PPA Assignment 9

Wonjoon Seol, Computer Science with Intelligent Systems, K1631098

December 11, 2016

1 Introduction

In this assignment we simulate the game of Battleship. Every turn a player chooses a coordinate to shot at the hidden ships on the board. If the shot hits a part of a ship, then it is destroyed. Every type of ship has a different number of parts and the player wins if all parts in every ship are destroyed. I need to demonstrate my ability to use inheritance and override necessary methods, especially *equals* and *toString* methods to suit our specification. (Topics from week 1,. . . ,5 and 7,. . . ,10).

2 Pseudocode

2.1 Class Part

Pseudocode 1: This class represents the characteristics of a part of a ship.

```
Initialise private integer row
Initialise private integer column
Initialise private boolean isDestroyed
Define Part
    Set row
    Set column
Define toString
    Initialise String status
    if isDestroyed is true then
        Set status to "[X]"
    else
        Set status to "[]"
    end
    Return status
Define setDestroyed
    Set isDestroyed to be true
Define getIsDestroyed
    Return isDestroyed
Define equals
    Initialise boolean flag to be false
    if supplied object is type Part AND has same row and column with another part then
        Set flag to be true
    end
    Return flag
```

2.2 Class Battleship

Pseudocode 2: This class represents the characteristics of a Battleship.

```
Initialise private ArrayList<Part> part
Initialise private Integer numOfParts
Initialise private boolean isSunk
Define Battleship
    Set numOfParts
    Initialise ArrayList parts type Part
    for integer i between 0 and numOfParts - 1 do
        Initialise Part with supplied row and column(i) and add it to ArrayList parts
    end
Define equals
    Initialise boolean flag to be false
    if supplied object is type Ship AND not Sunk AND has same number of parts then
        Set flag to be true
    end
    Return flag
Define toString
    Initialise partStatus
    for integer i between 0 and size of ArrayList parts - 1 do
        if last iteration then
            Add String returned from ith element of parts to string
        else
            Add String returned from ith element of parts and ", " to string
        end
        Return partStatus
    end
Define hit
    Initialise boolean flag to be false
    Initialise integer destroyedParts to be 0
    for integer i between 0 and size of ArrayList parts - 1 do
        if ith part from parts is equal to Part with supplied x and y co-ordinates then
            ith part from parts are set to be destroyed
            Set flag to "true"
        end
        if ith elements in parts is destroyed then
            Add 1 to destroyedParts
        end
    end
    if destroyedParts is equal to number of parts then
        Set isSunk is true
    end
    Return flag
```

2.3 Class Cruiser

Pseudocode 3: This class represents the characteristics of a Cruiser type Battleship.

```
Define Cruiser
    Call superclass Battleship constructor with supplied row and 4 (4 parts)
```

2.4 Class Frigate

Pseudocode 4: This class represents the characteristics of a Frigate type Battleship.

```
Define Frigate
    Call superclass Battleship constructor with supplied row and 3 (3 parts)
```

2.5 Class Minesweeper

Pseudocode 5: This class represents the characteristics of a Minesweeper type Battleship.

```
Define Minesweeper
    Call superclass Battleship constructor with supplied row and 2 (2 parts)
Define hit
    Initialise boolean flag to be false
    if random number generated from Math library is less than or equal to 0.5 then
        Set flag to be boolean returned from superclass hit method with supplied x and y value
    end
    Return flag
```

2.6 Class Board

Pseudocode 6: This class represents the characteristics of a game board.

```
Initialise private ArrayList<Battleship> ships
Initialise new Battleship with row = 0 and add it to ArrayList ships
Initialise new Cruiser with row = 1 and add it to ArrayList ships
Initialise new Cruiser with row = 2 and add it to ArrayList ships
Initialise new Frigate with row = 3 and add it to ArrayList ships
Initialise new Minesweeper with row = 4 and add it to ArrayList ships
Set boardSize to be 5
Define getShips
    return ships
Define toString
    Initialise String string to be ""
    for integer i between 0 and boardSize - 1 do
        if ship type is Cruiser then
            Add String returned from ith ship of ships with ", []\n" to string
        else if ship type is Frigate then
            Add String returned from ith ship of ships with ", [], []\n" to string
        else if ship type is Minesweeper then
            Add String returned from ith ship of ships with ", [], [], []\n" to string
        else
            Add String returned from ith element of parts and "\n" to string
        end
    end
    Return string
Define hit
    Initialise boolean flag to be false
    if x or y are not greater or equal to boardSize then
        for integer i between 0 and size of ArrayList ships do
            if ith ship in ships is hit then
                Set flag to "true"
            end
        end
    end
    Return flag
Define countShips
    Print Number of Battleships using frequency methods in class Collection
    Print Number of Cruisers using frequency methods in class Collection
    Print Number of Frigates using frequency methods in class Collection
    Print Number of Minesweepers using frequency methods in class Collection
```

2.7 Class Game

Pseudocode 7: This class is going to drive our program.

```
Initilise Board board
Initialise Scanner in
Initialise String nextUserInput
do
    Print number of ships using countShips method
    Print board
    Print ask user to input value
    Set userInput to read next input value
    if userInput is not equal to "quit" then
        Initialise Array String[] string and split the string userInput by space
        if Array size of string is 2 then
            Initialise integer x and set it to be the first element of Array string converted to
              integer
            Initialise integer y and set it to be the first element of Array string converted to
              integer
            if board is hit with the supplied x and y integer co-ordinates then
                Print "Hit"
            else
                Print "Miss"
            end
        else
            Print input value is unable to interpret
        end
    end
while userInput is not equal to "quit"
Close scanner in
```

3 Class Diagram

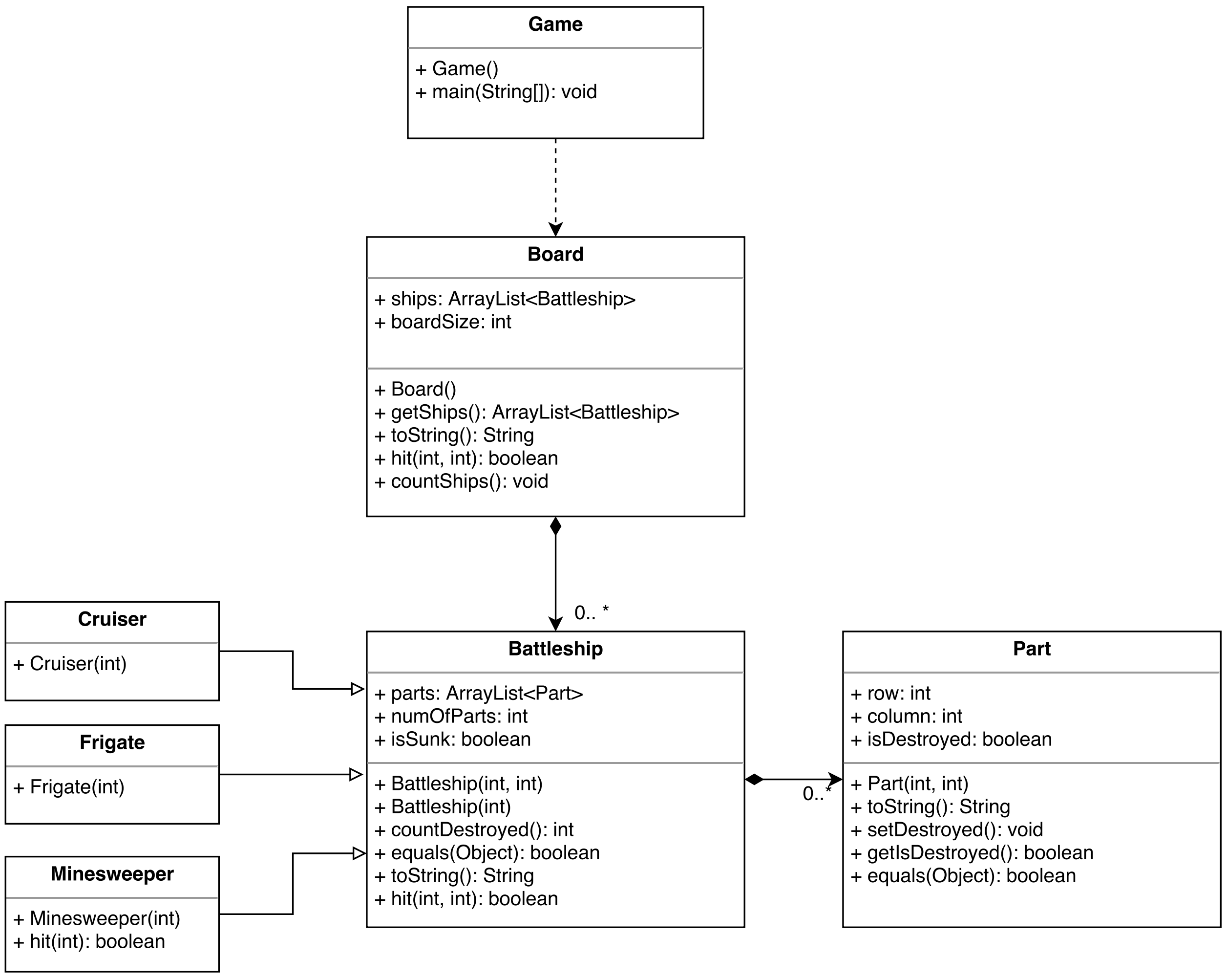

Figure 1: Battleship game Class Diagram.

4 Description

1. Class Part
   This class overrides default *toString* and equals method. *toString* method returns either [x] or [ ] depending on whether the part is destroyed or not and equals methods checks whether they have same rows and columns. No get methods for row and columns were implemented here such that no classes can access these fields from outside.

   Initially, separate method *search*(int x, int y) was defined to perform similar to the *equals* method because I was not exposed to the notion of creating objects on-the-fly. For such a trivial task this seemed to increase the memory complexity of the program unnecessarily. This assignment simulates a game, and a game object tends to have a large number of graphic polygons and the current method might consume too much computing resources compared to the previous solution(comparing a id or a tag for each elements within the ArrayList). However, as far as this assignment is concerned, this method allow us to explore the power of the equals method.

2. Class Battleship
   Again *toString* and *equals* methods are overridden. *equals* method checks whether the supplied object is a type Battleship and then carries on checking ship sunk status and number of parts. My method looks somewhat different to the method shown during the lecture because I wanted to have a single return statement rather than multiple returns. *toString* returns concatenated *toStrings* returned from class Part, separated by commas.

3. Class Frigate, Cruiser, Minesweeper
   Each class inherits from its parents class Battleship. The class Frigate and Cruiser just calls superclass constructor with different number of parts and Minesweeper has one more methods *hit*. This overrides the superclass *hit* method to provide 50% hit chance.

4. Class Board
   The board creates number of ships and stores them inside an ArrayList. This means that they have composite relationships. *toString* methods checks each type of ship and add appropriate number of blank brackets and a newline to make the grid 5x5. *hit* method checks whether a ship stored in the ArrayList *ships* is hit from a supplied coordinate.

   The method *countShips* uses the static method $frequency$ from class Collection to count number of ships. Initially I did not know that the $frequency$ method uses *equals* method to compare objects. Therefore I made a private string field in each ship class, acting as a "tag" and these individual tags in the arrayList *ships* were copied to a new ArrayList. The frequency method then compared this tags to count the number of ships.

5. Class Game
   Our *userInput* variable checks two things, "quit" or integer x or y. When a user types a command it checks whether a supplied string can be separated into two parts and then convert each of these parts into integer values. In this assignment a user is expected to input the right command every time. For example, the code will fail if user types in "2 x". This string will be split into two but then fail to convert these to integer. In order to prevent this, try and catch exception handling can be used but it is outside of the current assessed topics.