Rapport Java

Andreas Brunström

Introduction:

The project task for the course was to create a distributed framework, that simulates agents or pedestrians walking down a room or a corridor from one side to the other. The goal is to make the pedestrians walk wiyout colliding with other pedestrians on the way. The program language to use in this project is ofcourse Java.

Implementation:

The first thing to decide was how the distributed system were going to be implemented. The core solution is a Server-client model, but there were a lot of possible solutions. The server should focus on handling the network communication, and also spawn and despawn pedestrians on each side of the corridor, while the clients should do the movement calculation of each pedestrian and check for collisions.

I decided to go for a model with two clients, one for each side, so all pedestrians that spawned from the right side of the corridor where handeled by one of the clients, and all pedestrians from the other side by the other client.

The network handling is fairly simple, it only handles text *Strings* and sends the pedestrian data from the server to the client and from the clients to the server. The pedestran consists of three values, its x position, its y position and its owner client. The textstring for each pedestrian looks like this:

x-val/y-val/id:

And all pedestrians are sent in a single long text string and are chopped up to its components and puzzled togeather by the network handling thread at each end of the connection.

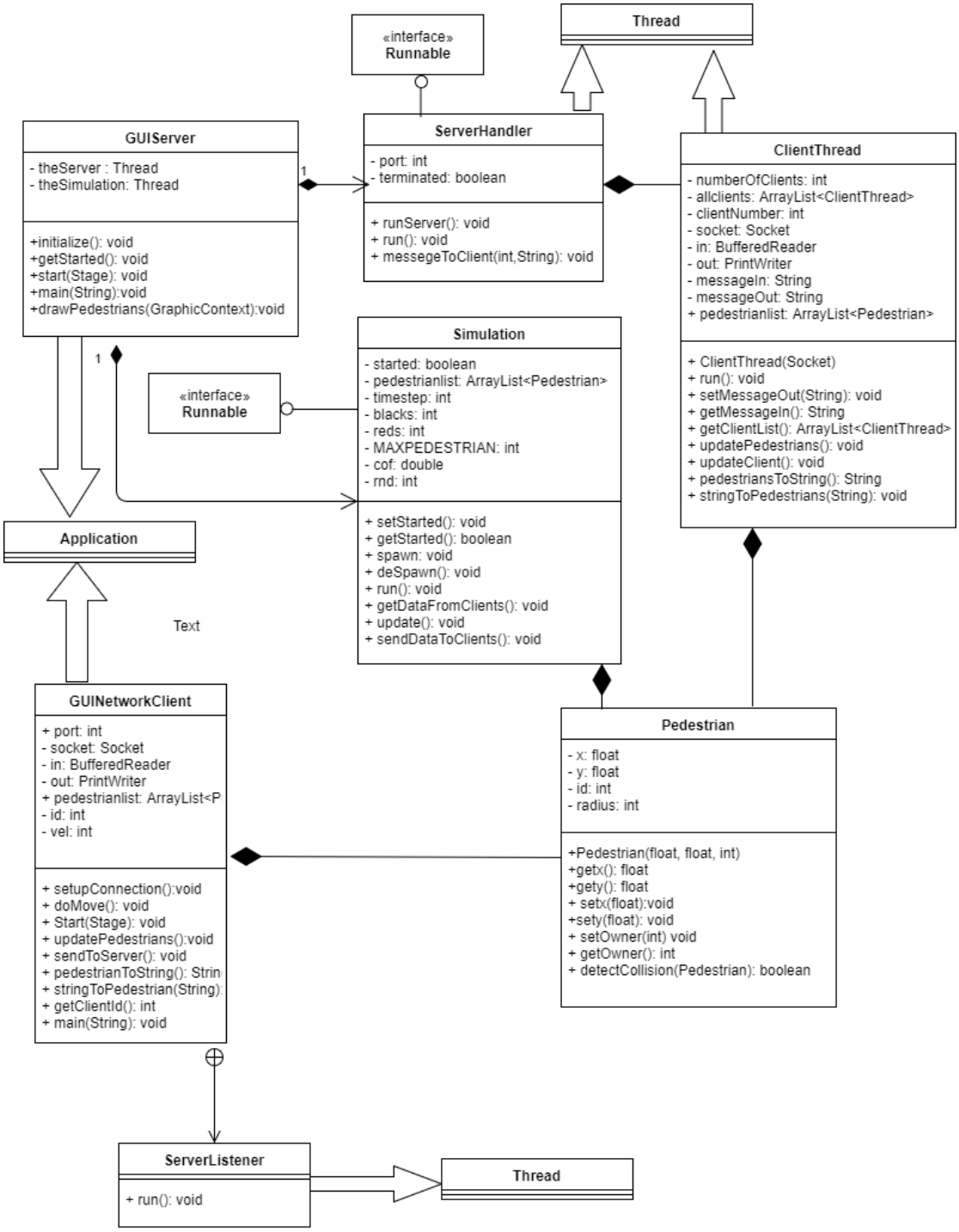
On the server side there are three primary threads running the program, it is the network thread, the GUI thread and the simulation thread.

The network thread handles all the setup and communication between the server and the client.

The GUI thread handles the GUI, it is using JavaFX and it creates a simple window containing two buttons, one for starting the simulation ones the two clients are connected, and one for closing the program, it also contains a canvas where it animates the simulation using Timeframe animation. The GUI thread extends Application and also contains the programs *main()* method.

Finally, the simulation thread handles all the simulation parts. It is running a loop containing three diffrent methods. It gathers data from all the clients and merge it into one single arraylist of all the pedestrians in the simulation using *getDataFromClients(),* it then spawns and despawns according to certain conditions using *update(),* and last it sends the new data to each of the clients using *sendDataToClients().*

The client side consists of two threads, the first is the server listener, that handles the network and calculates the new position of the pedestrians as soon as it gets anything from the server and sends it back. And a GUI thread that starts the program and has a small GUI with just two buttons, one for connecting to the server and one for closing the program. When the simulation runs the client just waits in the serverlistener thread for any message, and as soon as it gets it, it calculates and sends the result back to the server.



Thoughts on the project:

One of my biggest misstake I did was to underestimate the time it would take for me to get a working program up and running. I made a major design flaw at the very beginning, when I created a "Server" class that would be running all the three threads and contain the main() method instead of just let the JavaFX thread that extends application be the startingpoint of the program. It made it nearly impossible to get the GUI to interact with the other threads in a good

way, and in the end I decided to scrap the first version and just go for the JavaFX thread to be the startingpoint.

I also spent a lot of time on the network, first I decided to send information as Strings, it is something I am familiar with and I thought it to be the easiest way. But the more I learned I realized that I wanted to serialize it instead, because I just think it looks nicer. So when I had to redo som of the project because of the GUI problem described above, I also decided to go for serialize instead.

But I ran into alot of problem, I didnt realize that I did actually send String to the client, I send the id for example. Because of that, I needed to imlplement some kind of state machine or somethin to keep track of when I was to send a String and when I was to send an object using serialize. Since I already was late and my first tries didnt get it to work, I decided to go back to sending only Strings, since I knew that it worked.

The algoritm for moving the pedestrians isnt as "fancy" as I maybe wanted it to be. It gets the job done in getting the pedestrians from one side to the other and "trying" to avoid collisions, but really nothing more than that. But it was lowest on prioritylist regarding all the problems I ran into during the development.

I have done some debugging. I used the console to write out what is happening and also using exceptions. I have done some runnings of the program and think it works fairly good, but not perfect, the pedestrians sometimes tend to "scratch" one another when they are on a collision course, and I have fiddled with the diffrent variables but havnt got anything to work perfectly.