

CSE1322L Assignment 5

Description

Most games that involve movement around a game board will use a grid to track where the user is. The size of the grid is determined by the game. For example, chess uses an 8x8 grid. Whereas minesweeper might use a 16x16 or a 40x40 or bigger. Typically these grids are stored in a two dimensional array.

For this assignment, we won't actually write a full game, we'll just write the board creation, and movement part of a game. You could later add additional logic to implement the game of your choice.

Tasks

- Create an abstract class called Board.
 - It will have a private attribute called rows (int)
 - It will have a private attribute called columns (int)
 - It will have an array of characters called theBoard
 - Add a constructor which takes in how many rows (int) and how many columns (int) that the board should have.
 - Create theBoard (the 2 dimensional array of characters) of that size
 - Add a method getCell which takes an x (int) and y (int) coordinate for theBoard and returns the content of that cell (char).
 - Add a method setCell which takes an x (int) and y (int) coordinate as well as a new value (char) and sets theBoard at position x,y to value.
 - Add a method getWidth() which returns the current width of the board
 - Add a method getHeight() which returns the current height of the board
 - Add a method initializeBoard which takes in a character, and sets all cells of theBoard to that character.
 - Finally add a toString (Java) or ToString (C#) override which returns a string that shows the board.
 - Start with a string that is empty ""
 - Add the line at the top of the board. You'll need to add 2xWidth + 1 '-'s. Then add a "\n"
 - Next you'll need a nested loop to add the center of the board.
 - For each row of theBoard you'll do:
 - Start by adding a '|' for the left boundary.
 - Next add the value of the current cell on this row
 - Next add another |
 - Repeat for each column.
 - Then add a "\n";
 - After you add the values from theBoard, you'll need to add another line of -'s. It should be the same size as the one you added at the start.
 - Look at the sample output below to see how it should look.

- Create a concrete class called Board4x4 which inherits from Board.
 - It will only have a constructor that creates a 4x4 board. You should simply be able to call your parents' constructor.
 - Initialize the board to have all spaces ' '. You already have a method for this.
- Create a concrete class called Board8x8 which inherits from Board.
 - It will only have a constructor that creates a 8x8 board. You should simply be able to call your parents' constructor.
 - Initialize the board to have all spaces ' '. You already have a method for this.
- Create an interface called IMove.
 - It will have 4 methods, none of which take any parameters:
 - moveUp() which returns a boolean.
 - moveDown() which returns a boolean.
 - moveLeft() which returns a boolean.
 - moveRight() which returns a boolean.
- Create a class called BasicGame which implements IMove
 - It will have a private attribute x (int) which holds the players x position.
 - It will have a private attribute y (int) which holds the players y position.
 - It will have a Board called myBoard which is either a Board4x4 or Board8x8 object.
 - It will have a default constructor which takes no parameters and creates a Small Board.
 - It will have an overloaded constructor which takes in a string that will be equal to either "Small" or "Big".
 - If the string is "Small" it should create a 4x4 board. It should set x to 2, and y to 2 as a starting position for the player. It should then put the letter P into the board's cell at position 2,2.
 - If the string is "Big" it should create an 8x8 board. It should set x to 4 and y to 4 as the starting position for the player. It should put the letter P into the board's cell at position 4,4
 - If the string is anything else, it should print an error.
 - It will have a method called moveUp which takes no parameters and returns a boolean.
 - This method will look to see if the player can move up. If the cell above the current location is a valid cell, it will be allowed, otherwise it won't.
 - The current location of the player is stored in x and y. x represents the row, while y represents the column. So to see if the player can move up, you'll see if x-1 is greater than 0.
 - If the player is allowed to move up, first change the current player's cell from P to ' ', then put a P in the new player's cell. Be sure to update x and or y as appropriate.
 - It will have a method called moveDown which takes no parameters and returns a boolean.
 - This works the same as moveUp, expect you'll need to see if x+1 is less than getHeight() on the board.
 - It will have a method called moveLeft which takes no parameters and returns a boolean.
 - This also works the same as moveUp and moveDown, except it'll check the y value and move in the y direction.
 - It will have a method called moveRight which takes no parameters and returns a boolean.

- Same as the other three methods, but moves right.
- It will have an override of toString (Java) or ToString (C#) which prints out the current state of the board. Note you only have to call the toString method on the board.
- In your main class you'll ask the user what size game they want to play, and allow them to move around.
 - Ask the user what size game (Small or Big)? Read in their response and instantiate a BasicGame object passing the size.
 - Use a loop to keep asking the user for a movement until they choose Q.
 - Print a menu for the user:
Q to quit, or move by pressing:
8
4 6
2
 - Read in the user's choice.
 - If the user choose 8, you'll call moveUp, if they choose 2, you'll call moveDown, if they choose 4 you'll call moveLeft, and if they choose 6 you'll call moveRight. If they enter Q you'll stop the program. Otherwise you'll tell them you don't understand what they mean.
 - If their move is illegal you should let them know.

Sample Output:

What size board do you want? (Small or Big)

Small

```

-----
| | | | |
-----
| | | | |
-----
| | |P| |
-----
| | | | |
-----

```

Q to quit, or move by pressing:

8
4 6
2
8

```

-----
| | | | |
-----
| | |P| |
-----
| | | | |
-----
| | | | |
-----

```

Q to quit, or move by pressing:

8

4 6

2

8

```
-----  
| | |P| |  
-----  
| | | | |  
-----  
| | | | |  
-----  
| | | | |  
-----
```

Q to quit, or move by pressing:

8

4 6

2

8

You can't go there!

```
-----  
| | |P| |  
-----  
| | | | |  
-----  
| | | | |  
-----  
| | | | |  
-----
```

Q to quit, or move by pressing:

8

4 6

2

4

```
-----  
| |P| | |  
-----  
| | | | |  
-----  
| | | | |  
-----  
| | | | |  
-----
```

Q to quit, or move by pressing:

8

4 6

2

4

```
-----
|P| | | |
-----
| | | | |
-----
| | | | |
-----
| | | | |
-----
```

Q to quit, or move by pressing:

8

4 6

2

4

You can't go there!

```
-----
|P| | | |
-----
| | | | |
-----
| | | | |
-----
| | | | |
-----
```

Q to quit, or move by pressing:

8

4 6

2

2

```
-----
| | | | |
-----
|P| | | |
-----
| | | | |
-----
| | | | |
-----
```

Q to quit, or move by pressing:

8

4 6

2

6

```
-----  
| | | | |  
-----  
| |P| | |  
-----  
| | | | |  
-----  
| | | | |  
-----
```

Q to quit, or move by pressing:

8

4 6

2

2

```
-----  
| | | | |  
-----  
| | | | |  
-----  
| |P| | |  
-----  
| | | | |  
-----
```

Q to quit, or move by pressing:

8

4 6

2

2

```
-----  
| | | | |  
-----  
| | | | |  
-----  
| | | | |  
-----  
| |P| | |  
-----
```

Q to quit, or move by pressing:

8

4 6

2

Q

Sample Output 2:

What size board do you want? (Small or Big)

Big

```
-----
| | | | | | | | |
-----
| | | | | | | | |
-----
| | | | | | | | |
-----
| | | | | | | | |
-----
| | | | | P | | |
-----
| | | | | | | | |
-----
| | | | | | | | |
-----
| | | | | | | | |
-----
```

Q to quit, or move by pressing:

8

4 6

2

8

```
-----
| | | | | | | | |
-----
| | | | | | | | |
-----
| | | | | | | | |
-----
| | | | | P | | |
-----
| | | | | | | | |
-----
| | | | | | | | |
-----
| | | | | | | | |
-----
| | | | | | | | |
-----
```

4 6

2

8

| | | | | P | | | |

A blank number line for a student to draw a number. It consists of a horizontal dashed line at the top and a series of vertical tick marks below it, spaced evenly apart.

| | | | | | | | |

A blank number line for a student to draw a number. It consists of a horizontal dashed line at the top and a series of vertical tick marks below it, spaced evenly across the width of the page.

4 6

2

8

A horizontal number line with 10 tick marks. Below the number line is a dashed line.

[illegible][illegible]

Q to quit, or move by pressing:

8

4 6

2

4

[illegible]

Q to quit, or move by pressing:

8

4 6

2

4

[illegible]

Q to quit, or move by pressing:

8

4 6

2

4

You can't go there!

```
-----
| | | | | | | | |
-----
| P | | | | | | |
-----
| | | | | | | | |
-----
| | | | | | | | |
-----
| | | | | | | | |
-----
| | | | | | | | |
-----
| | | | | | | | |
-----
| | | | | | | | |
-----
```

Q to quit, or move by pressing:

8

4 6

2

Q

Submitting your answer:

Please follow the posted submission guidelines here:

<https://ccse.kennesaw.edu/fye/submissionguidelines.php>

Ensure you submit before the deadline listed on the lab schedule for CSE1322L here:

<https://ccse.kennesaw.edu/fye/courseschedules.php>

Rubric:

- Abstract class Board (25 points total)
 - Private rows attribute which can hold an int (1 point)
 - Private columns attribute which can hold an int (1 point)

- 2D array of characters called the Board (2 points)
- Constructor which sets rows, columns and instantiates an appropriately sized array on theBoard (4 points)
- getCell() takes in a row and column (both ints) and returns the value of theboard at those coordinates (2 points)
- setCell() takes in a row, column and value and sets theBoard at row,column to value (2 points)
- getWidth() returns the width of the 2d array (2 points)
- getHeight() returns the height of the 2d array (2 points)
- initializeBoard, iterates over each row and column to set each cell to the passed in character (4 points)
- toString/ToString override, successfully renders the board (5 points)
- Board4x4 class (5 points total)
 - Has a constructor which calls parent appropriately to make a 4x4 board (4 points)
- Board8x8 class (5 points total)
 - Has a constructor which calls parent appropriately to make a 8x8 board (4 points)
- IMove interface (10 points total)
 - Correctly defined as an interface (2 points)
 - Has moveUp method which returns a boolean (2 points)
 - Has moveDown method which returns a boolean (2 points)
 - Has moveLeft method which returns a boolean (2 points)
 - Has moveRight method which returns a boolean (2 points)
- Basic Game Class (35 points total)
 - Correctly implements IMove (2 points)
 - Has private x and y attributes (2 points, 1 each)
 - Has theBoard attribute which can hold an object (4 points)
 - Has basic constructor which takes no parameters and makes a small board (2 points)
 - Has overloaded constructor which takes in a string containing either “Small” or “Big” and creates an appropriate sized board. (8 points)
 - Has concrete moveUp method (4 points)
 - Checks if you can move up, and successfully moves you up.
 - Has concrete moveDown method (4 points)
 - Checks if you can move down, and successfully moves you down.
 - Has concrete moveLeft method (4 points)
 - Checks if you can move left, and successfully moves you left
 - Has concrete moveRight method (4 points)
 - Checks if you can move right, and successfully moves you right.
 - Has override of toString (Java) or ToString (C#) which successfully prints the board. (1 point)
- Main class (20 points)
 - Prompts the user asking what size game they wish to play. Reads in answer (4 points)

- Using a loop, continue to ask the user which direction to move, and calls the appropriate move method in the Basic Game object. (10 points)
- Detects if a movement is invalid and warns the user (4 points)
- Successfully stops when the user chooses Q for quit (2 points)