Part I

General Contents

1. You have to design, develop (write a code), run, test and evaluate the software for 2-players Battleship Game between the users according to the specifications given below. You have to develop an electronic network version of the game whose brief description is attached.
2. You have to try to make your product faster and more reliable. Although addressing security issues is not required in part 1, you might like to start thinking about making it more secure while working on part 1 also.
3. You have to develop and submit proper documentation (see project report section).
4. You have to document
   a. the product itself and describe how to use it,
   b. describe testing you performed and how you evaluated the product based on test results,
   c. the development process (for group projects only): how you distributed the workload, who did what, how much time you spent on different aspects, etc.

User requirements:

Please, note that requirements given below are formulated as user requirements, i.e. given by a non-expert in software engineering or computer science. These requirements could be a bit fuzzy and vogue in some parts. You have to develop an exact specification for your project based on these requirements.

1. The software should be multi-game, meaning that multiple games may try to get an access to the game-server at the same time.
2. Each game should have a unique game name, which will be formed by concatenating of the players names. Games between the same players are not allowed (i.e. game Allyson-Bob and Bob-Allyson are considered the same).
3. Developing of a nice GUI is not required and generally will not be included into assessment but you might get some bonus points for outstanding solutions

Software Requirements

Overall Requirements

1. The software must be written in Java. It must run on CS Lab computers without any additional software being installed.
2. The Java classes for the software must not be in a package. (That is, no **package x.y.z;** statements.)
3. The program must use client-server architecture with a server process and a process for each user.
4. You have to implement both TCP and UDP protocols in your project.
5. The software must use sockets (TCP) and datagrams (UDP) to communicate between users and the server.

You have to continue this list yourself and develop further requirements as a part of specifications.

Server-Client Protocol Requirements

1. A user connects to the server by sending the following information:
   a. Protocol type (TCP/UDP) and port to connect to.
   b. Player Name (the Player name is one or more characters and may not contain blanks)
   c. Player Name of the person he/she wants to play the game in the same format.
2. If the request to start game is legal (no such session on the server), the server starts session between these users by sending following to one or both users:
   a. Player Name
   b. Ready or not ready to communicate. If a first player has made a request and a second one has not yet, the server informs the first player to wait until the second user issues the same request. If a second player made a request, then the server informs both users that they are ready to communicate.
3. If a request to player is illegal (a session between the same players is active or one of the user has already made a request to play with another player), the server does not start a new session; instead, the server sends an error message to the player.
4. When game is over server sends message to the players with the name and fleet of the winner, and then disconnects them from the server.
5. The protocol must use a textual format for a TCP client and a binary format (be read or written using binary input or output) for a UDP client for each message.

Server Protocol Requirements

1. The server must support multiple simultaneous game sessions.
2. The server must keep track of the state of each separate session.
3. For each session, the server must perform the server side of the server-client protocol specified above.
4. When a user requests to play in a session that does not exist, the server must create the session.
5. When a user requests to play and the request is legal, the server must verify if both partners are ready and inform them applying protocols users are utilizing. That means that players may use different protocols to communicate, and server must perform accordingly.

6. The server process **must** be run by typing this command line:

```
java BattleshipServer <host> <TCPport> <UDPport>
```

where

<host> is the name of the host computer where the server is running and

<TCPport> is the port number on which the server is accepting socket connections

<UDPport> is the port number of the server's datagram mailbox.

NOTE: This means that the server main program class **must** be named BattleshipServer, and this class must not be in a package. The violation of this convention will be penalized.

7. The server process must continue running until externally stopped.

Client Protocol Requirements

1. The player can participate in one game session only at the same time.
2. The player must perform the client side of the server-client protocol specified above.
3. When a client starts, it must immediately randomly generate and output the locations of own fleet (*see game rules document*) and request a game session to start with an own grid.
4. When a client receives the server's response, it must output the following line on the standard output:

- If it is an initial request and the server is not ready to establish play session:

  ```
  Game with <OtherPlayer> is not available. Hold on ...
  ```

  where <OtherPlayer> is a name of the partner

- If the server is ready to establish a play session:

  ```
  Game with <OtherPlayer> is established ...
  ```

  where <OtherPlayer> is the partner's name.

5. When a player receives an error message from the server in response to a game request, the player must print the error message on the standard output.

6. A player must keep track of the own and target grids and output them next to each other after each shot.

7. After the 15 seconds waiting time, a player generates next shot and sends it to the server along with own grid as the server must know the state of the game.

9. Output Definitions:

- Empty spot - . (dot)
- Occupied spot – 0 (zero)
- Missed shot - * (star)
- Hit shot - X (Capital X)

10. When a player receives a winning message from the server, it outputs it along with both grids

11. The above-specified printouts are the only printouts a player may output.

12. A player must exit after receiving the end of the game message from the server.

13. The player's process **must** be initiated by typing this command line:

```
java BattleshipPlayer <host> <protocol> <port>
<yourName> <otherName>
```

where

<host> is the name of the host computer where the server is running,

<protocol> is the protocols to use "TCP" or "UDP" only (all capital letters),
<port> is the port number on which the server is accepting socket connections or the server's datagram mailbox,

<yourName> is the Player name for this game session, and

<otherName> is the partner name for this session.

NOTE: This means that the player main program class **must** be named BattleshipPlayer, and this class must not be in a package. The violation of this convention will be penalized.

Submission Requirements for Project 1.

(1) Submission deadline 12.01 am, Tuesday January 20, 2004,

(2) Please, use the following command to submit your project:

submit –v icss420 420_02-proj1 <files>

(3) A file named pr1_yourname.* should contain your documentation in doc or txt formats.

(4) A file named pr1_yourname.jar should contain the Java source file for all classes in your program. Each source file should include Javadoc comments describing the class or interface. Each method within each class or interface must include Javadoc comments describing the overall method, the arguments if any, the return value if any, and the exceptions thrown if any. Apply the command jar cvf pr1_yoursurname.jar package.html *.java to create this file. These two files should be submitted by the deadline.

Documentation

Suggested report format

Executive summary (1-2 paragraphs)

Concise description of problem addressed and results

Requirements (1-2 paragraphs)

Your brief understanding of what the instructor requires in this project - informally stated

Specification (1-2 pages)

Precise definition of what you are to do and what results you have to achieve. You might like to formulate it as protocol requirements, separating them into user and server parts. Please, be as specific as possible and provide as much detail as you can here.

Feasibility study (1 page)

Possible solutions, protocols, models and methods

Comparison (at least three comparisons of different options) of solutions (you have to compare at least two in each comparison, e.g. TCP vs. UDP is one example)

Conclusion:

Choose one of each and explain why

Implementation (1-2 pages)

Structure, contents, user interface, limitations, software and hardware requirements, etc. Describe all the classes applied. If you used a code from some library, provide the reference to this library.

Product testing (2-3 pages)

Compilation:

- State the time of your final compilation, hardware and software environment, messages (if any) received and the result of your compilation.

Testing:

-Actual executed test cases must be described briefly (you should run and describe at least 10 test cases). Briefly describe how you designed your tests: your main concept and ideas, why you tested that and this, what you expected

- The results of your tests
- Evaluation of your product performance and reliability based on your test results

Development process documentation (1-2 pages for group projects only):

Describe the process of the software development and testing; briefly describe who did what and how your team work was organized.

Attachment 1

BATTLESHIP

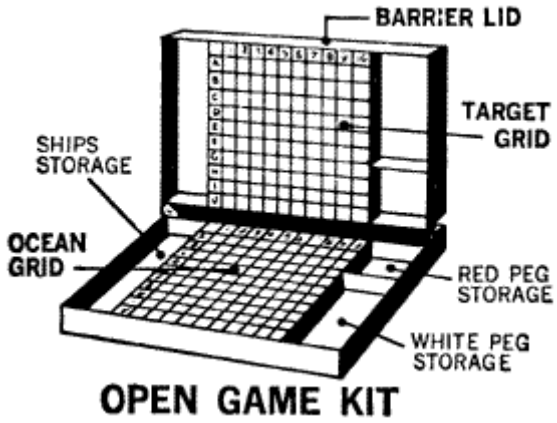
A Two Player Game

Rules for Desktop Game are taken from:
http://www.centralconnector.com/GAMES/battleship.html

BATTLESHIP is the old favorite Navy Game where two players try to sink each other's hidden ships. It is conveniently packaged in self-contained sturdy plastic game kits, complete with storage compartments for the small ships and marker pegs. It can be set up, ready for play, in minutes and quickly stored away when not in use. In this modern, compact form, BATTLESHIP will provide pleasant entertainment for the whole family.

Players place their "fleet" of 5 ships on their "ocean", hidden from the opponent's view. Taking turns, players call out their "shots" attempting to get "hits" on the opponent's ships in order to sink them. Strategy and some luck must be combined to be the first to locate and sink all 5 opponent's ships to win the game.

TO PREPARE THE KITS
The two plastic game kits (I red and I blue) are used for playing the game and storing of the small parts. The diagram below of an OPEN KIT shows the various parts.



OPEN GAME KIT

The lid of the box acts as a BARRIER SCREEN to block the view of the opponent and contains the TARGET GRID for marking a player's shots. The bottom of the box contains the OCEAN GRID for placement of a player's fleet. It also contains bins for storage of the SHIPS and MARKER PEGS.

The SHIPS and PEGS are supplied on "runners". Remove these by carefully TWISTING until they break off. Each kit should have FIVE SHIPS as follows:
"Carrier" (5 holes), "Battleship" (4 holes), "Cruiser" (3 holes), "Submarine" (3 holes), and

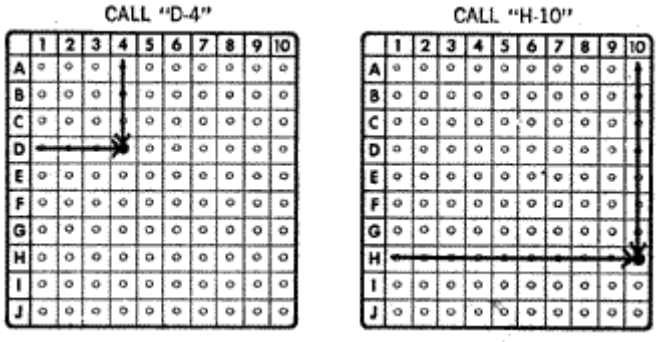
"Destroyer" (2 holes). Each kit should have 2 runners of WHITE pegs (84 pegs) and I runner of RED pegs (42 pegs). Each player has a kit.

THE OBJECT OF THE GAME is to be the first player to sink all five of his opponent's ships.

RULES FOR THE BASIC GAME
(one shot in a turn)

SET UP THE FLEET
I. Two players sit FACING EACH OTHER, each with his kit on the fable in front of him. They open their BARRIER LIDS so they cannot see the OCEAN GRID of their opponent's kit. The lids are kept open all during the playing of the game.
2. Each player SECRETLY places his fleet of 5 ships on his OCEAN GRID. The BOTTOM of each ship has two "anchoring pegs" which must be pushed THROUGH the holes in the OCEAN GRID for placing them, Ships may be placed in ANY HORIZONTAL (back and forth) or VERTICAL (up and down) position - but NOT DIAGONALLY. The ship's "anchoring pegs" will NOT fit in the grid holes if placed diagonally. All holes on the TOP of the SHIPS must be lined up over holes on the OCEAN GRID. DO NOT place a ship so that a part of it is overhanging the grid holes or over letters and numbers.
3. When both players have placed their 5 ships as desired, they announce "READY." From then on, during the game, they MAY NOT change the position of any ship. To do so would be cheating!

CALL OUT THE SHOTS
In this BASIC Game, the players call out ONE shot each turn to try to hit an opponent's ship.
I. The player with the RED kit fakes the first shot. Players then alternate, taking one shot at a time (red, blue, red, etc.).
2. A shot is made by calling a LETTER and a NUMBER to locate which hole in the opponent's OCEAN GRID that shot is to be placed. That hole is located by going straight across, horizontally, from the called LETTER (printed on the side) and down, vertically, from the called NUMBER (printed across the top). The diagrams below show examples of how shots are located. The black dot shows where the shot is placed.



3. When a shot is called, the opponent immediately tells, the player whether it s a "hit" or "miss". If is a "hit" if the called hole on his OCEANGRID is covered by a ship; and a "miss" if no ship occupies

that hole. If the shot is a "hit", the opponent fells the player what KIND of SHIP was "hit" (cruiser, carrier, etc.)

MARK SHOTS WITH PEGS
1.After a player has called his shot and found out whether it s a "hit" or "miss," he places a marker peg in his TARGET GRID (the one in the lid) -a WHITE peg for a "miss" and a RED peg for a "hit", to mark the location of that shot. This will guide him in placing future shots and prevent him from calling the same holes more than once.
2. A player does not have to mark his opponent's "misses" with white pegs, but, he MUST MARK any 'hits" that the opponent makes on his ships with a RED peg. When a hit has been made on a player's ship, he places a RED peg in the SHIP at that location on his OCEAN GRID.
Examples of marking the shots -

a. John calls "F-44' to Harry. Harry announces f as a "miss". John places a WHITE PEG at "F.4" on his TARGET GRID. Harry does not place a peg in his kit.
b. Harry calls "H-6" to John. John announces if as a "ht" - "on a Destroyer." Harry places a RED PEG at "H-6" in his TARGET GRID. John places a RED PEG in a hole of his Destroyer at "H-6" on his OCEAN GRID.

SINK THE FLEET
1. Players continue taking turns, Calling shots and marking them.

2. Whenever a ship has received enough "hits" to fill all of ifs holes with RED PEGS, if is SUNK and is removed from the OCEAN GRID. The player whose ship is sunk must announce if to his opponent.

3. The number of "hits" each ship must receive to be SUNK is as follows: Carrier - 5 hits, Battleship - 4 hits, Cruiser - 3 hits, Submarine - 3 hits, Destroyer - 2 hits.
4. It is expected that players will be HONEST in announcing "hits" when they are made. Occasionally players may make a mistake in calling a hole they didn't mean or in locating the correct hole called. If a player feels an error has been made, he may call a TRUCE - and stop the game temporarily to review shots he has made in past turns. He can easily do this by calling out the location of the pegs he has placed on his TARGET GRID and asking the opponent to verify the "hits" and "misses" he has marked.

WIN THE GAME
The first player to sink all 5 of his opponent's ships is the WINNER.

RULES FOR THE SALVO GAME
The SALVO game version is recommended for more experienced players who have become familiar with the basic game. It differs mainly in how many shots are taken in a turn by each player. Use the same rules as in the Basic Game except:

1. Each player at the start takes a SALVO of 5 SHOTS in his turn. As he calls out each shot, he marks them with WHITE pegs in his TARGET GRID. At the END of the SALVO, his opponent announces which shots were "hits" and on which ships. For example, John called a SALVO of 5 shots at F-4, F-6, F-10, A-2 and A-4. Harry announced one "hit" on his Destroyer at F-6, and two hits on the Submarine at A-2 and A-4.

2. After learning of any "hits", the player changes his WHITE peg at that location to a RED peg, to help guide him for future shots. The opponent must continue to place RED pegs in the holes of any ships that were hit.

3. Whenever a player has had one of his ships SUNK, he LOSES ONE SHOT for his next salvo. As his ships are removed from the game, the shots for each salvo are reduced. For example, if Harry has had 2 of his ships SUNK, he may only shoot 3 shots in his next turn. One shot is lost for each ship sunk, regard. less of what KIND it is. For example: losing a Destroyer counts as much as losing a Carrier.

ADVANCED SALVO GAME

This game offers the most challenge for the "expert" player. It s played as SALVO, except:

1. After a salvo of shots is called, the opponent simply announces how many HITS were made - but NOT WHERE or on WHAT SHIPS.

2. Since players are never sure which shot was a hit in any salvo, they cannot use RED pegs in their TARGET GRID. If is advised to keep a record of hits for each SALVO on a separate piece of paper.