Content

About the Project

The projects task is to develop an application to render Mandelbrot se. providing a 'Client' and 'Server' which together renders parts of the Mandelbrot set. Server accepts requests on a TCP port. The client spreads the workload over a set of servers.

The project is published on GitHub
https://github.com/alaulwan/Fractals

To clone the repository
https://github.com/alaulwan/Fractals.git

Interface

The server accepts a request that has this form:

GET /mandelbrot/{min_c_re}/{min_c_im}/{max_c_re}/{max_c_im}/{x}/{y}/{inf_n}

It should return a gray-scale image of dimension x times y, where each pixel corresponds to the number of iterations it takes, until $|z_n| > s$, $z_n + 1 = z_n 2 + c$, (c is complex, $s \in R^+$).

*In the current version of the server, **s = 2** by default.*

The client is a command-line tool with these arguments:

min_c_re min_c_im max_c_re max_c_im max_n x y divisions list-of-servers

Tools Used

For development, the following tools is used:

- Eclipse (Java IDE)
- Java SE Development Kit 8 (JDK-8)

Manual (Demo video: https://youtu.be/2D1pf9yfhQI)

Download client.jar and server.jar from Here.

To run the server and the client, in a console, in the same directory as .jar files is stored:

For the server:

java -jar server.jar {*port*}

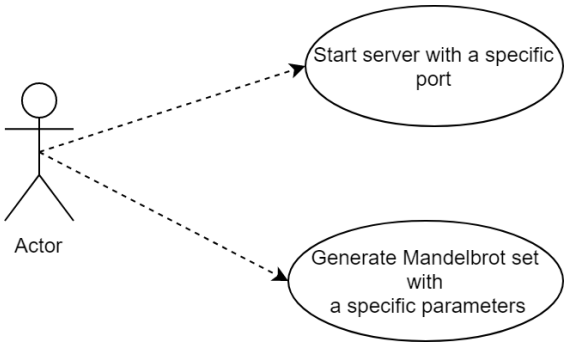
For the client:

java -jar client.jar {*min_c_re*} {*min_c_im*} {*max_c_re*} {*max_c_im*} {*max_n*} {*x*} {*y*} {*divisions*} {*list-of-servers*}

The result will be saved in the same folder of the client.jar with tha name Mandelbrot.pgm.

OBS: It requires a **JRE-8** (Java Runtime Environment) or later to be installed on your host machine. You find the latest version to download at
https://www.java.com/inc/BrowserRedirect1.jsp?locale=en

Use Case Diagram



Use Case Descriptions

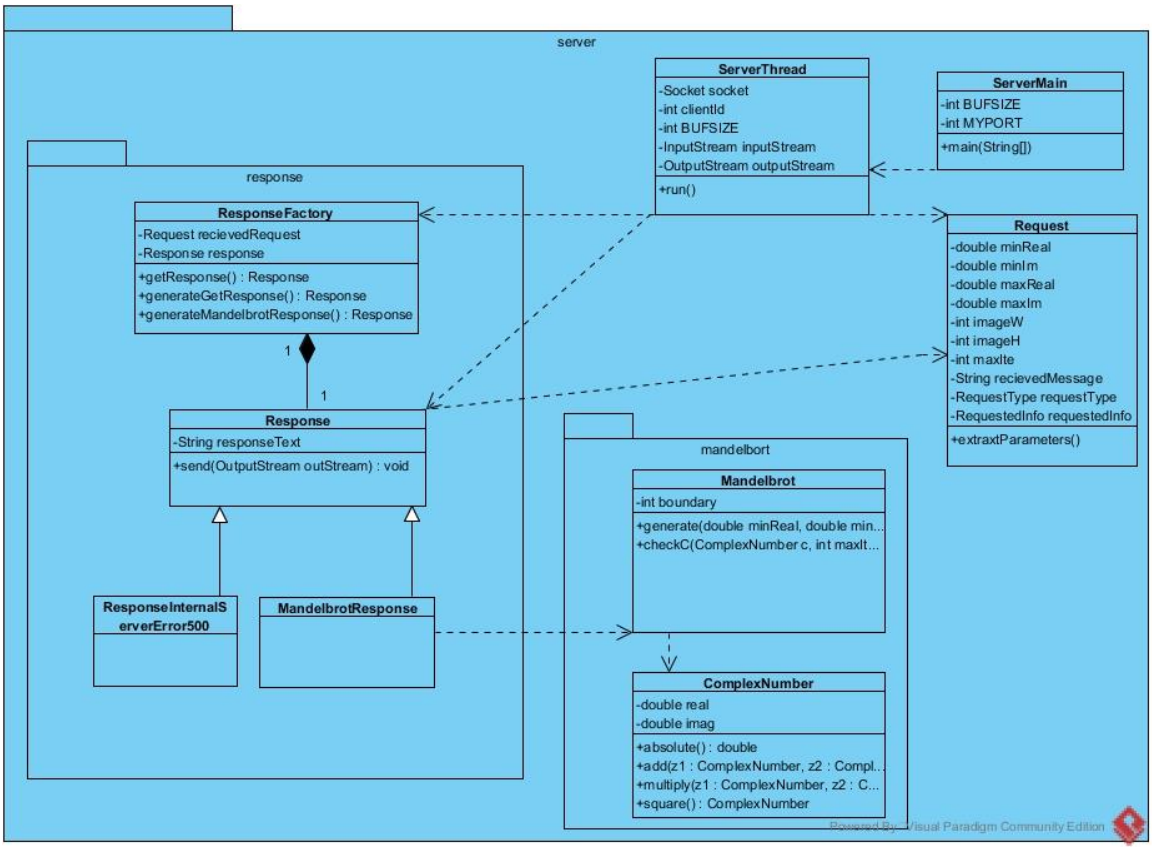
Use case #1
Actors: Any one
Description: Run the server
Pre-Conditions: • JDK-8 is installed. • server.jar is downloaded.
Main flow: 1. In the console, navigate to the same folder as server.jar 2. Type the command java -jar server.jar {*port*} 3. The server is running and listen to the defined port.
Post-Condition: The server is running on the defined port.
Alternative flow: 2.a The user does not define the port. 3.a The server listens to the default port 8888.

Use case #2
Actors: Any one
Description: Generate Mandelbrot set
Pre-Conditions: • JDK-8 is installed. • client.jar is downloaded. • One or more server is running
Main flow: 1. In the console, navigate to the same folder as server.jar 2. Type the command java -jar client.jar {*min_c_re*} {*min_c_im*} {*max_c_re*} {*max_c_im*} {*max_n*} {*x*} {*y*} {*divisions*} {*list-of-servers*} 3. The client sends a request to servers-list. 4. Each server send back a response. 5. The client merge the received responses and save the result in .pgm file in the same directory of client.jar.
Post-Condition: Mandelbrot set is generated and saved in .pgm file.
Alternative flow: 2.a The user does not provide all parameters. 3.a The client view an error message.

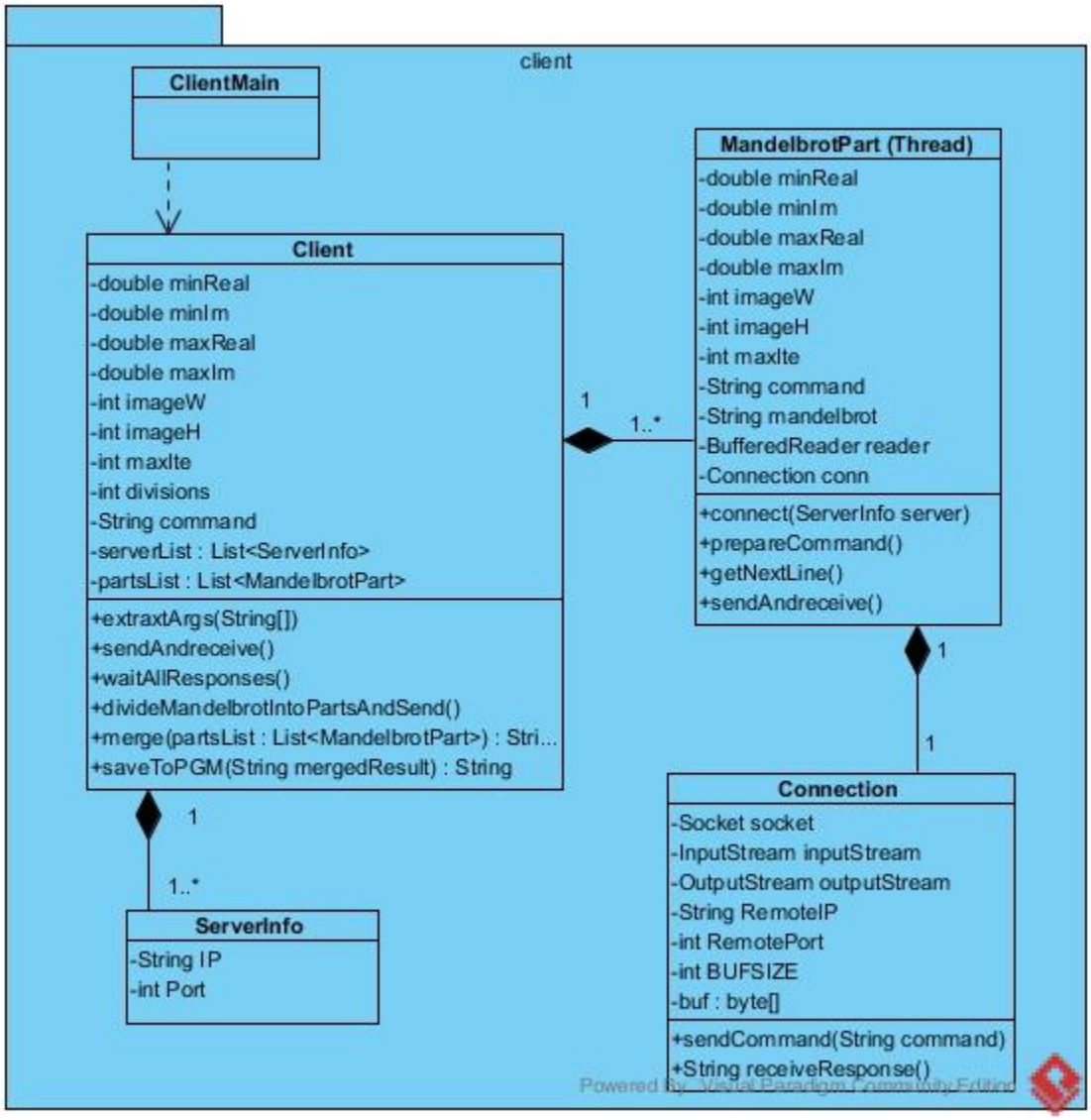
Class Diagram

For the server:

The benefits of having the class “ResponseFactory” and the abstract class “Response” is to make it easier to add more functionality to the server in the future.

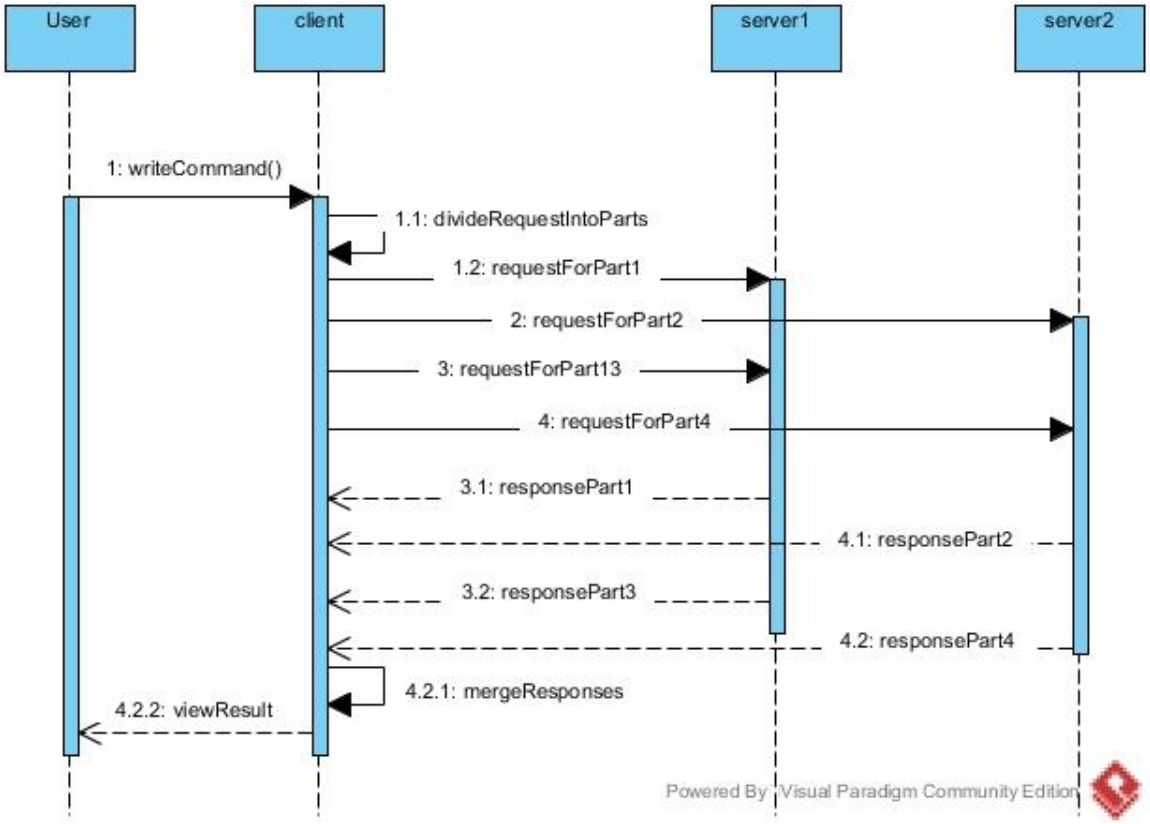


For the client:



Sequence Diagrams

This sequence diagram for division = 2, and list of two servers.



The following sequence diagram illustrate how a server manipulate the request: