AP Computer Science — Project – Mugwumps

ASSIGNMENT OVERVIEW

In this assignment you'll be implementing a Java version of a classic computer game called "Mugwumps."

This assignment is worth a total of 50 points and is due during your last week of class.

BACKGROUND

For as long as there have been computers, there have been computer games. Whether allowing two people to play against each other, one person to play against the computer, or even one person just trying to play a solo game, playing computer games is fun. You might find that *programming* computer games is even *more* fun.

In the game of Mugwumps, four fictional Mugwump creatures are "hiding" on a 10×10 grid. The user has ten guesses to try to identify the location of the Mugwumps. After each guess, the user is given information on each of the mugwumps, including a) which mugwumps have already been found, and b) the distance between the last guess location and where the remaining hidden mugwumps are located. See the *Sample Interactions* at the end of this document for an example of gameplay.

ASSIGNMENT SPECIFICATION

Write a program **Mugwumps.java** that allows someone to play Mugwumps on the computer. Your project:

1. will have a main method that runs the game
2. might include a `Mugwump` class to manage the four mugwumps, although this isn't required
3. might include a `Board` class to manage the grid of locations, although this isn't required
4. will use a two-dimensional `String Array` to track the state of the game
5. will give the user a brief description of the game at the start
6. will give the user a maximum of 10 guesses to locate the hidden mugwumps
7. after each guess, will update the board, and indicate to the user the state of all the mugwumps
8. if all mugwumps have been found, will end with a message of congratulations
9. if all mugwumps haven't been found after ten guesses, will reveal where the mugwumps were hiding

DELIVERABLES

Mugwumps.zip

This single file will be a zipped directory (folder) of your project. It will include at a minimum your `Mugwumps.java` main program, any other classes you create during the development of your program (possibly `Mugwump.java` and `Board.java`), and possibly a `package.BlueJ` file (if you used BlueJ).

To submit your assignment for grading, copy your file to your directory in `/home/studentID/forInstructor/` at *crashwhite.polytechnic.org* before the deadline.

ASSIGNMENT NOTES

- This program is typical of many turn-based games: the computer presents the current state of the game, the user takes their turn (by guessing a location), the computer presents an updated view of the game state, the user takes another turn, and so on, until the game is won or lost. Which type of *loop* do you think would be appropriate for running the game?

- It's possible to write the game in a single `main` method, but it certainly helps to manage code by breaking logical sections off into their own `static` methods. Consider writing a few "helper methods" such as an `initializeBoard` method, a `displayWorld` method, a `getUserGuess` method, or any other method that you think would help you manage your code-writing. Keep in mind that any objects from the `main` method that need to be modified will have to be passed into these methods as parameters.

- You can also write separate `Board` and/or `Mugwump` classes to help manage the game. There are trade-offs here: using a separate `Board` class allows you to simplify some of your main method, but requires writing and testing a separate class. Writing a `Mugwump` class will require consideration of what that class needs to include as instance variable, methods, etc. (Of course you'll need to consider those things anyway as you write the larger program.)

  The decision of where to put your complexities—in the `main` program, or encapsulated in a separate class—is a design decision that you'll face again and again.

- Another challenge: it's not uncommon for some amount of code to be discarded after trying an approach to a problem that ultimately doesn't work. That's very much part of the process, and not a bad thing, particularly as you are learning. Be patient with yourself, and leave some time for unforeseen circumstances.

GETTING STARTED

1. *Before you start coding*, spend some time reflecting on the game you're about to write, and how you might think you'll need to organize your programs, methods, other classes. Talk to others, talk to the instructor… and write down your initial ideas on paper or on a whiteboard.

2. Consider different strategies of development:
   a. Working in isolation, on your own
   b. Developing ideas with another person, then proceeding on your own
   c. Developing ideas with another person, then continuing to discuss development with them even as you write your own code.
   d. Developing ideas with another person, then "pair-programming": One person types while the other monitors what is being typed, offering ideas and corrections. Then the roles are switched, and the second person types while the first person monitors. In this strategy, a single project is developed by two people in cooperation.
   e. In no case should one person's code be copy-pasted by another person. That's not "development."

3. Larger projects should inspire you to start considering "version control" and "backups." Having multiple copies of your work as it develops over time is a good way of tracking your progress, even if it introduces a new concern: how to manage multiple versions of your work?

A related question: What happens if work is inadvertently lost or modified? Do you have a backup copy somewhere that you can use to recover the previous work you've done? This, too, introduces a new concern: how to manage backups.

QUESTIONS FOR YOU TO CONSIDER (NOT HAND IN)

1. Working with computers is typically considered a solitary activity, and many people consider tech-savvy people to be "geeky" and socially awkward. Is there something inherent in technology or in computer programming that rewards working alone? What are the advantages and disadvantages to working alone on a project?

2. A large-scale project like this differs quite a bit from the day-to-day exercises that are part of this class. Which aspects of a larger-scale project do you enjoy? Which aspects are more challenging?

3. What data do you have on your computer or phone that you absolutely would not want to lose? Photos? Text threads? Music that you've created? Poetry that you've written? Do you have a strategy for keeping copies of that data safe somewhere?

SAMPLE INTERACTIONS

```
Oh, those crafty Mugwumps! There are four of them hiding
on this map, but I'll bet you can find them all, and
practice your Pythagorean problem-solving at the same time!
Guess a coordinate on the map (column, row), and see if
you can find one. The distance to missing mugwumps are
given after each guess.
Choose wisely! You only have 10 guesses to find them all!
Good luck!
 ~~~~~ MUGWUMP LAND ~~~~~
 9 | . . . . . .
 8 | . . . . . .
 7 | . . . . . .
 6 | . . . . . .
 5 | . . . . . .
 4 | . . . . . .
 3 | . . . . . .
 2 | . . . . . .
 1 | . . . . . .
 0 | . . . . . .
   +--------------------
     0 1 2 3 4 5 6 7 8 9

Enter guess #1 as col,row coordinate: 2,4
2,4
Mugwump #0 -- 5.656854249492381
Mugwump #1 -- 3.0
Mugwump #2 -- 1.4142135623730951
Mugwump #3 -- 3.1622776601683795
 ~~~~~ MUGWUMP LAND ~~~~~
 9 | . . . . . .
 8 | . . . . . .
 7 | . . . . . .
 6 | . . . . . .
 5 | . . . . . .
 4 | . . * . . . . .
 3 | . . . . . .
 2 | . . . . . .
 1 | . . . . . .
 0 | . . . . . .
   +--------------------
     0 1 2 3 4 5 6 7 8 9
```

```
Enter guess #2 as col,row coordinate: 5,4
5,4
Mugwump #0 -- 4.123105625617661
Mugwump #1 -- 4.242640687119285
Mugwump #2 -- 4.123105625617661
Mugwump #3 -- 3.605551275463989
 ~~~~~ MUGWUMP LAND ~~~~~
 9 | . . . . . .
 8 | . . . . . .
 7 | . . . . . .
 6 | . . . . . .
 5 | . . . . . .
 4 | . . * . . * . . . .
 3 | . . . . . .
 2 | . . . . . .
 1 | . . . . . .
 0 | . . . . . .
   +----------------------
     0 1 2 3 4 5 6 7 8 9

Enter guess #3 as col,row coordinate: 2,7
2,7
Mugwump #0 -- 8.06225774829855
Mugwump #1 -- 6.0
Mugwump #2 -- 2.23606797749979
Mugwump #3 -- 6.082762530298219
 ~~~~~ MUGWUMP LAND ~~~~~
 9 | . . . . . .
 8 | . . . . . .
 7 | . . * . . . . . .
 6 | . . . . . .
 5 | . . . . . .
 4 | . . * . . * . . . .
 3 | . . . . . .
 2 | . . . . . .
 1 | . . . . . .
 0 | . . . . . .
   +----------------------
     0 1 2 3 4 5 6 7 8 9

Enter guess #4 as col,row coordinate: 2,1
2,1
Mugwump #0 -- 4.123105625617661
Mugwump #1 -- found!
Mugwump #2 -- 4.123105625617661
Mugwump #3 -- 1.0
 ~~~~~ MUGWUMP LAND ~~~~~
 9 | . . . . . .
 8 | . . . . . .
 7 | . . * . . . . . .
 6 | . . . . . .
 5 | . . . . . .
 4 | . . * . . * . . . .
 3 | . . . . . .
 2 | . . . . . .
 1 | . . M . . . . . .
 0 | . . . . . .
   +----------------------
     0 1 2 3 4 5 6 7 8 9

Enter guess #5 as col,row coordinate:
```