portal
  - Witches are known from the beginning
- Undead:
  - There is a fixed number of undeads at the beginning, a random number between 20-50

  - Every random time (between 500-1000 milliseconds), they randomly visit covens
  - Every visit at a coven, per undead, is translated as follows:
    * A random number of demons, between 5-10 is retired, being too scared to work anymore
    * Current available ingredients are lost
  - If there are witches visiting the coven then:
    * A witch will fight an undead
    * She will request 2-5 potions from the Grand Sorcerer to fight the undead
    * If there are available potions, then the undead will run scared
    * If not, for every undefeated undead, 10% of the ingredients available in the coven will be lost, but no demon will be lost

- The Grand Sorcerer's Circle
  - this Circle contains all covens
  - it creates the covens
  - it creates the Grand Sorcerer helper that spawns demons and gives them a random coven to work in (remember that each demon is responsible to register itself to the coven)

2 Implementation

The ingredients used to make the potions are:

- *Anethum graveolens*
- *The legs of the Funnel-web spider*
- *Student's tears*
- *Ambrosia seeds*
- *The skin of a Black Mamba snake*
- *The venom of an Inland Taipan*
- *A crow's claws*
- *Sand from the Mariana Trench*
- *Aloe Vera*
- *A rock from the Vesuvius vulcano*

The potions are:

- *Healing Potion*
- *Fearless Elixir*
- *The Sun's Elixir*
- *The Moon's Elixir*
- *The Death's Taste*
- *The Venom*
- *Black Armour*
- *Anti PTSD Elixir*
- *Future Reading Potion*
- *Localisation Potion*
- *The strongest shield*
- *The indestructible sword*
- *Light-Speed Elixir*
- *Invisibility Potion*
- *Undead Camouflage*
- *Dead for an hour*
- *Teleportation Potion*
- *The strongest for 60 seconds*
- *Sleeping Potion*
- *Freezing Potion*

The recipes for the potions will be randomly created and every recipe requires a number between 4-8 of ingredients.

For this homework, I have implemented multiple Java Projects, one for every task:

- The project called TheGreatSorcererWorkingDemon is used to solve the problem with the demons that are producing the ingredients and then they are resting for 30 milliseconds.
- The project called TheGreatSorcererRetireManager is used for the first extra task, where the demons ask a retirement manager to be retired and then, after they are marked as retired, they are asking the coven to be removed from the coven's list of demons.

- The project called TheGreatSorcererDiagonalSleepingSemaphores is used for the second extra task, where every demon will sleep when he is on the diagonal and he will wake up when all the current coven's demons are sleeping. This is the version with N+1 semaphores.
- The project called TheGreatSorcererDiagonalSleepingCyclicBarrierDemon is used for the third extra task, where the demons sleep when their room is on the diagonal This is the version with a CyclicBarrier object. The demons will sleep and they will wake up after all the current coven's demons are sleeping.
- The project called TheGreatSorcererDiagonalSleepingMyCyclicBarrier is used for the first extra task, where the demons sleep when they move into a room that is on the diagonal of the matrix of rooms. They will wake up after all the current coven's demons are sleeping.

2.1 The classes used in all the implemented projects:

2.1.1 The class GrandSorcerer

This class represents the Great Sorcerer, who communicates with the witches using a TCP/IP "magic portal". The Grand Sorcerer can receive a message through the portal and this message can be a potion or a help request. If the Sorcerer receives a potion, he will add it into his potions list and if he receives a help request, he will generate a random number and he will check to see if he has enough potions to send (he needs to send a number of potions equal with the randomly generated number). If he has enough potions, he will send them and if he does not have enough potions, he will send a message used to notify the witch that he can not help her.Here, the Grand Sorcerer uses the portal to communicate with the witches and the portal is implemented using a ServerSocket object. Using this socket, the connection with a port is made, in this problem, the port's number is 4999. After doing this, the Great Sorcerer will wait until the Sorcerer Circle is connecting to the same port. When the Sorcerer Circle is connected, the portal is functional and it can be used.In this program, the Great Sorcerer is the server and the Sorcerer Circle is the client. Now, the portal can be used and the Grand Sorcerer waits to receive a message.

2.1.2 The class SorcererCircle

This class represents the client used to make the connection between the Grand Sorcerer and the witches. It will be connected to the port 4999 using the localhost. This calss is also used to create:

- the PrintWriter object used by the witches to send the messages.
- the BufferedReader object used by the witches to receive the messages.
- the covens - a random number of covens between 3 and 10.

- the witches - there are 10 witches
- the Witch Craftbook - the book that has the recipes for the potions.
- the undeads - a random number of undeads between 20 and 50.
- the manager that creates the demons
- the retirement manager(just in the project called TheGreatSorcererRetireManager)

2.1.3 The class Coven

This class represents the coven and it extends the class Thread. Every 10 seconds, a coven will scare its demons and it will make them to lose their recently produced ingredients that were not sent to the coven. This class has the list of the ingredients produced by the demons and these ingredients will be taken by the witches. In order to access the ingredients, the witches and the demons need to acquire the semaphore called ingredientsSem. Because they needed to be synchronized, I used monitors to make sure that the demons and the witches will have access to the ingredients. While it is the witches' turn, the demons will wait until it is their turn. It is the same for the witches. This class has a matrix of rooms and it is modified when a demon changes his room, a demon is spawned in the coven or when a demon is retired and he leaves the coven(just in the project TheGreatSorcererRetireManager). The method visit() is used by the witches when they want to take some ingredients from the coven. Firstly, the current witch will be added to the vector of visiting witches and this is done by using a lock. The shared resource will be locked and then the witch will be added to the vector of witches and then the resource will be unlocked in order to be accessed by other witches. If it is the witches' turn, the witch can take the ingredients just after she acquires the semaphore called ingredientsSem. Then the witch will take the ingredients that are different than the ingredients she already has. Then, if the coven is attacked, the witch will acquire the semaphore for undeads in order to remove an undead from the vector and she will fight with him. She will ask for help from the Grand Sorcerer and she will wait for a response. If she receives some potions from the Grand Sorcerer, the undead will be scared and he will run away and the ingredients and the demons will be saved. If the Grand Sorcerer does not have potions, then 10% of the ingredients are gone. Then the witch will leave the coven and she will try to make some potions. The method attack is used when an undead attacks the coven. Firstly, the undead will be added into the vector of undeads but after the semaphore for undeads is acquired. After the undead is added, the semaphore is released and the undead checks to see if there is a witch in the coven. If there is a witch, the undead will sleep for 1 second. He will be removed from the coven by the witch that fights him. If there are not witches into the coven, then a random number between 5 and 10 of demons will be scared and they will not work anymore and all the coven's ingredients are gone. Then the undead will leave the coven.

The method changeRoom is used when a demon changes his room. Firstly, he will acquire the rooms semaphore in order to leave the old room and to go into the new one and then he will release the semaphore. The method receiveIngredient is the method used by a demon when he wants to send the produced ingredients to the coven. If it is the witches' turn to get the ingredients, the demons will wait until it is their turn to send the ingredients. Firstly, the current demon will acquire the semaphore for the ingredients list and then he will add his ingredients into the ingredients list. Then he will release the semaphore and if there are witches that are visiting the coven and there are ingredients into the list of ingredients, turn will become true. Then the witches can take the ingredients. The method addDemonIntoTheCoven is used by a demon when he wants to be added into the coven. Firstly, the demon will wait in line to be serviced then he will acquire the semaphore for the list of demons and he will let the next one in line to be serviced. Then the demon is added into the coven's list of demons and the demons semaphore is released. Then he will want to go in his room assigned by the Demon Distribution Manager. He will acquire the rooms semaphore and he will "enter" in that room, then he will release the rooms semaphore.

2.1.4 The class Demon

This is the class Demon which extends the class Thread. The method moveUp() is used by the demon when he tries to move up. Firstly, the row is decremented. If the row < 0, the demon kicked a wall and he will go back in the previous room. If he kicked 10 walls, the demon will be demoted with 100 levels on their social skills and the number of kicked walls is set to zero. If row ≥ 0, then the demon will check to see if there is someone in the targeted room. If there is another demon in the desired room, the demon will go back in his previous room and they both increase their social skills rating. Else, if the targeted room is empty, the demon will go there and a local variable(initially set to false) will be set to true. The methods moveDown(), moveLeft() and moveRight() are similar to the moveUp() method. The method createIngredients is used by the demons to create ingredients. This method is used to make the ingredients produced after just one move. The demons can create maximum 10 ingredients at once depending on their social skill. The method run() overrides the method with the same name from the class Thread. While the demon is not scared, he will try to change his room. If he can change it, he will create ingredients and he will send them to the coven. Then he will rest for 30 milliseconds. If he can not move, then he will sleep for a random time (between 10 and 50 milliseconds). In the project TheGreatSorcererDiagonalSleepingSemaphores, the class coven will have capacity/2 + 1 semaphores: one common barrier semaphore, which demon threads release when they reach the synchronization point, and an array of "continue" semaphores, indexed by thread, which threads acquire in order to continue beyond the barrier and in the project TheGreatSorcererDiagonalSleepingCyclicBarrierDemon, the coven will have its CyclicBarrier object and the demons will sleep when they will go into a diagonal room and they will call

the method await() for the barrier and when all the demons are asleep, all of them will woke up. The barrier will receive as the number of parties the coven's capacity/2.

2.1.5 The class DemonDistributionManager

This class is used to create the demons and to spawn them into a coven. It extends the class Thread. The method run() overrides the method with the same name from the class Thread. The manager will spawn a demon into a random coven at a random time between 500 and 1000 milliseconds. He will try to find an empty room for the demon and he will create him and put him to register into his coven.

2.1.6 The class Potion

This is the class Potion and it is represented by a recipe and a name.

2.1.7 The class Undead

This is the class Undead. It extends the class Thread.The method run overrides the method with the same name from the class Thread. The undead will sleep for a random time(between 500 and 1000 milliseconds). Then he will attack a random coven.

2.1.8 The class Witch

This is the class Witch that extends the class Thread.The method getHelpFromTheSorcerer() is used by a witch to get help from the Great Sorcerer. It is used when the witch fights the undead. If the portal is not used, the witch will send a message through it and she will wait for the response. There are two types of responses:

- "I don't have enough potions to give you. Good luck!" - the sorcerer can not help her
- A potion - she will receive some potions and she will fight the undead.

The method makePotion is used by a witch to make a potion. Firstly, she choose a potion and she tries to make it. She checks if she has all the ingredients and if she has all of them, she will make the potion. The method run() overrides the method with the same name from the class Thread. The witch will visit just the covens that have ingredients. If she has more than 4 ingredients, she will try to make a potion using one of the 20 available recipes. If the witch made potions, she will send them to the Grand Sorcerer through the portal. And then she will let the waiting witches to use the portal.

2.1.9 The class WitchCraftBook

This class represents the Witch Craft Book. It has all the recipes used to make the potions. The method createBook() is used to create the book's recipes. It creates 20 recipes with distinct ingredients. Every recipe will have a number of ingredients included in the interval [4, 8].

3 The classes used just in some of the projects:

3.1 The class DemonRetireManager

This class is used in the project TheGreatSorcererRetireManager. It is used to retire the demons. It extends the class Thread and the method run() overriedes the method with the same name from the class Thread. Firstly, the manager will release the semaphore from the class Demon called retireSem and he will let the demons to send him the retirement requests. Then, in a infinite loop, he will sleep for a random time (between 200 and 300 milliseconds) and then he will check if he has demons that sent retirement requests. If there are requests, he will pick a random demon and he will mark him as retired and then he will remove the demon from the list and he will release the semaphore. The method sendRetireRequest is used to receive the requests and the demons that sent the request will be added into the list. The demons will call the method tryAcquire() and if they can acquire the semaphore, they will be added into this list. Then, they will be marked as retired and they will ask the coven to remove them from the demons list. Then the semaphore will be released to let the next demons to be retired.

3.1.1 The class MyCyclicBarrier

This is the class used to solve the last extra task and the project is called TheGreatSorcererDiagonalSleepingMyCyclicBarrier. This is my custom Cyclic Barrier. It has a constructor which receives the number of parties and it also has the method await used to make the threads wait until all of them arrive to the synchronization point. The covens will have its cyclic barrier and every demon from that coven will have access to it. The demons will sleep when they go in a room that is on the diagonal. They will call the method await() for the barrier and when all the demons are asleep, all of them will woke up. The barrier will receive as the number of parties the coven's capacity/2.

4 What changes you need to make and how to run it properly:

4.1 Settings:

The output is printed in the Console. If you do not want to have a truncated output, you need to go to the Window → Preferences → Run/Debug → Console and uncheck the option called Limit console output like in the pictures 1 and 2.

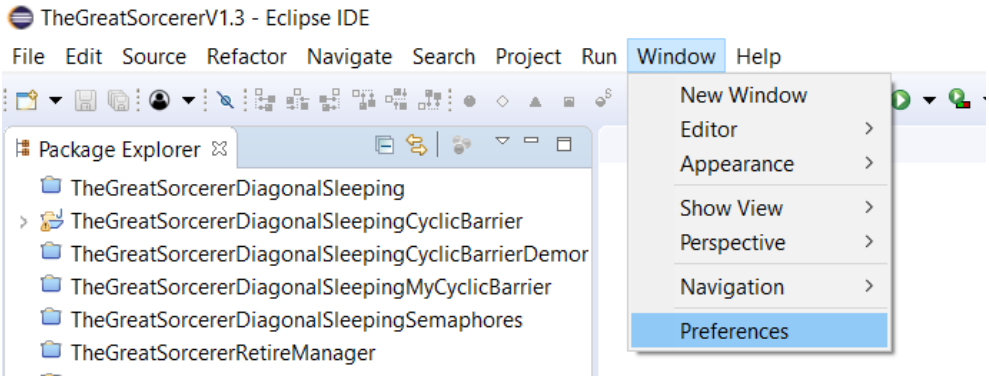

Figure 1: How to make the console bigger - part 1

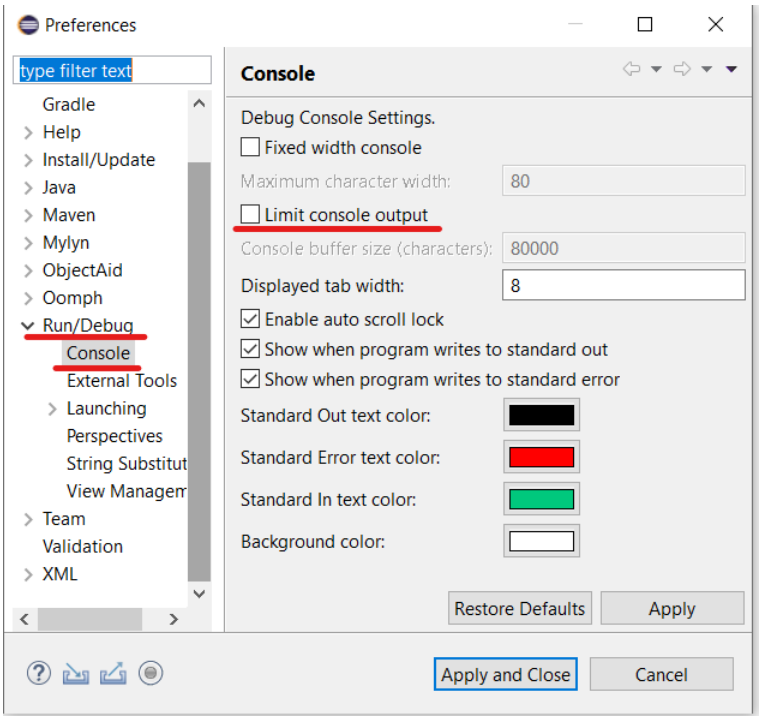

Figure 2: How to make the console bigger - part 2

4.2 How to run it properly:

If you want to see the output from the server and the output from the client at the same time, you need to create one more Console by clicking the button shown in the picture 3.

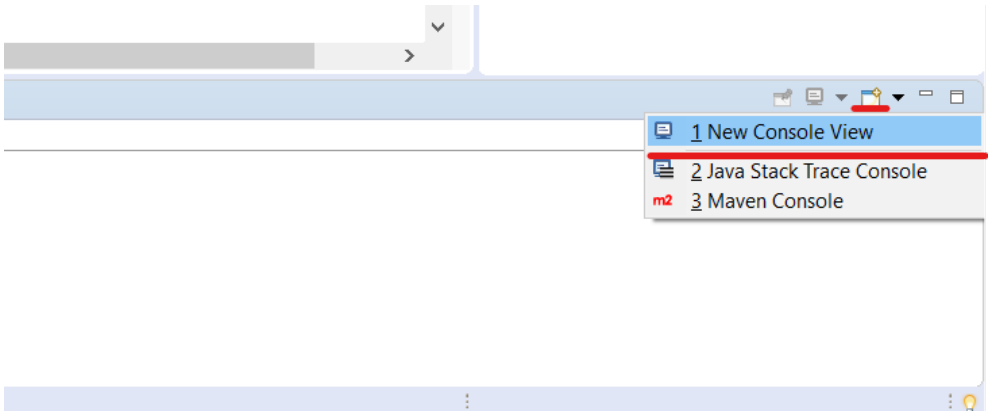

Figure 3: How to create a new Console

Now you have two consoles. Now, to run the program, firstly you need to start the server(run GrandSorcerer) and then the client (SorcererCircle) and then you need what you want to see in every console (if you want the output from the server or from the client). To do that, you need to click on a console and then you need to click the button Display Selected Console marked with the number 1 and the red square in the picture 4. Sometimes, it does not work because Eclipse has some bugs and in that situation, you need to click the button Pin Console marked with the number 2 and the with the purple square in the picture 4 and then the button Display Selected Console. Now you can see how the program works.

Now, you can see output from the server and the client in real time and it is very easy.

To finish the execution of the program, the client needs to be stopped the first and then the server. If the server or the client could not be stopped and the program can not be stopped from Eclipse (it is a rare situation), the socket is still used. To close the socket, open the command line and write the next command: **netstat -ano — find "port_number"** where the port_number is 4999. The result will be similar with the results from the picture 5.

Now, go to the Task Manager and in the Details tab, serach the processes with the PIDs from the command line results(the last column) and select them and then click End Task. See the picture 6.

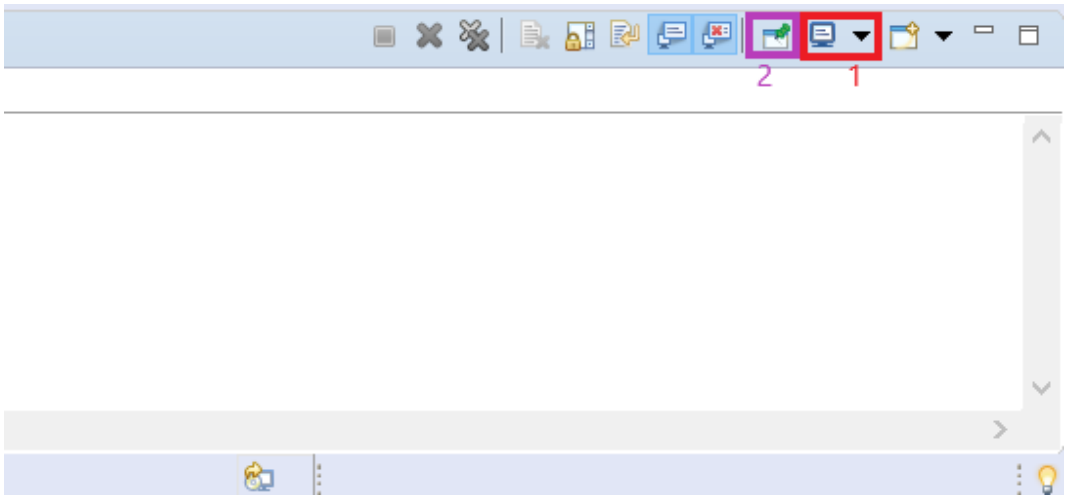

Figure 4: How to show the output from the server and the client

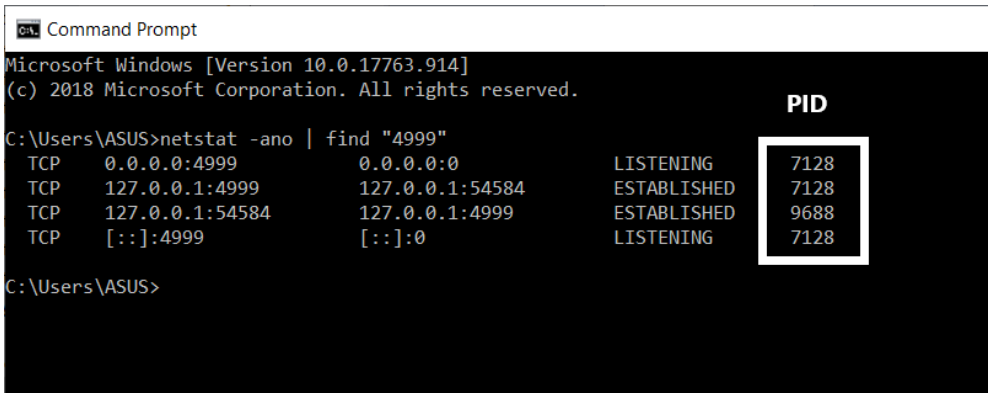

Figure 5: Command line results

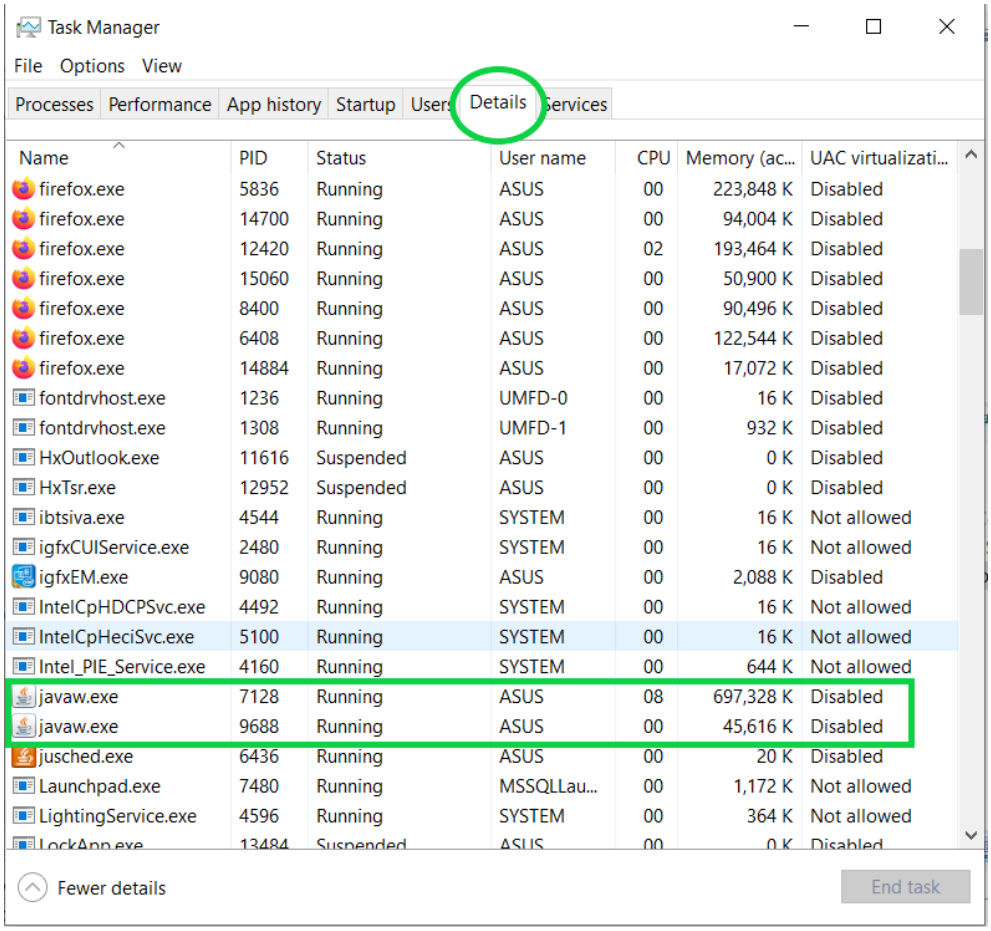

Figure 6: Task Manager

5 Resources

The resurces used to implement this homework are:

- The Little Book of Semaphores
- Cyclic Barrier
- For multiple consoles
- How to close the port from the command line
- TCP/IP Wikipedia

6 Conclusions

This homework was interesting and it made me understand how to use multiple synchronization mechanisms together and how to implement a TCP/IP "portal" and my custom cyclic barrier. I also learned how to make a cyclic barrier using some semaphores. It was a challenging homework, but it helped me learn new things.