Programming Languages TC2006

Teacher: Benjamin Valdes

Student: Mariana Favarony Avila

Final Project:

Ambulances Attending Car Accidents Cali, Colombia

TABLE OF CONTENT

1. Introduction

The present project makes use of concurrency and object oriented programming for creating an ideal scenario where the ambulance that attends a car accident is the one that is closer in time. The use of threads helps to simulate the ambulances and synchronization for simulating how ambulances act on the street, if they are available or attending an accident showing a closer solution to reality.

2.Context of the problem

On average 1,078 car accidents are reported every month in Cali, Colombia. For 2019 there were reported a total number of 12,934 car accidents (Figure 1) in the city. Although the value was less than the one in 2018, it is still a high number and what worries the most is that the ambulances do not attend it on time which directly affects the person involved. We understand by car accident a collision between two or more transports (car, motorcycle, bicycle or bus).

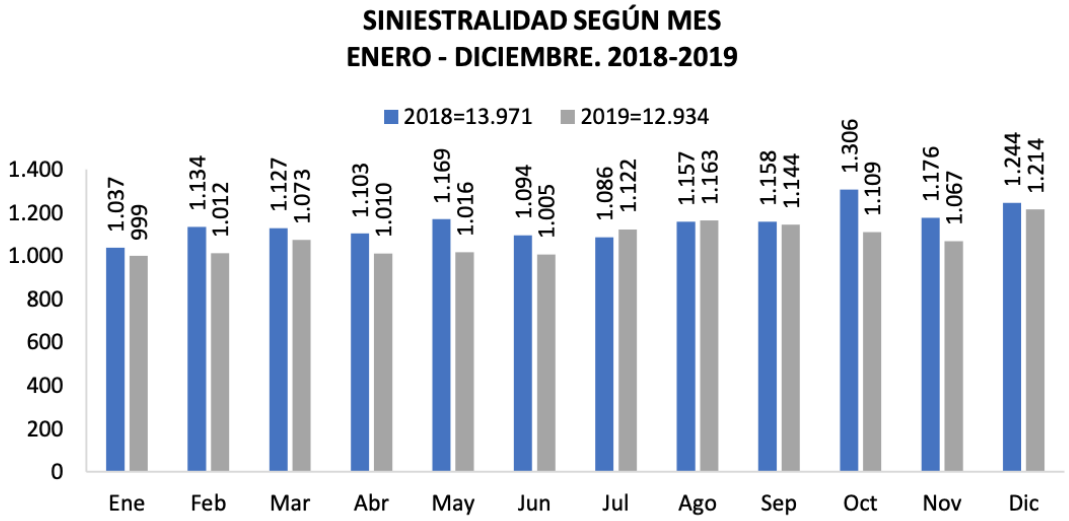


Figura 2. Siniestralidad según mes.

Figure 1: Accidents by month for 2018 and 2019 Cali, Colombia

There are three types of ambulances, each one has its own purpose. Public ambulances are the ones that make part of the pool of ambulances established by the government for public hospitals and whose purpose is to cover the institutional transfer (taking a patient from one hospital to another) and accidents at home. This mainly happens when the patient is

in a hospital level 3 and if it's injury can be treated in a hospital level 2 he/she is taken to the other hospital to set free the hospitals with more equipment in case of an emergency. It can also be the case that the patient is so ill that needs to be treated in a hospital level 4 so he is transferred.

Private ambulances are the ones that are owned by private health entities and cover only people who have their insurance. They attend issues at home and have the service for taking the patient to the hospital for medical exams or medical appointments. Last but not least are the SOAT ambulances. SOAT stands for *"Seguro Obligatorio de Accidentes de Tránsito"*, it is a mandatory insurance for every person who owns a car. This pool of ambulances are the ones in charge of the car accidents. The people who own this type of ambulances need to have special permissions in order to transit.

For the last 2 years it has been reported an issue with the SOAT ambulances because there's evidence that they are in a constant fight for who is attending what accident. The main reason for this is because of the commission behind it. An ambulance is paid depending on the degree of the accident, and some have agreements with certain hospitals that assure an additional commission if they take the patient to that specific hospital.

SOAT ambulances also make part of a business and the more accidents they cover, the higher their income will be. It sounds a little inhuman but that's the reality. The real issue is not that they take it as a business; it is the effects that their fights for an accident leave. The more ambulances compete for arriving first to an accident there will be more probability for another car accident to occur. If two or more ambulances arrive to the same accident they create more traffic in the area, and is it possible that there's another accident that is not being attended. And the worst scenario is that the ambulance says I can arrive but at the end it is so far away that only for winning the commission speeds up and tries to arrive, at the end, the patient and the city is affected. Neither the patient is attended on time nor the traffic is fluid.

3. Solution

For the solution I proposed an ideal scenario where the ambulances are positioned in different areas of the city and the one that is closer in time

will be the one that is going to be assigned for attending the accident. At the same time, the pool of ambulances will be updated, having one ambulance less every time one is assigned to attend an accident. I am aware that it is a problem that may be difficult to solve because of all the social factors added to it which are not controllable, but that's why I proposed the ideal scenario of how it should work. For this project the solution proposed will only cover the part from assigning an ambulance to attend an accident.

4. Implementation:

The project is programmed in Java. Java is a programming language which was designed to have as few implementation dependencies as possible. When we mention programming paradigms, we refer to the features of the programming language. A programming language can be classified into multiple paradigms and the use of each one depends on the needs for the solution of the posed problem. In this case the solution was programmed in java making use of concurrency programming and object oriented programming, both programming paradigms.

Concurrency programming is defined as the ability to run several instructions, several programs or several parts of a program in parallel. This can be executed with threads. A thread is also called a *lightweight process*, it is in charge of its own activities which are established in a stack but it shares data of other threads in the same process. Threads exist within a process, and every process has at least one. In java by default each program runs in a process and can work with many threads to achieve concurrency. An application is considered concurrent with the use of threads, sharing resources and having synchronized methods which can assure that the information that is being shared is the same for all threads.

Object Oriented Programming is defined as a programming paradigm that relies on the concept of classes and objects. This is done with the objective of structuring a program into simple and reusable pieces of code which are called classes and are used and modified in individual instances of objects. Objects are created to access the different methods and attributes of a class but with the specific values that represent the object.

First of all, I'll break up the solution in different parts to make the explanation clearer.

1) Graph

First of all I defined a graph (Figure 2) to simulate a part of the city with different points. This graph has vertices and edges. The vertex represents a specific coordinate and the edges the cost of the distance between vertices. In this case the vertex represents the vertex where the ambulance is at and the vertex where the accident is at and the edges represent the cost of the distance in time from the ambulance location to the accident location. For the test the time is represented in seconds.

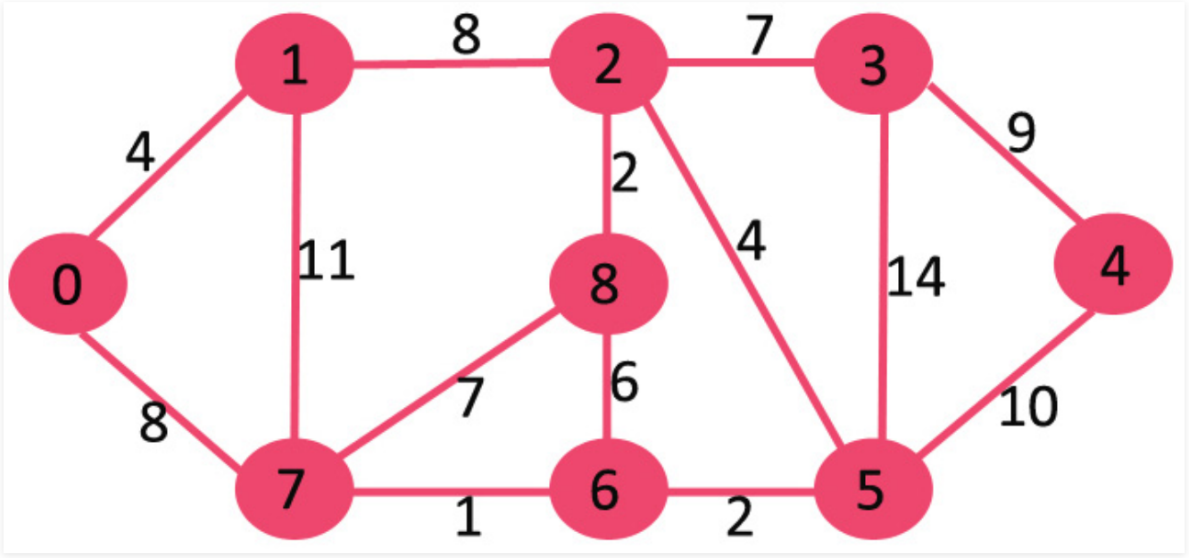


Figure 2: Graph for implementation of the dijkstra algorithm (Geeks for Geeks, 2021)

2) Dijkstra Algorithm

Now that I have the graph as presented in Figure 1 I can now start to define how the shortest path is going to be calculated. The shortest path will be calculated with the Algorithm of Dijkstra, Figure 1 gives us a clear example to understand how the shortest path is being calculated.

Given a vertexA in the graph, in this scenario the vertexA is the accident location, the dijkstra algorithm will find the shortest paths from the verticesN given, in this case the verticesN given are the location of the ambulances to the vertexA. This operation is done by checking all the possible paths from a source(vertex of the ambulance) to a destination(vertex of the accident) and start comparing which one has the least cost, in this case the least number of seconds and that will be output as the shortest path. For example check Figure 1, if the ambulance is in

vertex 8 and the accident in vertex 6 the shortest path will be of 6 visiting directly the vertex 6. Because there's another option to visit vertex 2,5,6 but the cost will be 8 and if it goes through vertex 7,6 the cost will be 8.

3) Defining the ambulances

Now that I know how to calculate the shortest path from an ambulance to the accident location, I need to define the ambulance. Each ambulance is represented by one thread. Each thread is created, then it executes Dijkstra Algorithm and saves its value in an array. When all the threads have run and saved their values in the array, then the one that has the shortest path within all will mean that is the ambulance that is closer to the accident. Therefore, this ambulance is going to be assigned to take the accident.

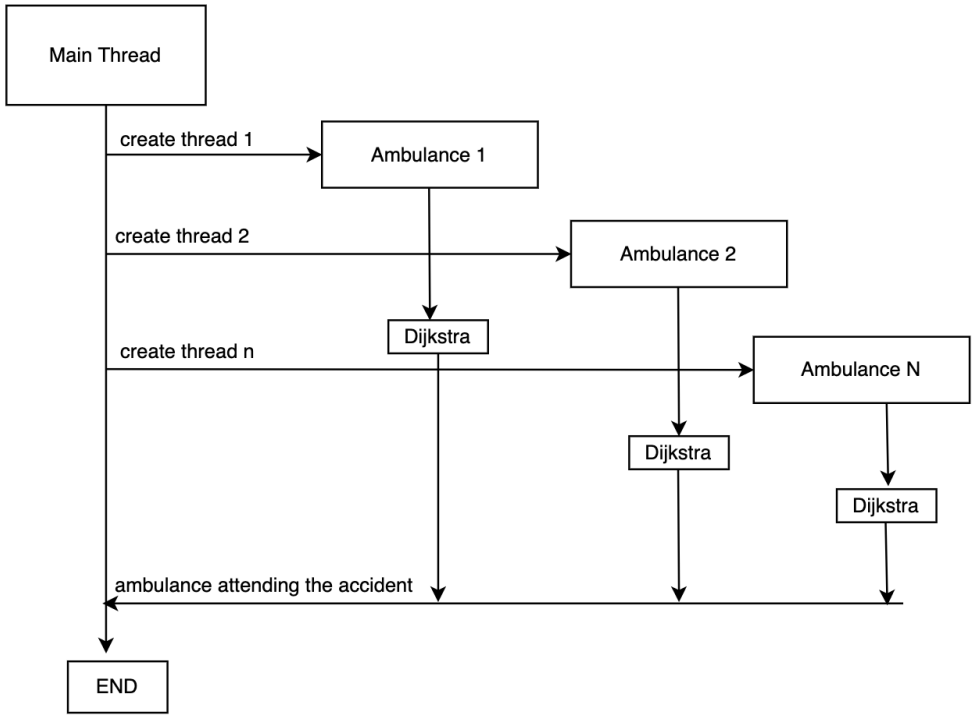


Figure 3: Threads diagram for explaining how each of them runs the dijkstra algorithm.

For the project there is a maximum of 8 ambulances and 5 accidents. When an accident is reported, all the threads are created and each one will calculate its distance and then the shortest one will be assigned to attending the accident (Figure 3), this process is repeated for each accident.

4) Street

The ambulances are on the street which is defined as the graph previously presented (Figure 1). The street is the resource that all the ambulance share because all of them are going to be active and are going to be assigned to an accident, when this happens, the counter of the number of ambulances which are available increases as well as the number of the ambulances attending an accident, but the number of free ambulances will keep decreasing as the ambulances are assigned to an accident. These methods are synchronized for having the same results on all the threads.

5) Objects and Classes

Classes:

- Ambulances: For the creation of each ambulance and the run method for each thread.
- Street: For manipulating the shared variables which are the shared resources.
- Djikstra: The execution of the dijkstra algorithm.

Objects:

- Street: For each thread to run the synchronized methods.
- Dijkstra: For each thread (ambulance) with its location runs the dijkstra algorithm

5. Results

The program works correctly the ambulance that is assigned to the accident is the one that is closest and it actually covers the ideal scenario where the ambulance that arrives is not the one that received first the call but the one that is closest, making sure that the patient is attended, decreasing the traffic congestion of the accident and completing the ideal scenario. Concurrency permits that the number of ambulances in service in the street are modified depending on what happens, if an ambulance is assigned to attend an accident or not.

As an advantage concurrency allows faster response which makes the task that is being processed more efficient, allowing all threads to run the dijkstra algorithm and together get the one that is closer and at the same time updating the number of free ambulances. If two ambulances have

the same distance then the one that is going to attend the accident will be the first one reported.

Note: The shortest path is filled in with the answer of each thread, it will be modified when each thread executes it, that's why in the first cycle that you see it stays with the value of the first thread.

a) Number of ambulances is greater than number of accidents

If the number of ambulances is greater than the number of accidents then only X ambulances will be attending the X accidents and the rest will remain available to attend if another accident appears.

```
Pool of ambulances: 2
Number of accidents: 1
Ambulance 0 will be in service in 11 seconds.
Ambulance 0 is in service
ACCIDENT LOCATION: 7
Source (Ambulance Location): 0 Destination (Accident Location): 7 Distance: 8
------------------------------------------------------------------------ Ambulance = 0 Distance(seconds) = 8 Shortest Path= 8
------------------------------------------------------------------------ Ambulance = 1 Distance(seconds) = 0 Shortest Path= 8
Ambulance 1 will be in service in 8 seconds.
Ambulance 1 is in service
ACCIDENT LOCATION: 7
Source (Ambulance Location): 1 Destination (Accident Location): 7 Distance: 11
------------------------------------------------------------------------ Ambulance = 0 Distance(seconds) = 8 Shortest Path= 8
------------------------------------------------------------------------ Ambulance = 1 Distance(seconds) = 11 Shortest Path= 8
------Ambulance attending the accident in location 7 is ambulanceID: 1 it will take 8 seconds to arrive------
***Free ambulances 1
```

Figure 4: Result for No. of ambulances > No. of accidents

b) Number of accidents is greater than the number of ambulances

If the number of accidents is greater than the number of ambulances then only X accidents will be attended by the X ambulances, the rest of the accidents will stay in queue waiting to be attended by one that is free.

```
Pool of ambulances: 1
Number of accidents: 2
Ambulance 0 will be in service in 8 seconds.
Ambulance 0 is in service
ACCIDENT LOCATION: 4
Source (Ambulance Location): 0 Destination (Accident Location): 4 Distance: 21
------------------------------------------------------------------------ Ambulance = 0 Distance(seconds) = 21 Shortest Path= 21
------Ambulance attending the accident in location 4 is ambulanceID: 0 it will take 21 seconds to arrive------
***Free ambulances 0
------------------------ACCIDENT IN LOCATION: 3 IS WAITING TO BE ASSIGNED AN AMBULANCE--------------
```

Figure 5: Result for No. of ambulances < No. of accidents

c) Number of accidents is equal to the number of ambulances

If the number of ambulances is equal to the number of accidents then all the accidents will be covered but no ambulances will remain free.

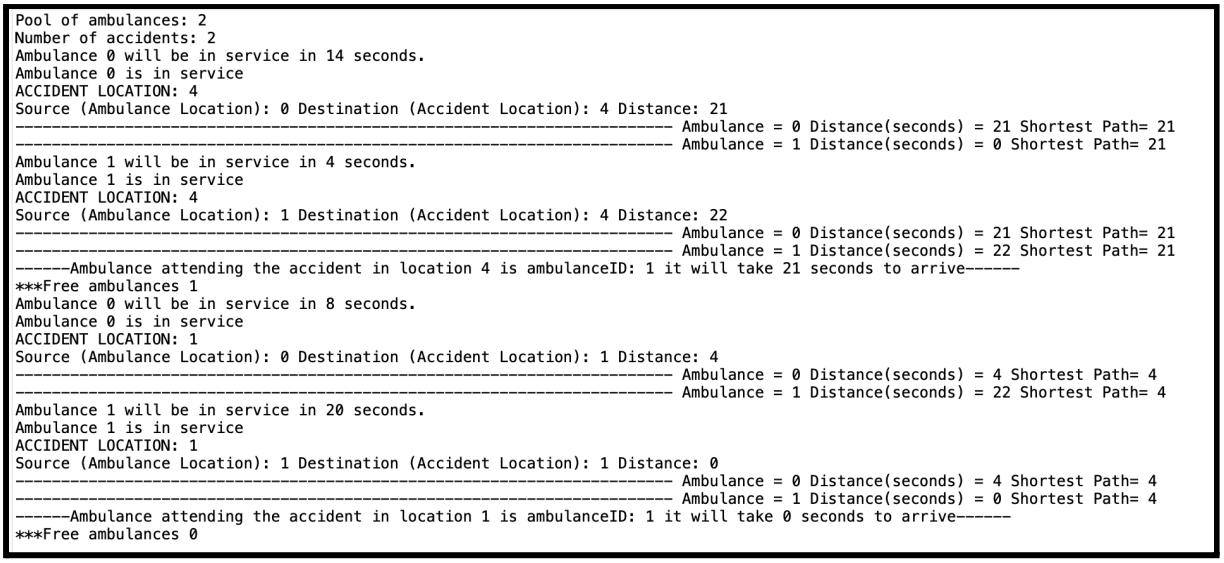


Figure 6: Result for No. of ambulances = No. of accidents

d) Number of accidents=0

If the number of accidents is equal to 0 then there are no accidents to be attended and all ambulances are free.

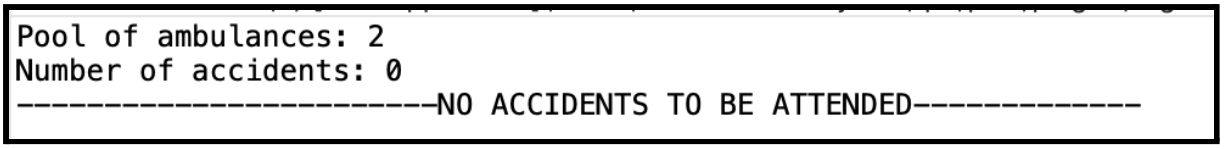


Figure 7: Result for No. of accidents =0

e) Number of ambulances=0

If the number of ambulances is equal to 0 then there are no ambulances attending an accident, and accidents will have to wait till they are in service.

Figure 8: Result for No. of ambulances =0

6. Conclusions

The project helped me understand how to apply all the knowledge that has been connecting from all these semesters into a real-life scenario. Every time I pass by a car accident and see the scenario described in the context of the problem, I asked myself “what can I do, it will be ideal that...” and now that I could at least program an approximation to a solution, to the reality makes me happy and encourages me to continue looking for solutions for implementing a complete solution for contributing to society.

With the help of threads which simulated the ambulances, the dijkstra algorithm for the distances, and the synchronized methods, via concurrency and object oriented programming I could present the first solution to a social project that with more time and effort could end in a very important solution that may also be applied in other areas not only for the health industry but for public transportation for example.

7. Further Implementations:

For further implementations it is necessary to add the functionality for the path that the ambulance should take to the closest hospital. In this way the project can finish the cycle of the activities that an ambulance has to complete once it has been assigned to an accident. Another implementation that may be added to the project is in terms of the measure of the cost of the path, for example, having the cost in km, minutes and evaluate which one is actually the shortest because sometimes the fact that is closer in km doesn't necessarily mean that its closest than another one that's further away, you will have to evaluate time and traffic to answer which one. More and more extra facts appear for making a more precise solution.

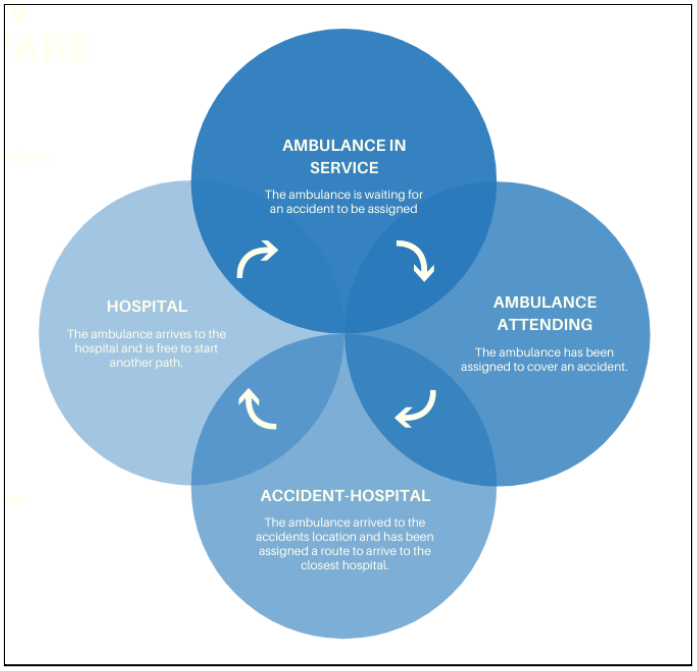
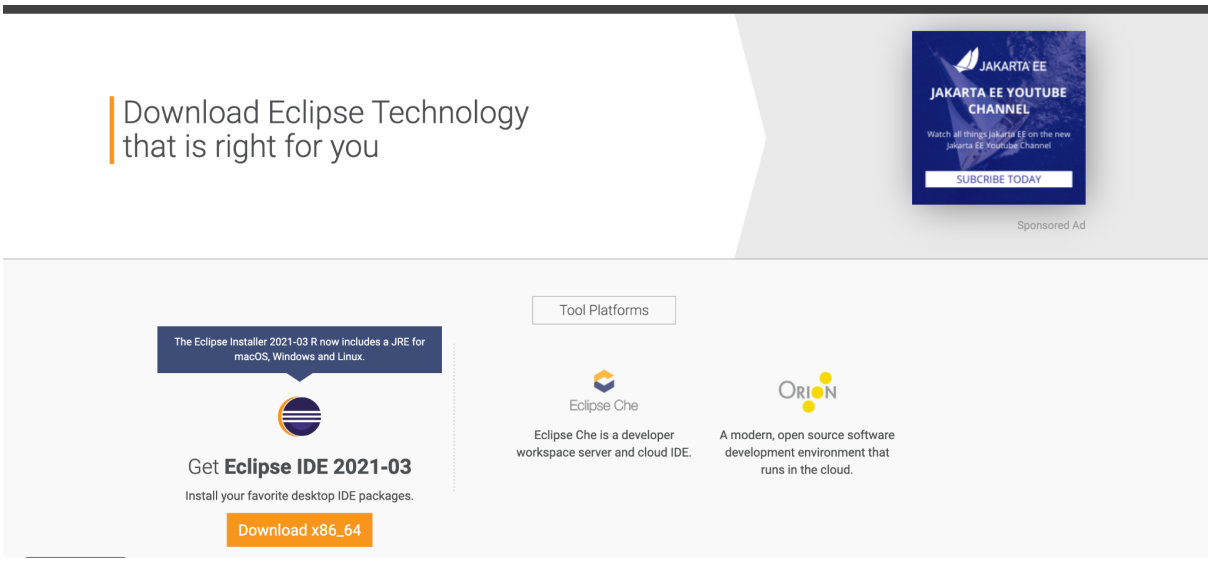


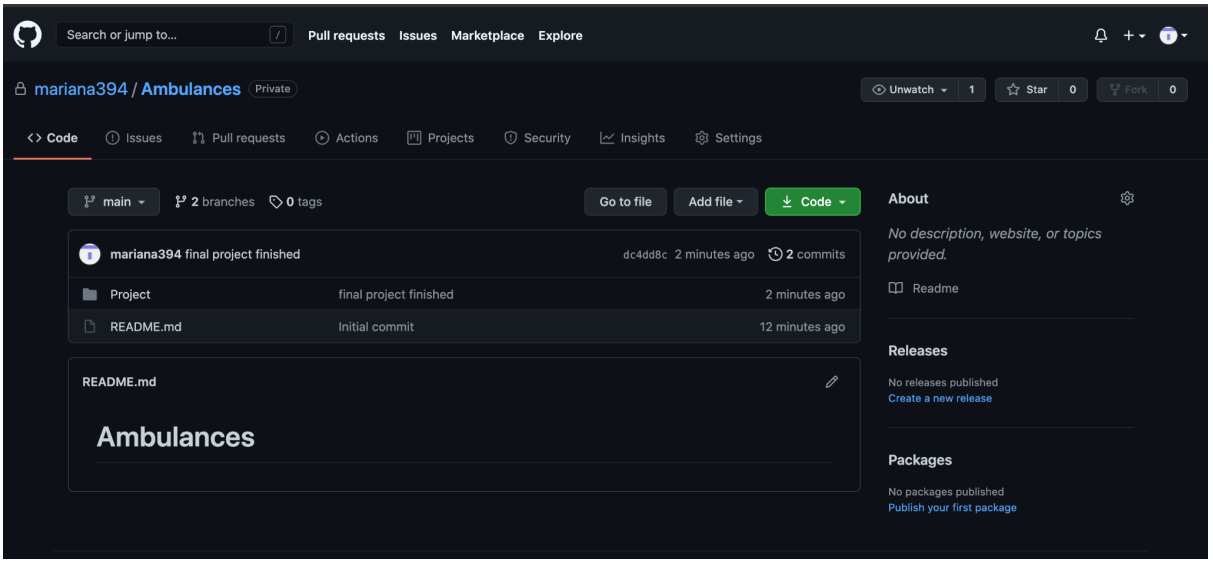
Figure 9: Cycle of the complete project implementation.

8. Set up information

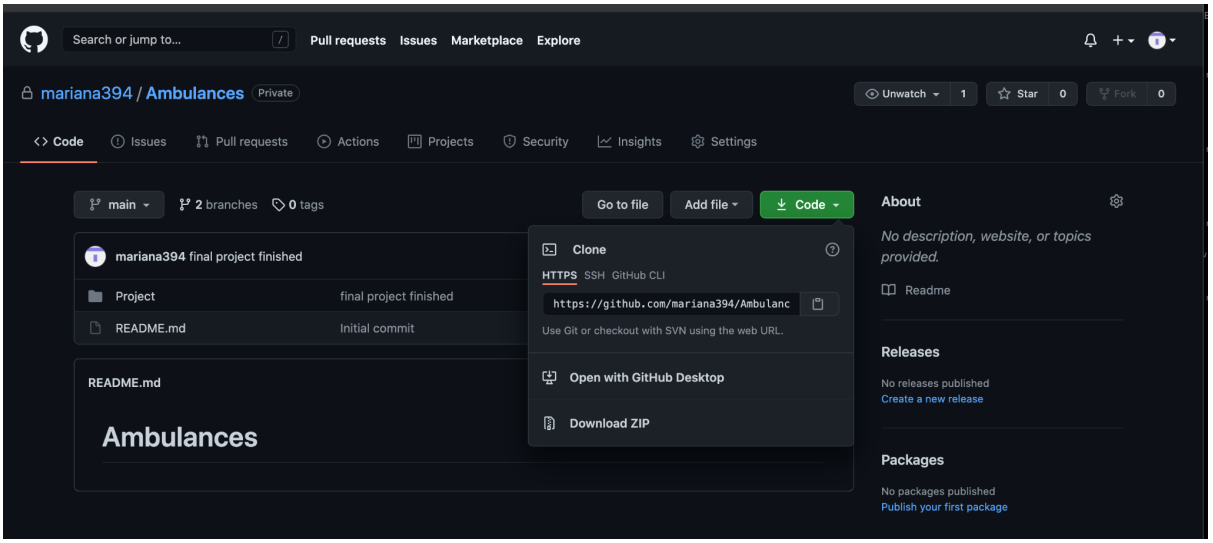
1) Download Eclipse: https://www.eclipse.org/downloads/



2) Go to the projects repository: https://github.com/mariana394/Ambulances/tree/main

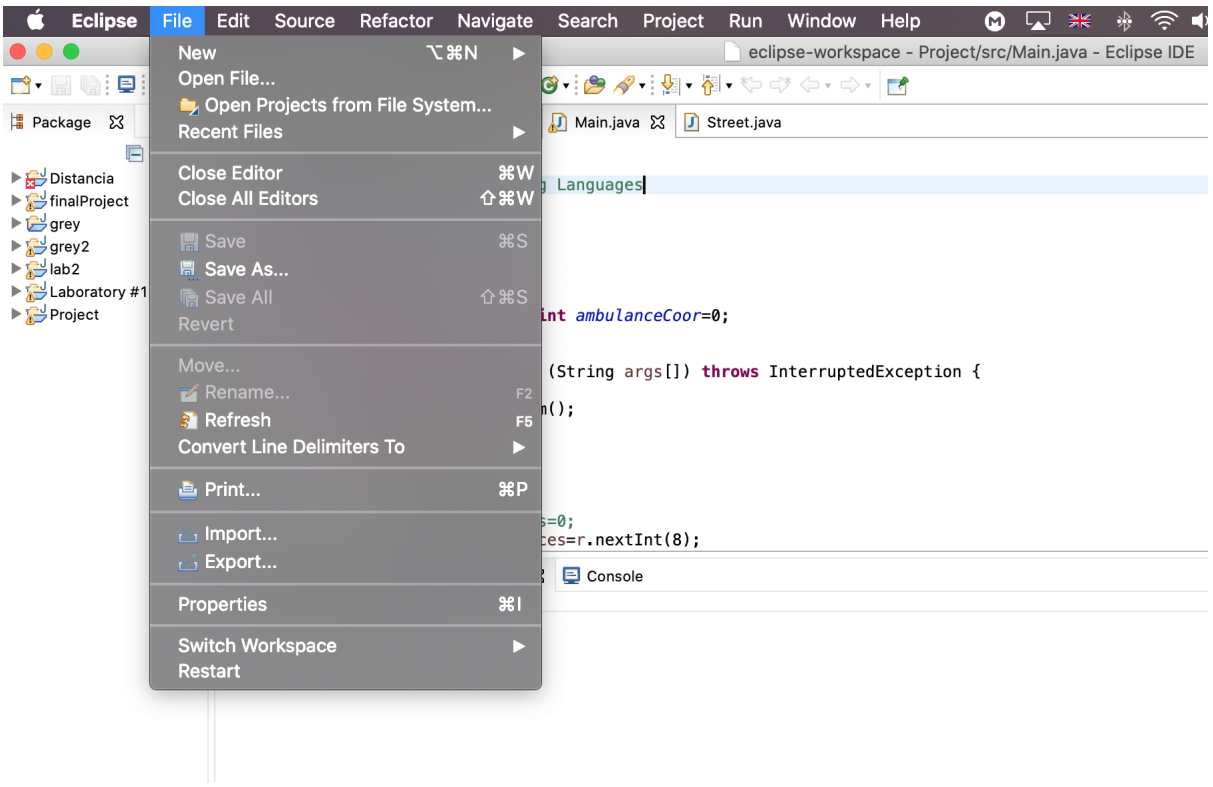


3) Click on code and then click on download ZIP or clone the repository in your computer, whichever you prefer.

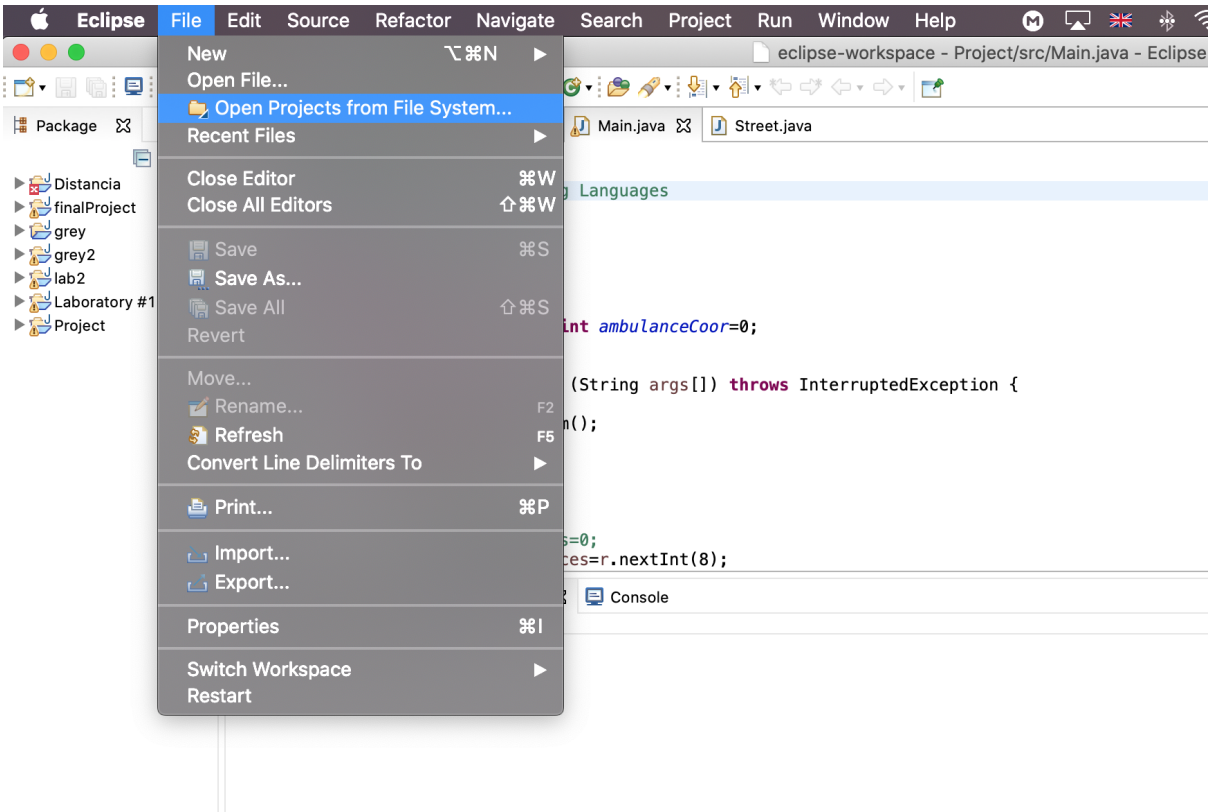


4) If you downloaded a zip file, unzip it, if not continue to step 5.

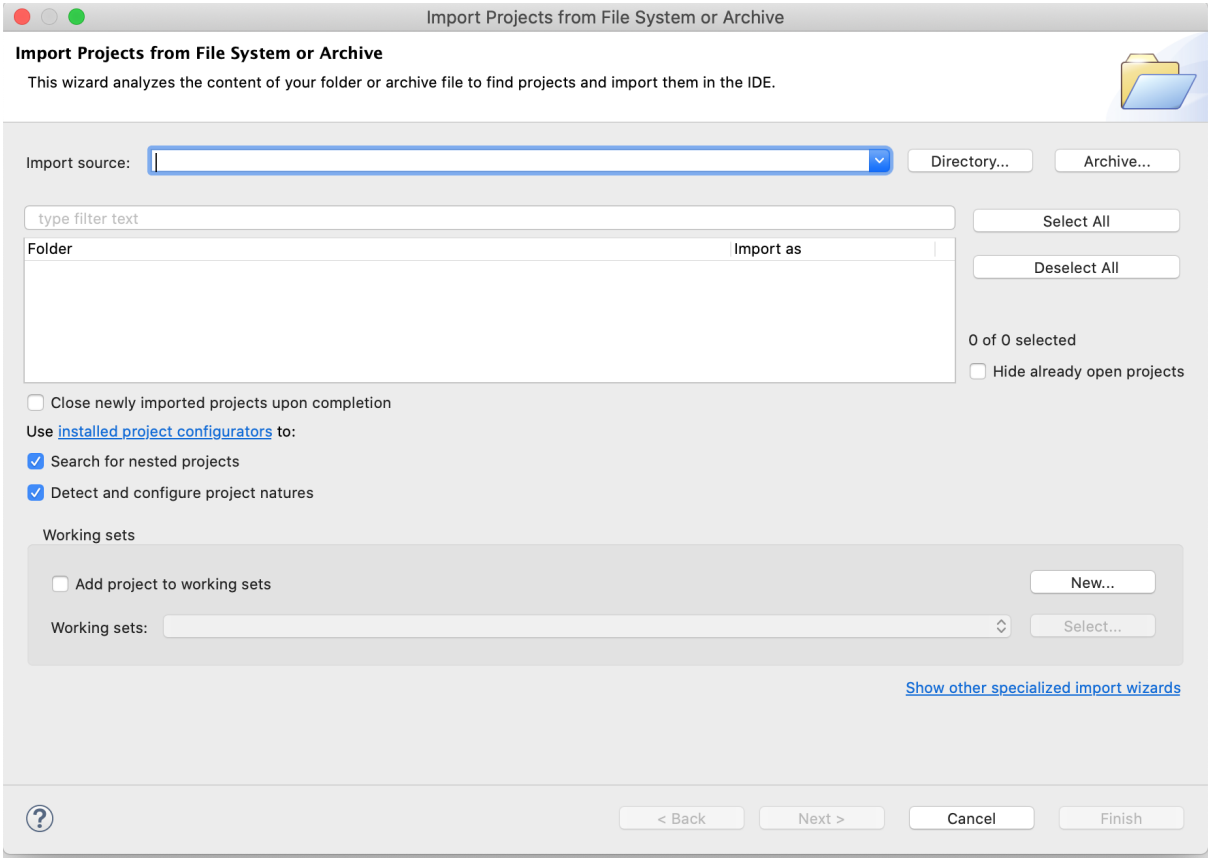
5) Open eclipse and select-> file



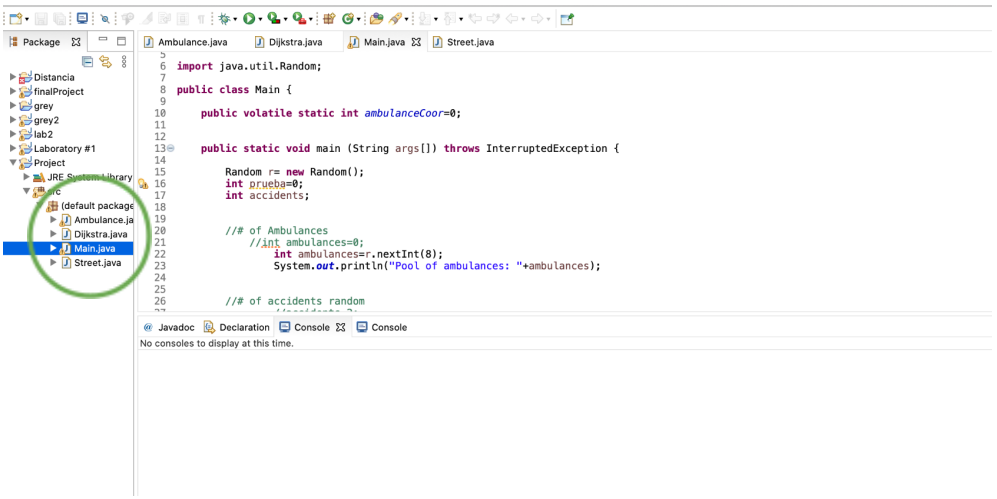
6) Select Open project from file system



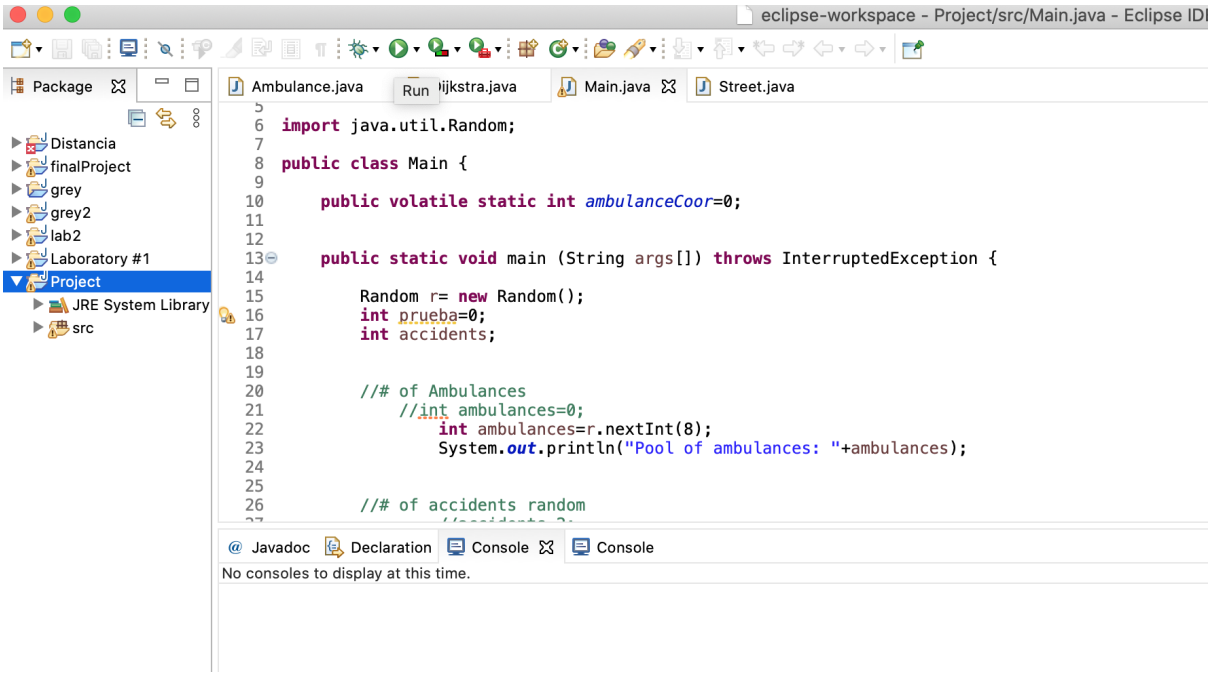
7) Select the path where you saved the folder



8) Open the Main class by expanding the folder in the left side:



9) Click on run



10) Observe the output

9. References:

Alcaldía de Santiago de Cali. (2020). Retrieved May 28, 2021, from Cali.gov.co website: https://www.cali.gov.co/documentos/1306/estadisticas-accidentalidad/

A. S. (2020, May 19). ¿Carrera por el SOAT? Fuerte accidente de una ambulancia en Cali - TuBarco Noticias. Retrieved May 28, 2021, from TuBarco Noticias website: https://tubarco.news/tubarco-noticias-occidente/tubarco-noticias-cali/carrera-por-el-soat-fuerte-accidente-de-una-ambulancia-en-cali/

Qué significa SOAT - Compra tu SOAT Online con descuento. (2021, March 15). Retrieved May 28, 2021, from Compra tu SOAT Online con descuento website: https://soatmundial.com.co/que-significa-soat/

Jakob Jenkov. (2020). Java Volatile Keyword. Retrieved May 28, 2021, from Jenkov.com website: http://tutorials.jenkov.com/java-concurrency/volatile.html#:~:text=The%20Java%20volatile%20keyword%20is%20intended%20to%20address%20variable%20visibility,read%20directly%20from%20main%20memory

baeldung. (2017, August 20). Guide to the Volatile Keyword in Java | Baeldung. Retrieved May 28, 2021, from Baeldung website: https://www.baeldung.com/java-volatile

Dijsktra's algorithm. (2012, November 25). Retrieved May 28, 2021, from GeeksforGeeks website: https://www.geeksforgeeks.org/dijkstras-shortest-path-algorithm-greedy-algo-7/

orellabac. (2015, April 13). orellabac/algoritmosS12015. Retrieved May 28, 2021, from GitHub website: https://github.com/orellabac/algoritmosS12015/blob/master/greedy/Dijkstra.java

Lesson: Concurrency (The Java™ Tutorials > Essential Classes). (2021). Retrieved May 28, 2021, from Oracle.com website: https://docs.oracle.com/javase/tutorial/essential/concurrency/index.html

Doherty, E. (2020). What is Object Oriented Programming? OOP Explained in Depth. Retrieved May 28, 2021, from Educative: Interactive Courses for Software Developers website: https://www.educative.io/blog/object-oriented-programming

Java Concurrency Tutorial - HowToDoInJava. (2020, October 22). Retrieved May 28, 2021, from HowToDoInJava website: https://howtodoinjava.com/java-concurrency-tutorial/

Estrada, J. and Saucedo Guapi, M., 2012. *SIMULADOR DE TRÁFICO VEHICULAR "MOTOR DE FLUJO"*. [online] Bibliotecadigital.usbcali.edu.co. Available at: http://bibliotecadigital.usbcali.edu.co/bitstream/10819/1447/1/Simulador_Tr%C3%A1fico_Vehicular_Estrada_2012.pdf

Interviews:

Seba, H., 2021. *Proyecto Ambulancias vision medica*.

Kevin, ., 2021. *Proyecto Ambulancias vision paramedico*