Data Structures in Object Oriented Programming: An example

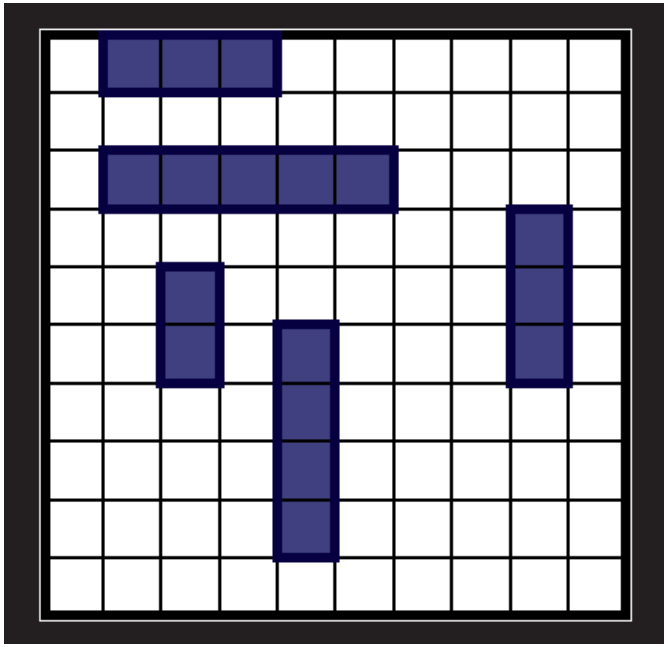
Battleship Game

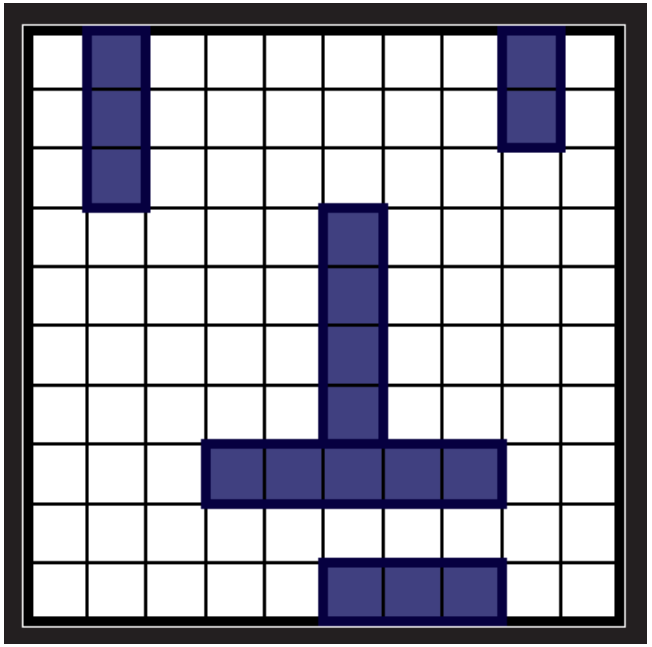
Iterative Development

Tugkan Tuglular, Ph.D.

Battleship Game

- On a board (typically 10 x 10 grid), 2 players “hide” ships of mixed length; horizontally or vertically (not diagonally) without any overlaps. The exact types and number of ships varies by rule.





Battleship Game

- We will be using ships of lengths: 5, 4, 3, 3, 2 (which results in 17 possible targets out of the total of 100 squares).

Type	Size
Aircraft Carrier	5
Battleship	4
Submarine	3
Cruiser	3
Destroyer	2

Battleship Game

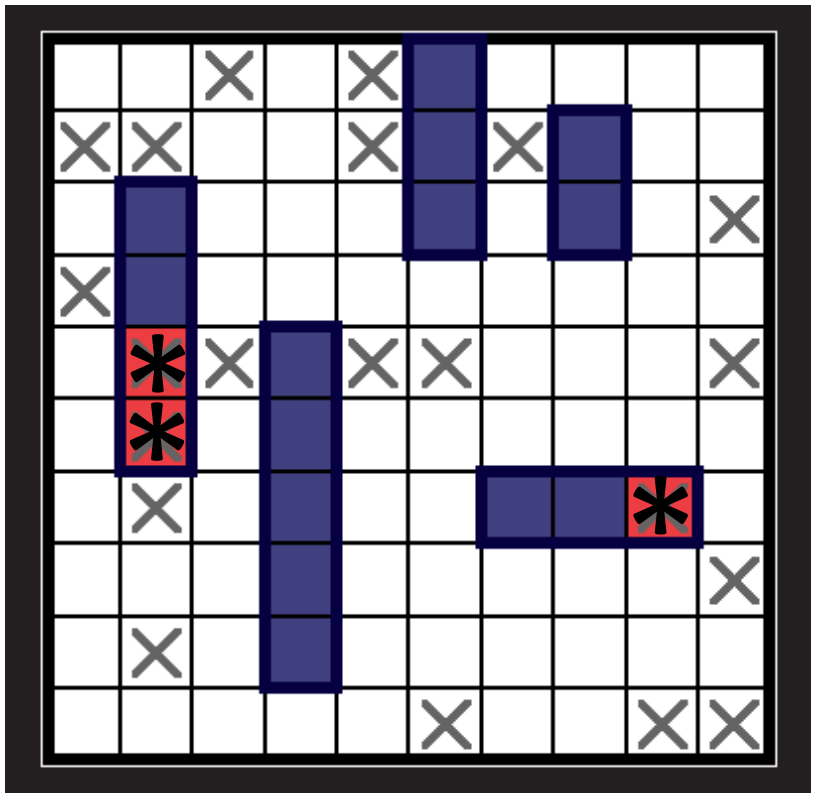
- After each player has hidden his fleet, players alternate taking shots at each other by specifying the coordinates of the target location. After each shot, the opponent responds with either a call HIT or MISS indicating whether the target coordinates have hit part of a boat, or open water.

- Misses are shown by crosses (x) and hits by stars (*).

- The first player to sink his opponent's fleet (hitting every location covered with part of a boat) wins the game.

Battleship Game

- Misses are shown by crosses (x) and hits by stars (*).



Battleship Game

Analysis

Battleship Game

Class

attribute

method

On a board (typically 10 x 10 grid), 2 players "hide" ships of mixed length; *horizontally or vertically (not diagonally) without any overlaps*.

pre-condition

- The exact types and number of ships varies by rule.

Battleship Game

- We will be using ships of lengths: 5, 4, 3, 3, 2 (which results in 17 possible targets out of the total of 100 squares).

Type	Size
Aircraft Carrier	5
Battleship	4
Submarine	3
Cruiser	3
Destroyer	2

Battleship Game

Enumaration

- After each player has hidden his fleet, players alternate taking shots at each other by specifying the coordinates of the target location. After each shot, the opponent responds with either a call HIT or MISS indicating whether the target coordinates have hit part of a boat, or open water.

presentation layer

- Misses are shown by crosses (x) and hits by stars (*).

- The first player to sink his opponent's fleet (hitting every location covered with part of a boat) wins the game.

endGame condition

Battleship Game

Design

Design (Textual) – Iteration 1-2

- BattleshipGame has 2 Board and 2 Player
- 1 Player has 1 Fleet
- 1 Fleet has 1 Carrier, 1 Battleship, 1 Submarine, 1 Cruiser and 1 Destroyer
- 1 Player takes N Shot
- 1 Shot has 1 Target and 1 Response
- 1 Target has 1 Coordinate

Design (Textual) – Iteration 3

- BattleshipGame has 2 Board and 2 Player
- 1 Player has 1 Board and 1 Fleet
- 1 Fleet has 1 Carrier, 1 Battleship, 1 Submarine, 1 Cruiser and 1 Destroyer
- Carrier, Battleship, Submarine, Cruiser, Destroyer have 1 Position
- 1 Player takes N Shot
- 1 Shot has 1 Coordinate and 1 Response

Design (Textual) – Iteration 4

- BattleshipGame has ~~2 Board and~~ 2 Player
- 1 Player has 1 Board and 1 Fleet
- 1 Fleet has 1 Carrier , 1 Battleship, 1 Submarine, 1 Cruiser and 1 Destroyer
- Carrier, Battleship, Submarine, Cruiser, Destroyer have 1 Position
- 1 Player takes N Shot
- 1 Shot has 1 Coordinate and 1 Response
- 1 Board has 1 ResultMatrix
- 1 Board has 1 BoardPresentatiom

Battleship Game

Implementation

BattleshipApp

- Main method
  - Just create an BattleshipGame object (initialize through arguments or a configuartion file)
  - And call its start() method
  - That's all !!!
  - The rest will be taken care of by BattleshipGame object

Iteration 1

Make sure your App is always executable after each iteration.

- BattleshipGameApp
  - main method calls BattleshipGame constructor and its start method
- BattleshipGame
  - just constructor and start()

Iteration 1

Make sure your App is always executable after each iteration.

- BattleshipGame
  - start() creates board1, board2, player1 and player2
- Board
- Player

Iteration 2

Make sure your App is always executable after each iteration.

- Player
  - constructor creates fleet
- Fleet
  - constructor creates fleet ships

Iteration 2 - output

BattleshipGame created
Board 1 created
Board 2 created
Carrier [id=1, size=5] created
Battleship [id=1, size=4] created
Submarine [id=1, size=3] created
Cruiser [id=1, size=3] created
Destroyer [id=1, size=2] created
Fleet 1 created
Player 1 created
Carrier [id=2, size=5] created
Battleship [id=2, size=4] created
Submarine [id=2, size=3] created
Cruiser [id=2, size=3] created
Destroyer [id=2, size=2] created
Fleet 2 created
Player 2 created

Iteration 3

Make sure your App is always executable after each iteration.

- BattleshipGame
  - in start(), ask players to place fleet
- PlaceFleetMethod – Enumaration
  - DEFAULT, FILE, MANUAL, RANDOM
- Coordinate
  - two constructors for two different formats and their converters

Iteration 3

Make sure your App is always executable after each iteration.

- attribute ArrayList<Coordinate> position to all ships
- Player
  - placeFleet(PlaceFleetMethod placeFleetMethod)

Iteration 4

Make sure your App is always executable after each iteration.

- Board
  - holds the matrix now
- BoardPresentation
  - Prints board to the screen

Iteration 4 - output

…
Player 1

```
 0 1 2 3 4 5 6 7 8 9
A #         # #
B #
C #   #
D #   #
E #   #
F     #   #
G       #
H       #   #
I         #
J         #
```

Player 2

```
 0 1 2 3 4 5 6 7 8 9
A #         # #
B #
C #   #
D #   #
E #   #
F     #   #
G       #
H       #   #
I         #
J         #
```

Iteration 5

Make sure your App is always executable after each iteration.

- BattleshipGame
  - in start(), build the game loop (modify it as required)

```
while (!endOfGame) {
    processInput();
    update();
    render();
}
```

Iteration 5

Make sure your App is always executable after each iteration.

- Shot
- Response – Enumaration
- Player
  - takeShot, getResponse, checkShot, checkEndOfGame
- Board
  - setResultMatrixCell

Many Many Iterations To Come

Make sure your App is always executable after each iteration.

- Board – ADT
- Fleet ships – Inheritance
- PlaceFleetFromFile – DAO
- BoardPresentation and takeShot – Swing
- and many TODOs in the source code
- and then REFACTORING
- and HumanPlayer & ComputerPlayer – Inheritance

To compare

- https://github.com/dariajung/battleship/tree/master/src
- https://glot.io/snippets/eeoqad7syj
- https://amorykcwong.ca/ICT/ProgJava/BattleShip.java
- http://www.progressivejava.net/2012/10/Battleship-game-in-Java--How-to-program25.html