Ira A. Fulton Schools of Engineering

school of computing, informatics, & decision systems engineering

CSE 110 – Assignment #8

Maximum points: 20 pts

Topics

- 2D Array (Chapter 6)
  - Object instance as an element of array.
- File I/O (Chapter 7)
  - Extract text information from a .txt file
  - Regular Expression and delimiter
  - File object
  - Exception Handlings

Your programming assignments require individual work and effort to be of any benefit. Every student must work independently on his or her assignments. This means that every student must ensure that neither a soft copy nor a hard copy of their work gets into the hands of another student. Sharing your assignments with others in any way is **NOT** permitted. Violations of the University Academic Integrity policy will not be ignored. The university academic integrity policy is found at http://www.asu.edu/studentlife/judicial/integrity.html

Use the following Guidelines:

- Give identifiers semantic meaning and make them easy to read (examples numStudents, grossPay, etc).
- Keep identifiers to a reasonably short length.
- User upper case for constants. Use title case (first letter is upper case) for classes. Use lower case with uppercase word separators for all other identifiers (variables, methods, objects).
- Use tabs or spaces to indent code within blocks (code surrounded by braces). This includes classes, methods, and code associated with ifs, switches and loops. Be consistent with the number of spaces or tabs that you use to indent.
- Use white space to make your program more readable.

Important Note:

All submitted assignments must begin with the descriptive comment block. To avoid losing trivial points, make sure this comment header is included in every assignment you submit, and that it is updated accordingly from assignment to assignment. **(If not, -1 Pt)**

```
//*******************************************************
// Name: your name
// Title: title of the source file
```

```
// Author: (if not you, put the name of author here)
// Description: Write the description in your words.
// Time spent:  how long it took you to complete the assignment
// Date: the date you programmed
//*********************************************************
```

Part 1: No writing questions

Part 2: Programming (20 pts)

The goal of this assignment is to develop a program that creates an ocean, which is two-dimensional grid space (x, y), and create ships by reading ships' information from a file. Some ships are in trouble in the ocean. The user moves a ship to save other ships in trouble. Download the following files and use them for this assignment.

***) Do not change any content of the following files.**

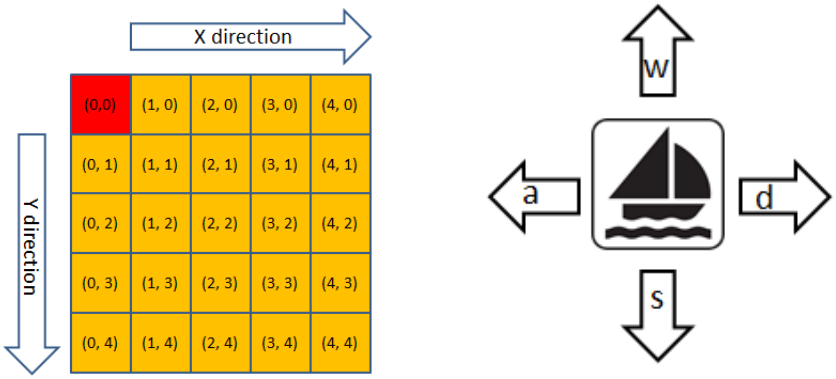
- Assignment8.java
- Ship.java
- shipData.txt
- shipData2.txt

The last two files are input files (text) that will be read from Assignment8 class. Save all files in the same folder. You will be creating a class called **Ocean**. This class should be defined in a file named "**Ocean.java**". The class Ocean will contain a two-dimensional array called "`grids`" of integers and another one-dimensional array called "`ships`" of Ship objects. It also has two static variables, `gridWidth` and `gridHeight`.

Ocean
- grids: int[][] - ships: Ship[] + gridWidth: int + gridHeight: int
+ Ocean(int width, int height, int shipNum) + setShip(String str) : void + isSafe() : boolean + getShip(int shipID) : Ship + message(int shipID, char commond) : void + updateSafe() : void + printInfo() : void

As shown in the image below, the x direction is from left to right, and the y direction is from top to down. The left top is represented as (x, y) = (0, 0). The image shows a case in which the grid size is 5 by 5. A ship can move in four directions by reading a corresponding key; up ('w'), left ('a'), down ('s') and right ('d').

Before starting the Ocean class, read the code of Assignment8.java and Ship.java carefully. Without the knowledge of methods in Ship or tasks in the Assignment8, it is impossible to complete this assignment.



The class Ocean **MUST** include the following constructor and methods. (If your class does not contain any of the following methods, points will be deducted.)

- `public Ocean(int a, int b, int c)`

  It instantiates the two dimensional array `grids`. The column and row are defined by the input first and second parameters respectively. It also instantiates the one-dimensional array `ships` of Ship objects with the size specified by the third input. Each ship object is constructed by using the Ship's default constructor, `Ship()`.

  Two static variables, `gridWidth` and `gridHeight` are initialized using the first and second inputs also. **(3 pts)**

- `public void setShips(String[] data)`

  It reads an array of strings and updates all elements in the `ships`. Each string element in the input, which includes the index, x position, y position, name, and safe condition of ship such as "1 5 5 Titanic false", will be used as an argument for Ship's `setShip(String)` method. **(3 pts)**

- `public boolean isSafe()`

  It returns true if and only if all ships are in safe. In other word, it returns false if there is at least one ship that is in trouble/distress. Use the Ship's `getSafe()` method. **(2 pts)**

- `private Ship getShip(int index)`

  It searches a Ship object, which has the shipID equal to the input. Use the Ship's `getID()` method to get the shipID value. **(3 pts)**

- `public void message (int index, char command )`

  It sends a message to a `Ship` object to move a direction. Find a `Ship` with the index by using the Ocean's `getShip(index)` method above, and apply the Ship's `move(command)` method to it. If the `move(command)` returns true, then display the message with the ship's name and its new position like below.

  ```
  d [Queens moved to (3, 1)]
  ```

  Otherwise, it displays the following message. **(3 pts)**

  ```
  The ship cannot move to the direction.
  ```

- `public void updateSafe()`

  It checks and updates the `isSafe` conditions of all ships. If a safe ship is next to the ship in-trouble, then the ship will be saved by setting the `isSafe` as true. Check all ships each other (use two for-loops), and if and only if 1) one ship is safe and another is not safe, AND 2) the distance is 1, then set the `isSafe` of ship in-trouble as true and display the following message. **(3 pts)**

  ```
  The Titanic is saved.
  ```

- `public void printInfo()`

  It updates the value of `grids` and displays symbols corresponding to the values. If there is no ship at the (x, y), the value of `grid[y][x]` is zero and the symbol is '*'. If there is a safe ship at the position, then the value and symbol is the index of ship (use the Ship's `getID()` method). If there is a ship in-trouble, then the value is -1 and the

symbol is 'X'. The following is a case when a ship (id = 1) is at (0, 0) and a ship in trouble is at (4, 4). **(3 pts)**

```
1 * * * *
* * * * *
* * * * *
* * * * *
* * * * X
```

Use only the Java statements that have been covered in class to date. **DO NOT** use any other items (array, array list, vector, etc.) out of the Chapter1-5 and 8. If in doubt, ask. If you use them, then you lose the points of task. Don't copy any code developed by others. Don't give your code to others. Don't use any algorithm, which you cannot understand. Your assignment file is checked by the MOSS (by Stanford Univ.), which is a program to detect cheatings.

Testing Results: Input in red (Not seen at online-submission) .

Program Output 1

```
****** Program (Save ships) Starts ******
Input the width of ocean:5
Input the height of ocean:5
Input the name of data file:shipData.txt
          1   *   *   *   *
          *   *   *   *   *
          *   *   *   *   *
          *   *   *   *   *
          *   *   *   *   X

Command Options --------------------
w: Move up
a: Move left
s: Move down
d: Move right
i: Display the information
?: Display this menu
q: Quit the program
-----------------------------------

****** Save ships in-trouble ******
Input a command:w
Input the index of ship to move.1
        The ship cannot move to the direction.
****** Save ships in-trouble ******
Input a command:a
Input the index of ship to move.1
        The ship cannot move to the direction.
****** Save ships in-trouble ******
Input a command:d
Input the index of ship to move.1
```

```
        d [Queen moved to (1, 0)].
****** Save ships in-trouble ******
Input a command:d
Input the index of ship to move.1
        d [Queen moved to (2, 0)].
****** Save ships in-trouble ******
Input a command:d
Input the index of ship to move.1
        d [Queen moved to (3, 0)].
****** Save ships in-trouble ******

Input a command:d
Input the index of ship to move.1
        d [Queen moved to (4, 0)].
****** Save ships in-trouble ******
Input a command:d
Input the index of ship to move.1
        The ship cannot move to the direction.
****** Save ships in-trouble ******
Input a command:i
        i [Display the information]
          *   *   *   *   1
          *   *   *   *   *
          *   *   *   *   *
          *   *   *   *   *
          *   *   *   *   X

****** Save ships in-trouble ******
Input a command:s
Input the index of ship to move.1
        s [Queen moved to (4, 1)].
****** Save ships in-trouble ******
Input a command:s
Input the index of ship to move.1
        s [Queen moved to (4, 2)].
****** Save ships in-trouble ******
Input a command:s
Input the index of ship to move.1
        s [Queen moved to (4, 3)].
        The Titanic is saved
****** All ships are saved ******

****** End of Program ******
```

/*

Submit your homework by following the instructions below:

/

• Go to the course web site (my.asu.edu), and then click on the on-line Submission tab. Log in the site using the account, which was registered at the first Lab session. Make sure you use the correct email address for registration. This will allow you to submit assignments. Please use your ASU e-mail address.

• Submit the downloaded **Assignment8.java, Ship.java, and your Ocean.java** files together with **shipData.txt and shipData2.txt** on-line. Make sure to choose **HW8** from drop-down box.

• The **MovieSeating.java** should have the following, in order:

- ✓ In comments, the assignment Header described in "Important Note".
- ✓ The working Java code requested in Part #2.
- ✓ All **jave** files must compile and run as you submit them. You can confirm this by viewing your submission results.

Important Note: You may resubmit as many times as you like until the deadline, but we will only mark your last submission. **NO LATE ASSIGNMENTS WILL BE ACCEPTED.**