Group E9: Diego A. Polanco L. - Andrés R. Chamorro M. - Oscar S. Muñoz R.

Problem identification and requirements analysis

Case Study: Pipe Mania

Client	**APO-2 teachers**
User	Course monitor and players
Functional requirements	RF0: Start game RF1: Show game RF2: Lay a pipe RF3: Evaluate (Simulate) the pipe network RF4: Calculate the score RF5: View the score
Context of the problem	Pipe Mania, which consists of a pipe simulator, is an 8x8 board that is displayed by console, in which the following are placed: vertical pipes (| |), horizontal pipes (=) and circular pipe (o). The idea of the game is to make a pipe that reaches the Drain space (D) from the Source space (F). It should be able to save player scores, detect if the pipe network is valid, exchange pipes, see the top 5 high scores, show the updated pipe and store the players name and their high score.
Non-functional requirements	RF0:The code must be written in Java, linked lists must be used,trees and recursive methods. RF1: Test management: A test plan must be developed that includes validation of the prototype's functionalities and its scalability. The test plan

	must define the test cases, acceptance criteria and the tools to be used for the execution and monitoring of the test. RF2: Performance: The system must be able to handle a large volume of simultaneous transactions and operations without affecting system performance or response speed. RF3: Usability: The menu that allows the user to view their score within the game must be intuitive and easy to use. In addition, the graphical representation of the dashboard must be clear and attractive to the user.

Requirements table:

<table>
<tr><td>Identifier and name</td><td colspan="3">[RF0-Start game]</td></tr>
<tr><td>Summary</td><td colspan="3">The system must allow the user to create a new game, where it will allow them to enter the user's nickname. Later it will show the user an 8x8 board placing the source (F) and the drain (D) at random within the same board to begin the connection of the source and the drain in the game. Additionally, the system must save the start and end time of the game in seconds.</td></tr>
<tr><td rowspan="2">Appetizer</td><td>Entry name</td><td>Datatype</td><td>Condition valid values</td></tr>
<tr><td>nickName</td><td>String</td><td>-The space should not be empty

- Only inputs of type string or string of characters

-That the nickname is not repeated within the list of players</td></tr>
<tr><td>Result or Postcondition</td><td colspan="3">The user's nickname will be saved in the list of registered players within the system.</td></tr>
<tr><td>Departures</td><td>Output name</td><td>Datatype</td><td>Format</td></tr>
</table>

	-	-	-

Identifier and name	*[RF1-Show Game]*		
Summary	*The board has dimensions 8x8, in each square there will be:* *A**'X'** representing the absence of a pipe, A**'='** representing a horizontal pipe, A**'||'** representing a vertical pipe, a**'O'** representing a circular pipe, a**'F'** of water source and a**'D'** drain*		
Appetizer	**Entry name**	**Datatype**	**Condition valid values**
	-	-	-
Result or Postcondition	Creates a String data that represents the pipeline board of doubly linked lists		
Departures	**Output name**	**Datatype**	**Format**
	board	String	*8*8 matrix showing randomly located source and drain.*

Identifier and name	*[RF2-Laying a pipe]*
Summary	*The system will allow the user to put a pipe into the board to connect the source to the drain. The player will enter the position of the row (x) and column (y) where he*

	*wants to place the pipe inside the board, either (=) for horizontal, (o) for circular in case of water flows at an angle of 90° and**(||)** for the vertical pipes. Later the pipe will be registered within the dashboard.* *It should not work if you want to change the font**(F)** or the drain**(D)** within the coordinates entered by the user.* *The coordinates entered by the user cannot be less than 0 or greater than 7.*		
Appetizer	**Entry name**	**Datatype**	**Condition valid values**
	row	int	*-The space should not be empty* *- Only inputs of integer type, no values of other types.*
	column	int	*-The space should not be empty* *- Only inputs of type integer, no values of other types*
	pipe	int	*-The space should not be empty* *- Only inputs of type int, no values of other types.* *-I only knowwill allow choose between three optionsuser pipes:* ***1. '='*** ***2. '||'*** ***3. 'O'*** ***4. "X"***

Result or Postcondition	The system will update the pipe registered by the user within the dashboard with the pipe within the indicated coordinate.		
Departures	**Output name**	**Datatype**	**Format**
	-	-	-

Identifier and name	*[RF3-Evaluate/Simulate pipe network]*		
Summary	*While the user is viewing their game matrix, they are given the opportunity to simulate the route of the pipes from the source to the drain (the only way to win).* *Horizontal pipes can only receive and give water to the sides and vertical pipes from above and below, the circular pipe cannot be connected to the source and drain, the circular pipe can only follow the water cycle at 90° angles, Two elbows cannot be connected in a row, nor can there be two pipes joined to the source or drain.* *The flow of water can go from top to bottom, make 90° turns and go from side to side.*		
Appetizer	**Entry name**	**Datatype**	**Condition valid values**
	-	-	-
Result or Postcondition	Simulates the flow of the pipe and creates a response based on whether it is valid or not. Next step stores the player's result.		
Departures	**Output name**	**Datatype**	**Format**

	valid	String	*"The solution is correct"*
	invalid	String	*"The pipe network is not working"*

Identifier and name	*[RF4-Calculate the score]*		
Summary	*When the user finishes the game, the system must calculate the score and record it in the score table. The calculation will be carried out with the following formula:* $Points = (100 - used\,pipes) * 10 - timeInSeconds$ Whereused pipes is the total number of pieces that the player used for his solution andtimeInSeconds is the time elapsed between the start of the game and the end of the game. The time in seconds will be calculated by subtracting the start and end time of the game.		
Appetizer	**Entry name**	**Datatype**	**Condition valid values**
	-	-	-
Result or Postcondition	The user's score will be saved in the recorded score table within a binary search tree.		
Departures	**Output name**	**Datatype**	**Format**
	-	-	-

<table>
<tr><td>Identifier and name</td><td colspan="3">[RF5-See other players' scores]</td></tr>
<tr><td>Summary</td><td colspan="3">Firstly, the system validates if there are registered players, if there are registered playersIt shows a list of the 5 highest scores next to the nickname of the player who previously entered and gives the option to show the final pipe of each player's game, otherwise it sends a message.</td></tr>
<tr><td rowspan="2">Appetizer</td><td>Entry name</td><td>Datatype</td><td>Condition valid values</td></tr>
<tr><td>-</td><td>-</td><td>-</td></tr>
<tr><td>Result or Postcondition</td><td colspan="3">Shows the player name, score and game of the 5 players with the highest score.</td></tr>
<tr><td rowspan="3">Departures</td><td>Output name</td><td>Datatype</td><td>Format</td></tr>
<tr><td>ranking</td><td>String</td><td>Player position + player name + score.

All this from each player</td></tr>
<tr><td>Board</td><td>String</td><td>Simulates the final pipeline of the selected player</td></tr>
<tr><td></td><td>Out</td><td>String</td><td>A message will be displayed if there are no registered players:

“There are no registered players”</td></tr>
</table>

Group E9: Diego A. Polanco L. - Andrés R. Chamorro M. - Oscar S. Muñoz R.