Department of Computer Science, Mathematics & Physics

COMP2232 – Object-oriented Programming Concepts
Semester 1, 2020-21: Assignment #1

Read this entire document before starting your assignment!

Zhen Hunters

The purpose of this assignment is to provide exposure to arrays, classes, encapsulation, messages and inheritance while increasing familiarity with their usage. It will also lay the foundation for building a framework for a two-player board game called ***Zhen Hunters***. The goal of the game is for either the Hunters to eat enough Zhens or the Zhens to surround the Hunters such that they cannot move.

The Game Grid

The grid is made up of 48 squares. The squares are arranged in such a way that they form a cross (*See Figure 1*):

- Maximum width of cross: 8 squares.
- Maximum height of cross: 8 squares.
- 16 squares in the centre.
- 8 squares on each arm.

There are 4 additional squares at **each corner**. These contain brambles and other game pieces are unable to occupy these positions.

*	*	Z	Z	Z	Z	*	*
*	*	Z	Z	Z	Z	*	*
Z	Z	Z	Z	Z	Z	Z	Z
Z							Z
Z			F	B			Z
*	*					*	*
*	*					*	*

Fig. 1: Layout of board for 20 zhens.

The Game Pieces

There are two types of **moving** pieces in this game: Hunter and Zhen.

Hunter:

- **Number on grid:** two.
- **Movement:**
  - Horizontally. vertically or diagonally.
  - Each Hunter moves once per round.
- **Eating:**
  - Consumes a Zhen by jumping over it in a straight horizontal or vertical line (not diagonally) into an empty cell beyond & adjacent to the Zhen.
  - Can perform multiple jumps, if available, resulting in the eating of multiple Zhens.
  - Energy level decreases each time a Hunter has completed three rounds without eating.

Zhen:

- **Number on grid:** 16, 18, or 20.
- **Movement:**
  - Horizontally or vertically only.
  - Only two Zhen may be moved per round.
- **Eating:**
  - Zhens which are eaten are removed from the grid.
  - May **not** eat a Hunter or perform jumps resulting in movements of more than one square.

Game Grid Setup

- The two players will be prompted to enter their names. One player controls the Hunters and the other player controls the Zhens.
- The zhen-player will also be prompted to select the number of Zhens they wish to play with (valid options: 16, 18 or 20). Hunter-player indicates characters to represent their two hunters (valid options: any letter other than 'Z').
- The grid is drawn, the brambles, Hunters and Zhens are placed on the grid in the correct start formation. (*See Figure 1*) For 18 or 16 Zhens, the usage of squares is removed vertically upwards, maintaining horizontal balance.
- Hunters each start with an energy level of 4. Their **maximum** level is 6.
- A summary is displayed below the board indicating the energy level of each Hunter and the number of Zhens left on the grid.

Gameplay

- At the **start** of the game, the grid is setup as above.
- The zhen-player moves first, selecting two Zhens and indicating the direction for each. The hunter-player then moves their two Hunters. This completes one round.
- The players take turns moving their pieces. Players may only move their pieces in valid directions based on the type of their piece.
- For each Zhen eaten by a Hunter, that Hunter gains a single energy point. Maximum energy level is not to be exceeded. Overeating will result in "Death by Gluttony".
- A Hunter loses an energy point each time it has completed three rounds without eating.

End-game Conditions

- **Zhens Win:** Both Hunters are trapped. This occurs when Hunters have no moves available i.e. no adjacent empty spaces and no possible moves to eat their way out.
- **Hunters Win:** There are too few Zhens to overwhelm and trap both Hunters.

Instructions for THIS assignment

For **this assignment**, create each of the classes below to start construction of the game. Each class is to be placed in its own file named using the format: ***<class-name>.java***

GamePiece	
Field	**Description**
`String type`	Contains the type of the game piece. The descriptor will be a word.
`char symbol`	Contains the character which will be used to represent the game piece on the grid.
`int rowPos`	Contains row in which game piece is positioned.
`int colPos`	Contains column in which game piece is positioned.
Method	**Description**
`GamePiece()`	Constructor used to initialize `type` to **"?"** and `symbol` to **'?'.**
`Mutators & Accessors`	Accessor & mutator methods for **each** of the data members.

Bramble (inherits from **GamePiece**)	
Method	**Description**
`Bramble()`	Constructor used to initialize `type` to "**bramble**" and `symbol` to '*'.

Hunter (inherits from **GamePiece**)	
Field	**Description**
`int energyLevel`	Indicates current evergy level of Hunter.
`int MAXENERGY`	**Constant** used to indicate maximum energy level of Hunter.
`char hName`	Character chosen by player for Hunter.
Method	**Description**
`Hunter()`	Constructor used to initialize type to "**hunter**" and symbol to '**H**'.
Mutators & Accessors	Accessor & mutator methods for **each** of the data members.

Zhen (inherits from **GamePiece**)	
Method	**Description**
`Zhen()`	Constructor used to initialize type to "**zhen**" and symbol to '**Z**'.

Player	
Field	**Description**
`String name`	Contains the name of the player.
Method	**Description**
`Player(String)`	Constructor used to initialize name to the string parameter.
Mutators & Accessors	Accessor & mutator methods for **each** of the data members.

GameGrid	
Field	**Description**
`int numRows`	Number of rows on grid. Default size is 8.
`int numCols`	Number of columns on grid. Default size is 8.
`GamePiece[][] grid`	2-d array holding GamePiece objects. Size is `numRows` x `numCols`.
`Player player1, player2`	Player objects.
`int numZhens`	Number of zhens selected for play. Valid values: 16, 18, or 20 only.
`int[] hunter1Coord`	Current position of Hunter 1 on grid.
`int[] hunter2Coord`	Current position of Hunter 2 on grid.
`int numHunters`	Contains the number of Hunters still in play.
Method	**Description**
`GameGrid()`	Default constructor which initializes data members as follows: • Rows and columns to 8 • Size of grid to that of rows and columns

`boolean addGamePiece(GamePiece piece, int row, int col)`	Method used to add the `piece` to the grid at position represented by `row` and `col`. Returns true if the piece was successfully added. *Recall:* *[1] A piece cannot share a position with another piece.* *[2] A piece contains data members* `rowPos` *and* `colPos`*.*
`void setupGrid()`	Sets up the grid by: • Getting player names and the number of Zhens to play with. • Adding Brambles to the grid corner; the Hunters and Zhens in the correct start formation. [Hint: `addGamePiece(…)`] • Update the members `numZhens`, `numHunters`, `hunter1Coord` and `hunter2Coord` appropriately.

`void displayGrid()`	Displays grid using format below.
	
	`... ... ... ... ... ... ... ...`
	`| * | * | Z | Z | Z | Z | * | * |`
	`... ... ... ... ... ... ... ...`
	`| * | * | Z | Z | Z | Z | * | * |`
	`... ... ... ... ... ... ... ...`
	`| Z | Z | Z | Z | Z | Z | Z | Z |`
	`... ... ... ... ... ... ... ...`
	`| Z | | | | | | | Z |`
	`... ... ... ... ... ... ... ...`
	`| Z | | | F | B | | | Z |`
	`... ... ... ... ... ... ... ...`
	`| | | | | | | | | |`
	`... ... ... ... ... ... ... ...`
	`| * | * | | | | | * | * |`
	`... ... ... ... ... ... ... ...`
	`| * | * | | | | | * | * |`
	`... ... ... ... ... ... ... ...`
	
	`Hunter F energy: 4`
	`Hunter B energy: 4`
	`Num of Zhens: 20`

1. Create a class called **`ZhenHunters`** which contains **`main`** and does the following:

    a. Welcomes the users to the game.
    b. Creates an instance of **`GameGrid`**.
    c. Sets up the grid.
    d. Displays the grid.

DO NOT CHANGE THE FIELD OR METHOD NAMES OR DEFINITIONS. YOU ARE REQUIRED TO USE EACH OF THE FIELDS AND IMPLEMENT EACH OF THE METHODS AS GIVEN. YOU MAY CREATE ADDITIONAL FIELDS AND METHODS, IF NECESSARY, TO COMPLETE YOUR PROGRAM.

Note: Marks will be allocated for documentation.

Deliverables

1. **Assignment 1 is due for submission on Sunday 1st November 2020 by 11:55pm via the Moodle/eLearning submission tool or via tessa.king-inniss@cavehill.uwi.edu (only if the tool does not work).**
2. **Only ZIP files will be accepted.** No other compression types should be used. Failure to comply with this instruction will incur a penalty.
3. You must submit a **Plagiarism Declaration Form** with this assignment.
4. This assignment is worth **15%** of your final course mark.

PLEASE NOTE: The specifications for this assignment are subject to change. You will be notified if any such changes were to occur.