CSCI-605 Advanced Object-Oriented Programming Concepts

Homework 10 - Concentration

1 Introduction

Concentration is a match making memory game. Pairs of cards are randomly placed face down in a grid-like fashion. The user turns the first card over and tries to find the matching card from the remainder of face down cards by turning over another card. If the cards match they stay face up, otherwise they are turned back over. The goal is for the player to find all the matching pairs - usually in the least number of moves.

In this homework you will be implementing a client-server networked version of this game:

- The game is single player only.
- The game is played across the network where the client represents the player and the server represents the game being played.
- The values of cards are capital letters and always guaranteed to be in pairs.
- The game board is a square grid and all cards are initially face down.

In order for the game to work, you will need to implement a protocol: a simple language with which the client and server can communicate. You will be using TCP/IP sockets to send information back and forth between the server and the client.

1.1 Goals

The goal of this homework is for students to gain experience with the following:

- TCP/IP Sockets
- Multithreading

1.2 Provided Files

- sample_run.txt, a sample run between a client and server on a 2x2 board. This is the client run, the symbol '>' is used to denote user input.

- sample_run_server.txt, the server side output. Note that all output on the server side is up to you and is used for debugging purposes only.
- `common.ConcentrationException`, the custom exception class when something goes wrong like a bad coordinate was indicated.
- `common.ConcentrationProtocol`, the protocol messages used for communication.
- `game.ConcentrationCard`, the representation of a single card in the game.
- `game.ConcentrationBoard`, the game "model" which represents the board. The server should have a copy of this in order to play the game and detect problems. This class is partially implemented, you are required to implement the stub methods.

2 Program Operation

A Concentration game should work as follows:

2.1 Client

- The client has minimal information about the game being played. It only knows the square dimension of the game board and that initially all the cards are face down.
- The only interaction the client has with the human player is to prompt for the coordinate of a card and then request to the server that the card be revealed. The client will not remember any of the coordinates the user has entered.
- When a card is revealed the server will indicate to the client what the face up value of the coordinate of the requested card is so that it can then be displayed to the user.
- When the client reveals two cards, it will be told whether the pair matched or not. In the case of a match, the cards can stay face up, otherwise the cards should revert back to being face down (and the client will not remember this information).
- Once all the matches have been made the client will be told from the server that the game is over.

2.2 Server

- The server can accept connections from multiple clients and simultaneously serve unique games to each client with game boards that are not shared between clients.
- The server is the brains of the game and the referee. It knows the square dimension of the board, and full information about each of the cards in the board, e.g. whether they are face down and hidden, or face up and what their true value is.
- Once the client has connected the server will first let the player know the square dimension of the board.
- A single game with a client sees the server waiting for the player to reveal a card. Once the request comes in, the server will turn the card face up and let the player know what the value is.
- Once a pair of cards has been revealed, the server will let the client know whether there was a match or not. In the case of a match the cards are left face up, otherwise they are turned back down.

- Once all the cards are face up and match, the game is over and the client will be notified. This ends the interaction between the server and client.

3 Design

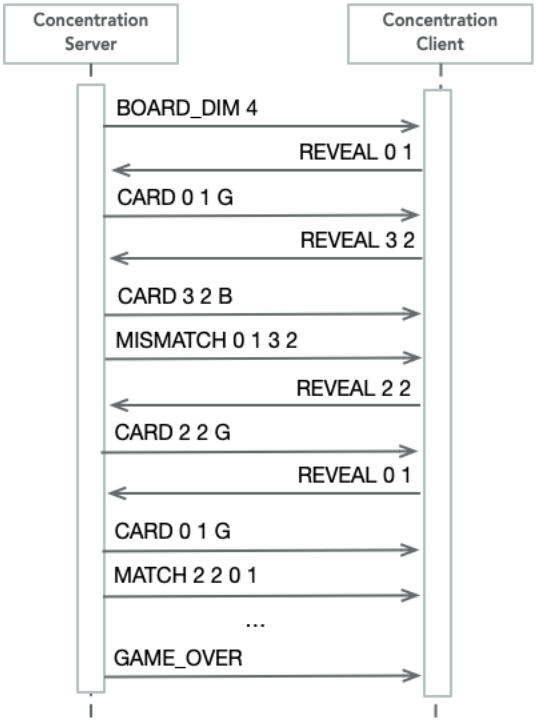
The game board is square in dimension (either 2x2, 4x4 or 6x6) and uses matching pairs of capital letters for the cards. The coordinates are 0-based and the row comes before the column.

3.1 Protocol Description

It is absolutely critical that your client and server follow the protocol exactly! Your client should work with a different server implementation, and vice versa! Once you have them developed, you are encouraged to test with another student, but as always do not share any of your code!!!

3.1.1 Sequence Diagram

Here is a sequence diagram showing a brief example of the messages that go between the server and client.



3.1.2 Server to Client

Message	Format	Example	Description
BOARD_DIM_MSG	"BOARD_DIM %d"	BOARD_DIM 4	The board square dimension is 4x4.
CARD_MSG	"CARD %d %d %s"	CARD 4 1 A	The card at (4,1) is A.
ERROR_MSG	"ERROR %s"	ERROR Invalid coordinate	A card at an invalidate coordinate was requested to be revealed.
GAME_OVER_MSG	"GAME_OVER"	GAME_OVER	The game is over and the player has won.
MATCH_MSG	"MATCH %d %d %d %d"	MATCH 3 1 0 2	The cards at (3,1) and (0,2) that were just revealed are a match.
MISMATCH_MSG	"MISMATCH %d %d %d %d"	MISMATCH 2 5 4 0	The cards at (2,5) and (4,0) that were just revealed do not match.

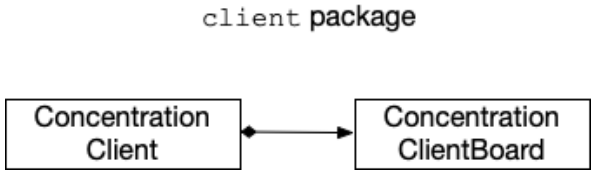
3.1.3 Client to Server

Message	Format	Example	Description
REVEAL_MSG	"REVEAL %d %d"	REVEAL 1 2	Reveal the card at (1,2).

3.2 Client Design

Part of your grade for this homework is based on the design of the classes you write. We expect a fully object oriented design with a reasonable separation of responsibilities between the classes. Here are some suggestions for you.

The client is not a very big program. Like previous homeworks you should try to minimize the amount of work that is done in the main program and pass control to instance methods.



You should write a separate class from the main class that represents the game board. Please note that the board provided to you, `game.ConcentrationBoard`, is way overkill for what you need on the client side. You just need:

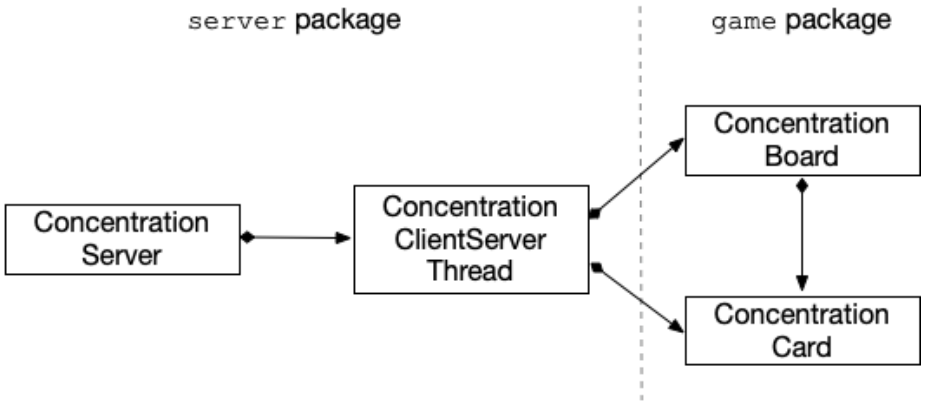
- A matrix that represents what the client shows the user, e.g. a grid of strings or characters.
- A way to build the initial grid once you know the square dimension of the board from the server.

- A way to access and modify the cells in the grid, based on messages from the server.
- A way to print out the board to the user.

Eventually, to make sure the client doesn't send errors to the server, you should have a way to verify the coordinates of a card are in range, and a card that has been revealed is not revealed again.

3.3 Server Design

The server is a bit larger of a program than the client. You should separate the work into different classes, as well as utilize the `game` package classes that represent a fully playable board.



To begin with, you do not need to make your server multithreaded. It is okay to have it work with a single client first, and then worry about handling multiple clients simultaneously at the end.

You are encouraged to add debug messages to your server so you can follow what is going on when a game is played with a client. While the rest of the server does not require thread cooperation, you will need to create a `synchronized` method/region where you do any debug output - so that threads do not step on each other when output occurs.

4 Implementation

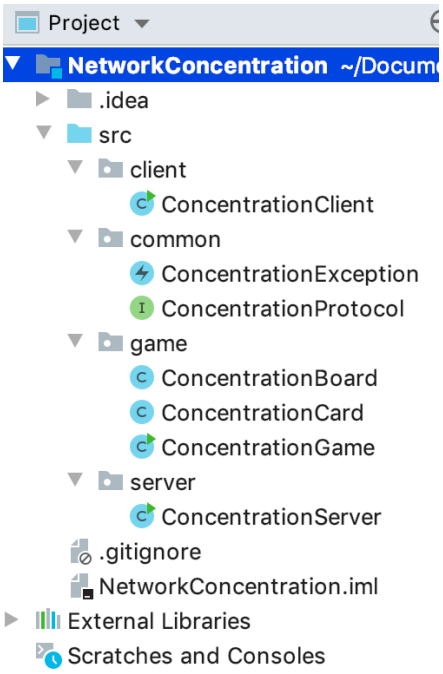
Your Java program should have two classes with executable main method.

- ConcentrationServer - accepts two command line parameters that specify the port number to use on the server and the board dimension.
- ConcentrationClient - accepts two command line parameters that specify the hostname and port number to connect to on the server.

The user should only ever have to input the cards coordinates. No other user input is required.

4.1 Project Structure

Your project should be structure as follows:



4.2 Program Requirements

Unlike previous homeworks we are not providing you with a UML class diagram or Javadocs. To receive full credit you should follow these design suggestions:

- Do not implement everything in the main! You should pass off work to a constructor and methods as soon as you have the necessary information from the command line.
- You are free/encouraged to use more than the main classes for your client and server.
- Your program must follow the protocol exactly. This means your client should be able to work with a reference server. And likewise a reference client should work fine. Your grader will have a reference client that they can test your server with.
- Your program must be robust. The user cannot cause the client to exit prematuraly. When the game is over the client should close down any resources and exit gracefully - there should be no unhandled exception that escapes the `main` method.
- The server must be implemented allowing multiple games to be played concurrently.
- The client should never be able to crash the server program. If it closes its connection early the thread responsible for handling the connection should gracefully exit without leaving any exception unhandled.
- Your server should deal with and return descriptive ERROR messages for:
  - Request to reveal a card at a coordinate that is out of range.
  - Request to reveal a card at a coordinate that is valid but has already been revealed.

5 Submission

You will need to submit all of your code to the MyCourses assignment before the due date. You must submit your `hw10` folder as a ZIP archive named "`hw10.zip`" (if you submit another format, such as 7-Zip, WinRAR, or Tar, you will not receive credit for this homework). Moreover, the zip file must contain only the java files of your solution. Do not submit the .txt files provided in this homework nor compiled files.

6 Grading

The grade breakdown for this homework is as follows:

- Design: 20%
- Functionality 70%
  - Server 40%
  - Client 30%
  - Error Handling: 5%
- Code Style 10%