PA5 Garden

Due: Wednesday 3/4 by 11:30PM
Submission: All the *.java files you used for your project to Gradescope. Do not submit the entire project. Remember you need to submit the inheritance hierarchy diagram as specified in class by 11:30PM on 2/26.

Overview

The goal of this assignment is to design a class hierarchy to solve a problem, implement that hierarchy, use polymorphism, and use two-dimensional arrays.

What is a class hierarchy you ask? Read about it on Wikipedia. The class diagram you will be submitting will be a simplified version of the UML class diagrams. There are many resources on the internet for UML class diagrams. We will discuss the format expected in class. UML diagrams will continue to come up (in 335, and in projects if you continue to use Java).

Assignment

The problem you will be solving is a garden simulation. The simulation will read commands like PLANT, PRINT, GROW, and HARVEST from a file and execute those commands. The garden that you will be implementing will consist of a number of rows and columns of plots. Within each plot there can exist a single plant, which is represented with a 5x5 grid of cells. Plants are divided up into three different categories: trees, flowers, and vegetables, all of which have unique characteristics. For example, trees grow up, vegetables grow down, and flowers bloom as they grow. See sample inputs/outputs in the PublicTestCases folder.

This is a two-week assignment. The class inheritance diagram will be all that is due after the first week. The rest of the assignment is due after two weeks.

Class inheritance diagram: Create this diagram however you would like (draw it by hand, use software, etc.). You need to create a pdf file of it and submit it to Gradescope. We will discuss the format of this diagram in class.

Your main program, which must be named PA5Main.java, will need to accept the name of an input file on the command line.

The input file will contain all of the garden initialization settings and the commands to simulate. Here is an example input file.

```
rows: 1
cols: 1
```

```
PLANT (0,0) banana
PRINT
GROW 1
print
```

Note that the commands should be case-insensitive. In other words, ″print″, ″PRINT″, and ″Print″ are all equivalent in the input file. Here is the output for the example:

```
> PRINT
.....
.....
.....
.....
..b..

> GROW 1

> PRINT
.....
.....
.....
..b..
..b..
```

The output should be printed to standard out. See PublicTestCases/ for more input and output examples. Note that in the above example, we asked for 1 row and 1 column. So that gives us one plot at (0,0). We then plant a banana in the one plot at (0,0). Since banana is a tree (more on that later), it starts in the bottom middle of the plot.

The following are the types of specific plants that could be planted:

```
FLOWERS      TREES      VEGETABLES
-------      -----      ----------
Iris         Oak        Garlic
Lily         Willow     Zucchini
Rose         Banana     Tomato
Daisy        Coconut    Yam
Tulip        Pine       Lettuce
Sunflower
```

The following are examples of plots for different types of plants. Plants will be represented with ascii characters. Use the lower case version of the first letter of the plant name. For ″Garlic″ use ′g′, for ″Daisy″ use ′d′, etc. Flowers should start in the middle of the 5x5 grid of cells in the plot it is planted in. Each location in a plot is called a cell. Vegetables should start at the top middle. Trees should start at the bottom middle.

```
Rose
.....
.....
..r..
.....
.....

Tomato
..t..
.....
.....
.....
.....

Coconut
.....
.....
.....
.....
..c..
```

Commands

Commands that need to be implemented. See the PublicTestCases/ for more examples.

- PLANT
  Example: PLANT (0,0) rose
  If the PLANT command is read, it should be followed by plot coordinates and the type of Plant to be planted. Use this type to plant the correct subclass of plant into the garden at given plot coordinates. The plot coordinates are given as row and column. Both rows and columns start at 0. Rows go down the screen, and columns go across the screen. Each plot will itself contain a grid of 5x5 cells (represented as characters). There is a restriction that the number of cells across should be less than or equal to 80, therefore the most plot columns allowed is 80/5 or 16.

- PRINT
  Example: PRINT
  If the PRINT command is read, then the entire garden should be printed to standard out.

- GROW
  Example: GROW 1
  If the GROW command is read, then each Plant should grow the specified number of times as seen in the input command. A plant cannot grow out of its plot. No error will happen, but growth should not occur outside the plot boundaries. Plots also cannot run into each other.

- GROW [num] (row,col)
  Example: GROW 1 (2,3)
  Grow whichever Plant is located in the garden at position (row,col) num times. If there is nothing at this position or the position is outside the size of the garden, print, ”Can’t grow there.” and continue.

- GROW [num] [type]
  Example: GROW 1 rose
  Grow only Plants of the specified type num times.

- GROW [num] [plant]
  Example: GROW 1 flower
  Grow only Plants of the specified class num times.

- HARVEST
  Example: HARVEST
  Remove all Vegetables from the Garden.

- HARVEST (row,col)
  Example: HARVEST (2,3)
  Harvest Vegetable at location (row,col). If not a Vegetable or outside of Garden, print, ”Can’t harvest there.” and continue.

- HARVEST [type]
  Example: HARVEST tomato
  Harvests all Vegetables of the specified type. If there are no Vegetables with that type, do nothing.

- PICK
  Example: PICK
  Remove all Flowers from the Garden.

- PICK (row,col)
  Example: PICK (2,3)
  Pick Flower at location (row,col). If not a Flower or outside of Garden, print, ”Can’t pick there.” and continue.

- PICK [type]
  Example: PICK rose
  Pick all Flowers of the specified type. If there are no Flowers with that type, do nothing.

- CUT
  Example: CUT
  Remove all Trees from the Garden.

- CUT (row,col)
  Example: CUT (2,3)
  Cut Tree at location (row,col). If not a Tree or outside of Garden, print, ”Can’t cut there.” and continue.

- CUT [type]
  Example: CUT PINE
  Cut all Trees of the specified type. If there are no Trees of that type, do nothing.

Error Handling

- Some of the commands above specify some error handling.
- The garden should never be more than 80 characters across. If it is, your program should print out the message "Too many plot columns." and then end. Think: How many characters across is each plot?
- Other than the above specified errors, the input can be assumed well formed.

Hints

- We recommend a Garden object with a 2D array or list of plant objects.
- We recommend a Plant class hierarchy of some kind.
- This assignment can be a lot more difficult or a lot easier depending on how well thought out your solution is. I would recommend spending some time before programming thinking about which classes you should have and which classes should inherit from which other classes.

Grading Criteria

For this PA, we are providing a few testcases, but we will also be testing your code on other testcases, make sure you do the same.

Grades for PA4 are due Wednesday 2/26 by 5:00PM. If you have not received your grade by then, it means your UGTA is slacking! Post privately to Clark and me and we will look into it.

Your grade will consist of similar style and code clarity points we have looked for in the first few assignments.

Write your own code. We will be using a tool that finds overly similar code. Do not look at other students' code. Do not let other students look at your code or talk in detail about how to solve this programming project. Do not use online resources that have solved the same or similar problems. It is okay to look up, "How do I do X in Java", where X is indexing into an array, splitting a string, or something like that. It is **not** okay to look up, "How do I solve this programming assignment from CSc210" and copy someone else's hard work in coming up with a working algorithm.