Concordia University
Computer Science &
Software Engineering
Faculty of Engineering and Computer Science

Concordia University
COMP 248 – Fall 2020
Assignment 4

Due Date:	By 11:55 PM, December 4, 2020
Evaluation:	8% of final mark (see marking rubric at the end of handout)
Late Submission:	none accepted
Purpose:	The purpose of this assignment is to help you learn classes and arrays of object.
CEAB/CIPS ATTRIBUTES:	Design/Problem analysis/Communication Skills

General Guidelines When Writing Programs:

Refer to assignment #1 handout.

Teams: The assignment can be done individually or in teams of 2. Team members must be in the same section. Submit one assignment per team; be sure to have both team members' name in the comments at the top of the assignment.

Battleship Simulation

In this assignment, you will write a program to play a version of the game of Battleship against the computer. Our simplified version of the game will be played on a single 8 by 8 grid. Before the actual game, each player secretly places 6 ships and 4 grenades on the grid. Ships and grenades are a single position on the grid. The position of these ships and grenades are of course hidden from the opponent. Once both players have placed their ships and grenades, the actual game starts. Each player, in turn, "shoots a rocket" on the grid (i.e. calls a position).

- If the rocket (the position called) falls on a position where there is nothing, then nothing happens, and the other player can shoot his/her rocket.
- If the rocket falls on a coordinate where the opponent (or the player) has a grenade, then the player loses a turn, and next time, the opponent will play twice in a row.
- If the rocket falls on a coordinate where the opponent (or the player…) has a ship, then that ship sinks.
- If the rocket falls on a coordinate that has been called before, regardless of what was there before, nothing happens. (So, for example, a grenade can only explode once).

The goal of the game is to sink all of your opponent's ships before your opponent sinks yours.

For the sake of simplicity, you can assume that ships and grenades cannot overlap. So, you cannot have 2 grenades on the same position, have 2 ships on the same position, or have a grenade on a ship.

For example, let's consider the following grid:

	A	B	C	D	E	F	G	H
1	S			G			S	g
2		g						s
3	s				s			
4	S		S				S	s
5					s			
6	S	G	g					G
7		G						g
8		s						

Here, one player (identified in black) has his/her 6 ships at positions A1, G1, A4, C4, G4 and A6 and 4 grenades at positions D1, B6, B7 and H6. The other player has his/her ships at H2, A3, E3, H4, E5 and C8 and 4 grenades at positions H1, B2, C6 and H7.

If black starts the game and "shoots a rocket" at position D2 then nothing happens, and we have the following grid.

	A	B	C	D	E	F	G	H
1	S			G			S	g
2		g		*				s
3	s				s			
4	S		S				S	s
5					s			
6	S	G	g					G
7		G						g
8		s						

It is white's turn now. White shoots at G4. The rocket hits one of black's ship. The ship sinks. We now have:

	A	B	C	D	E	F	G	H
1	S			G			S	g
2		g		*				s
3	s				s			
4	S		S				*	s
5					s			
6	S	G	g					G
7		G						g
8		s						

Black shoots at H1. The rocket hits a grenade. Black will lose a turn. We now have:

	A	B	C	D	E	F	G	H
1	S			G			S	*
2		g		*				s
3	s				s			
4	S		S				*	s
5					s			
6	S	G	g					G
7		G						g
8		s						

White shoots at H8. Nothing.

	A	B	C	D	E	F	G	H
1	S			G			S	*
2		g		*				s
3	s				s			
4	S		S				*	s
5					s			
6	S	G	g					G
7		G						g
8		s						*

White shoots again (remember, black lost a turn). White shoots at G1. It sinks another of black's ships.

	A	B	C	D	E	F	G	H
1	S			G			*	*
2		g		*				s
3	s				s			
4	S		S				*	s
5					s			
6	S	G	g					G
7		G						g
8		s						*

White plays now. He is a bit confused and shoots at H4. He just sunk his own ship...

	A	B	C	D	E	F	G	H
1	S			G			*	*
2		g		*				s
3	s				s			
4	S		S				*	*
5					s			
6	S	G	g					G
7		G						g
8		s						*

White shoots at a safe position H1. It is a grenade that already exploded. Nothing happens.

	A	B	C	D	E	F	G	H
1	S			G			*	*
2		g		*				s
3	s				s			
4	S		S				*	*
5					s			
6	S	G	g					G
7		G						g
8		s						*

The game goes on until one player has all his/her ships sunk and he/she loses the game.

Your program must simulate a game of battleship between a human user (H) and a computer (C). The program must:

1. Ask H to place his/her ships and grenades.

For each ship and grenade, 1 coordinate will be entered (a character followed by an integer). You can assume that the user will enter a character followed by an integer. However, your program must check that the coordinates are within the grid and that there is nothing already at that position. If the character is not between 'A' (or 'a') and 'H' (or 'h') or the integer is not between 1 and 8 or there is already a grenade or a ship at that position, then your program must display an error message and let the user try again.

2. Place C's own ships and grenades at random, making sure that the coordinates are valid and that there is nothing at that position already.

3. Display the initial grid hiding all ships and grenades. To display the grid, use the following convention:

a) a position that has never been called should be displayed with the character '_'
b) a position that has been called and hit a ship should be displayed with the character 's' if the ship belongs to the user and 'S' if the ship belongs to the computer.
c) a position that has been called and hit a grenade should be displayed with the character 'g' if the grenade belongs to the user and 'G' if the grenade belongs to the computer.
d) a position that has been called and did not hit anything should be displayed with the character '*'
e) a position that has already been called should be displayed the same way as it was after its first call

4. Let H play first. Ask for the position of H's rocket and modify the grid according to the rules of the game. Then display the new grid showing only the positions that have been called to date using the convention above.

You can assume that the user will enter coordinates of the correct type (i.e. a character followed by an integer), but you must verify that the values are within the grid. If the character is not between ‘A’ (or ‘a’) and ‘H’ (or ‘h’) or the integer is not between 1 and 8, then your program must display an error message and let the user try again.

5. Let C play. Generate a random position for C’s rocket and modify the grid according to the rules of the game. Then display the new gird showing only the positions that have been called to date using the convention above.

To have C play, your program will simply generate a position at random every time.

6. Keep simulating the game and display the grid after each turn, and eventually declare a winner.

7. Once the winner is declared, the grid is then displayed showing the position of all ships and grenades.

Advise:

- Write an algorithm on paper before you start programming. The problem is complex enough that you need to think it through before sitting in front of a computer.
- Your Code should contain at least the implementation of Battleship game grid class, and Battleship driver class. Classes must de defined where appropriate, methods should be decomposed appropriately. In particular, to represent the grid, you are expected to use a 2dimensional array of objects, where each object represents a position in the grid and should be contain at least:
    - the type of element at that position (a ship, a grenade or nothing).
    - the owner of the element (the user or the computer).
    - whether the position has already been called or not.

Here is an example of how your program should behave (assume that the characters highlighted are entered by the user):

```
Hi, let's play Battleship!

Enter the coordinates of your ship #1: A1
Enter the coordinates of your ship #2: A1
sorry, coordinates already used.  try again.
Enter the coordinates of your ship #2: A2
Enter the coordinates of your ship #3: A0
sorry, coordinates outside the grid.  try again.
Enter the coordinates of your ship #3: D2
Enter the coordinates of your ship #4: A5
Enter the coordinates of your ship #5: A6
Enter the coordinates of your ship #6: D5

Enter the coordinates of your grenade #1: H9
sorry, coordinates outside the grid.  try again.
Enter the coordinates of your grenade #1: A1
sorry, coordinates already used.  try again.
Enter the coordinates of your grenade #1: B3
Enter the coordinates of your grenade #2: E3
Enter the coordinates of your grenade #3: G3
Enter the coordinates of your grenade #4: H3

OK, the computer placed its ships and grenades at random.  Let's play.



 position of your rocket: A3
 ship hit.
                _ _ S _ _ _ _ _

                _ _ _ _ _ _ _ _

                _ _ _ _ _ _ _ _

                _ _ _ _ _ _ _ _
                _ _ _ _ _ _ _ _
 position of my rocket: D8
 nothing.
                _ _ S _ _ _ _ _
                _ _ _ _ _ _ _ _
                _ _ _ _ _ _ _ _
                _ _ _ _ _ _ _ _
                _ _ _ _ _ _ _ _
                _ _ _ _ _ _ _ _
                _ _ _   _ _ _ _
                _ _ _ * _ _ _ _
 position of your rocket: A4
 ship hit.
```

```
                    _   _   S   _   _   _   _   _
                    _   _   _   _   _   _   _   _
                    _   _   _   _   _   _   _   _
                    S   _   _   _   _   _   _   _
                    _   _   _   _   _   _   _   _
                    _   _   _   _   _   _   _   _
                    _   _   _   _   _   _   _   _
                    _   _   _   *   _   _   _   _
position of my rocket: H4
boom! grenade.
                    _   _   S   _   _   _   _   _
                    _   _   _   _   _   _   _   _
                    _   _   _   _   _   _   _   _
                    S   _   _   _   _   _   _   g
                    _   _   _   _   _   _   _   _
                    _   _   _   _   _   _   _   _

                    _   _   _   *   _   _   _   _
position of your rocket: H4
position already called.
                        _   S   _   _   _   _   _
                    _   _   _   _   _   _   _   _
                    _   _   _   _   _   _   _   _
                    S   _   _   _   _   _   _   g
                    _   _   _   _   _   _   _   _
                    _   _   _   _   _   _   _   _
                    _   _   _   _   _   _   _   _
                    _   _   _   *   _   _   _   _


...
<after a difficult, but fair game...>
...
position of your rocket: E6
ship hit.  You Win!
                  s   _   S   _   _   _   _   G
                  s               s           G
                  _   g   _   _   g   _   g   g
                  S   _   S   _   _   _   _   G
                  s   _   _   s   _   _   _   _
                  s               S
                  S   _   _   _   _   _   _   G
                  _   _   _   _   _   _   S   _
```

Submitting Assignment 4

What to submit:

Zip the classes source codes (the .java files only please, **not** the entire project) of this assignment as a .ZIP file (**NOT** .RAR) using the following naming convention:

a#_studentID, where *#* is the number of the assignment and *studentID* is your student ID number.

For example, for this fourth assignment, if the assignment is done by one student and the student 123456 would submit a zip file named a4_123456.zip. And if the assignment is done by two students 123456 and 9876543, they need to submit only one zip file named a4_123456_9876543.zip

How to submit:

For sections P&R&S&T, please check your Moodle course webpage and for section EC please check your eConcordia webpage for instructions on how to submit your assignment.

Evaluation Criteria for Assignment 4

Source Code	
Comments (3 pts.)	
Description of the program (authors, date, purpose)	0.5 pt.
Description of classes, methods, variables and constants	1 pt.
Description of the algorithm	1.5 pt.
Programming styles for all classes (3 pts.)	
Use of significant names for identifiers	1 pt.
Indentation and readability	1 pt.
Welcome Banner/Closing message/grid display	1 pt.
Part A: Implementation of the Battleship game grid class (7 pts.)	
Implementation Grid using array	3 pts.
The correctness of placement of pieces	2 pts.
Display correct grid and information	2 pt.
Implementation of Battleship driver (7 pts.)	
Creation of the grid and pieces	2 pts.
The correctness of game setup (player: the user and computer)	3 pts.
Correctness of the code	2 pts.
TOTAL	**20 pts.**