Due Date: 13.12.2024 23:55

CENG211 – Programming Fundamentals
Homework #3

In this homework, you are expected to implement a simple **"Garden Puzzle App"** in Java.
The submitted homework should fulfill the following concepts of:

- Lists and ArrayLists
- Interfaces
- Abstract Classes
- Inheritance
- Polymorphism
- Enumerations

In this application, you are required to create a brain teaser puzzle game that involves a Garden. You control a gardener through a GardenerApp. All menu operations can be handled inside GardenerApp. (Hint: There is no need to create Gardener class or objects.)

The goal of the gardener is to take 7 GardenObjects and place them inside the Garden so that a pre-determined spot on the garden can be populated with certain types of Pollens that are infused with certain Colors.

For example, the puzzle game will randomly tell the gardener to make Tree and Flower Pollens infused with red, blue, and green colors to reach square B5 in the Garden. Thus, the gardener will place the appropriate 7 GardenObjects in the Garden to achieve the designated goal for square B5. Another possible example goal could be Flower and Bush Pollens without any color infusion.

At the start of the game, you should randomly generate the goal requirements. The best-case scenario can be only a single Pollen type. The worst-case scenario can be all 3 Pollen types and all three colors. Only colors by themselves cannot be the goal.

Initially a Garden of size 8 x 6 (8 columns & 6 rows) should be created and placed inside GardenerApp. Each column is represented with a number from 1 to 8 (left to right), while each row is represented with a letter from A to F (top to bottom). You are expected to properly use List interface to keep the GardenSquares in the Garden. GardenerApp also possesses a StorageShed.

You are provided a file called *"storage_contents.csv"* that includes some GardenObjects. These GardenObjects are read from the file and placed inside a StorageShed.

A GardenObject can be either a GardenPlant, a LightSource, or a Statue. Statues are not provided inside the "storage_contents.csv" file. Instead, you should create 7 Statues with statueIds from 1 to 7. Then, you are going to randomly place them inside the Garden after initializing it.

There are 3 types of GardenPlants: Flower, Tree and Bush. GardenPlants in the csv file will have unique names, unique ids, and areaOfPollenSpread values.

There are 3 types of LightSources: SmallLamp, LargeLamp, and Spotlight. LightSources in the csv file will have unique ids, colors, and areaOfLightReach values.

GardenPlants must be searchable by types, ids, names and areaOfPollenSpread values. LightSources must be searchable by types, ids and areaOfLightReach values. (Hint: Use Searchable interface.)

The Garden is far away from the StorageShed. Additionally, the gardener can only **carry 7 GardenObjects to the Garden**. However, the gardener does not know the locations of the target square and statues. So, he must choose the GardenObjects while only being aware of the requirements for the goal square. While selecting the 7 GardenObjects, the gardener should be able to search them according to his criteria instead of viewing all of them at the same time. After carrying the objects, the gardener stores them inside the gardenStorage in the Garden.

After arriving at the Garden, the gardener starts placing the GardenObjects from the gardenStorage. These objects should be placed in such a way that causes the required Pollens infused with the required colors to reach the goal square.

After placing all GardenObjects or when the gardener simply chooses to stop placing them, he starts to *wait()*. Once he starts waiting, all LightSources *lightUp()* and all GardenPlants *bloom()*.

The gardener should keep in mind the following rules while placing any object in the Garden:

- Each GardenSquare must be able to contain a single GardenObject or PollenCloud. A GardenSquare cannot have both of them at the same time.

- Flower Pollens spread 2, 3 or 4 squares diagonally. Tree Pollens spread 3, 4 or 5 squares horizontally. Bush Pollens spread 1 or 2 squares vertically.

- The spread of Pollens happens across both sides of the plants. For example, a Bush with 1 square spread would result in both direct upper and lower squares of the Bush to have Pollens.

- SmallLamps light 1, 2 or 3 squares to only their right side. LargeLamps light 3 or 4 squares to only their left side. Spotlights light 4 or 5 squares downwards.

- All SmallLamps are blue-colored except one. All LargeLamps are green-colored except two. All Spotlights are red-colored except one.

- To combine different types of Pollens into a single square, a plant should be placed in the path of Pollens of another blooming plant. **For example,** if a Bush is planted directly to the left of a Tree, all Pollens coming out of that Tree will belong to both types of plants. Also, you don't need to consider which plant blooms first to affect another. You can simply think enough time passes for all plants to affect each other as long as they are in each other's area of effect. The Pollen contents of a square are stored in PollenClouds in GardenSquares.

- The previous rule also applies to infusing Pollens with a color. **For example,** if a blue-colored SmallLamp is placed directly to the left of a Tree, all Pollens coming out of that Tree will be infused with blue color. As those Pollens are continued to be spread by other plants they will continue to be infused with blue color. The infused colors in a PollenCloud can be contained in a separate list independent from the types of Pollens.

- All GardenObjects including Statues block the passage of both light and Pollens including diagonal directions.

The puzzle game ends by revealing all the contents of GardenSquares in the Garden. If the target square contains the required types of Pollens infused with the required colors, the result is announced as **SUCCESSFUL**. Otherwise, the result is **UNSUCCESSFUL**. Since there are random factors involved in this game, it is not a problem if you fail to reach a successful scenario while testing the app.

Example Scenario Output:

```
Welcome to Garden Puzzle App. Your goal Square needs BUSH pollens infused with
GREEN color(s).

==> Please search for Garden Objects from the Storage Shed. You can take up to 7
object(s).

Please select the type of object ([1] Plant, [2] Light): 2

Please select search filter for Light Sources ([1] Light type, [2] Light id, [3]
Color, [4] Area of reach): 4

Please enter a light area of reach value between 1 and 5: 3

The Light Sources that fit the given criteria:
- Type: Small Lamp, Id: L2, Color: blue, Area of reach: 3
- Type: Small Lamp, Id: L3, Color: red, Area of reach: 3
- Type: Small Lamp, Id: L4, Color: blue, Area of reach: 3
- Type: Large Lamp, Id: L8, Color: green, Area of reach: 3
- Type: Large Lamp, Id: L10, Color: blue, Area of reach: 3
- Type: Large Lamp, Id: L11, Color: green, Area of reach: 3
--> Please enter the id of the Light Source you would like to take: L8

You have taken 1 Garden Object(s). Would you like to select another one? ([1]
Yes, [2] No): 1

==> Please search for Garden Objects from the Storage Shed. You can take up to 6
object(s).

Please select the type of object ([1] Plant, [2] Light): 1

Please select search filter for Plants ([1] Plant type, [2] Plant name, [3]
Plant id, [4] Area of spread): 1

Please enter a plant type ([1] Flower, [2] Tree, [3] Bush): 3

The Plants that fit the given criteria:
- Type: Bush, Id: P14, Name: Deciduous, Area of reach: 1
- Type: Bush, Id: P15, Name: Evergreen, Area of reach: 1
- Type: Bush, Id: P16, Name: Conifer, Area of reach: 2
- Type: Bush, Id: P17, Name: Yew, Area of reach: 2
- Type: Bush, Id: P18, Name: Arborvitae, Area of reach: 2
--> Please enter the id of the Garden Plant you would like to take: P17

You have taken 2 Garden Object(s). Would you like to select another one? ([1]
Yes, [2] No): 2

==> The gardener carries selected objects to the Garden.

**************************************************************
```

```
Initial Garden Map:

      1       2       3       4       5       6       7       8
   ---------------------------------------------------------------
A |  S1   |       |       |       |       |       |       |       |
   ---------------------------------------------------------------
B |       |       |       |       |  S3   |       |       |       |
   ---------------------------------------------------------------
C |       |       |       |  S2   |Target|       |       |  S6   |
   ---------------------------------------------------------------
D |       |  S7   |       |       |       |       |       |       |
   ---------------------------------------------------------------
E |       |       |       |       |       |       |       |       |
   ---------------------------------------------------------------
F |       |  S4   |       |       |       |       |  S5   |       |
   ---------------------------------------------------------------

--> Your chosen Garden Objects:
- Type: Large Lamp, Id: L8, Color: green, Area of reach: 3
- Type: Bush, Id: P17, Name: Yew, Area of reach: 2

Enter the id corresponding to the Garden Object you would like to place: L8

Enter the location you would like to place the selected Garden Object: E7

Would you like to place another object? ([1] Yes, [2] No): 1

--> Your chosen Garden Objects:
- Type: Bush, Id: P17, Name: Yew, Area of reach: 2

Enter the id corresponding to the Garden Object you would like to place: P17

Enter the location you would like to place the selected Garden Object: E5

==> You have run out of objects to place.

==> The gardener starts waiting. All lights are lit up. All plants start
blooming.

**********************************************************************

Final Garden Map:

      1       2       3       4       5       6       7       8
   ---------------------------------------------------------------
A |  S1   |       |       |       |       |       |       |       |
   ---------------------------------------------------------------
B |       |       |       |       |  S3   |       |       |       |
   ---------------------------------------------------------------
C |       |       |       |  S2   |..u.g.|       |       |  S6   |
   ---------------------------------------------------------------
D |       |  S7   |       |       |..u.g.|       |       |       |
   ---------------------------------------------------------------
E |       |       |       |       | P17   |....g.|  L8   |       |
   ---------------------------------------------------------------
F |       |  S4   |       |       |..u.g.|       |  S5   |       |
   ---------------------------------------------------------------

-- L8: Large lamp with green color
-- P17: Yew bush

====>> SUCCESSFUL
```

Example Garden Square with All Pollen Types and Colors:

```
      1
  --------
A |fturgb|
  --------
```

Legend for Garden Map Notations:

```
S1: Statue 1

Target: The target square with the required pollen types and colors.

L8: Light 8

P17: Plant 17

f: Flower

t: Tree

u: Bush

r: Red

g: Green

b: Blue
```

"." is used inside squares that have a PollenCloud on them when their corresponding color or pollen type is not present in that square.

Important Notes:

1. You can use standard **java.io** packages to read files. You are also free to use anything inside **java.util** packages (including Collections). Do NOT use other 3rd party libraries.
2. You should use **relative paths** (e.g. Files/sample.csv) instead of absolute paths (e.g. C:\\user\\eclipse-workspace\\MyProject\\Files\\sample.csv). Please be sure of it, otherwise there will be no output of your application, and you certainly will lose points.
3. You are expected to **write clean, readable, and tester-friendly** code. Please try to maximize reusability and prevent redundancy in your methods.
4. To increase the readability of the code, you are expected to **comment on** your code as much as possible. You can simply use Copilot to generate comments as well. However, it is up to you to make sure that the comments generated are correct.
5. You should try to **adhere to the object-oriented principles as much as possible**. For example, you should place your files inside a proper package structure instead of placing everything into a single package.

6. We advise you to **plan how to write your code beforehand** since you have access to generative AI tools like Copilot. You should focus on how to apply concepts like polymorphism to your code without depending on the AI.

7. All **incorrect inputs should be handled** by asking the user to enter the input again. After analyzing the example scenario, you can see which inputs could be incorrect if entered differently. **Also, the inputs should not be case-sensitive.** For example, plant id inputs "P17" and "p17" should both be accepted.

8. **Your menus are expected to be almost the same as the given example scenario**. Naturally, there are situations not shown in the example. Your menus should also handle them while not straying too far from the menu style shown in the example. **Significant point deductions will be applied if difficulties arise while navigating your menus.**

Assignment Rules:

1. In this lecture's homework, cheating is not allowed. If cheating has been detected, the homework will be graded as 0 and there will be no further discussion on this.
2. You are expected to submit your homework in groups. Therefore, only one of you is sufficient to submit your homework.
3. Make sure you export your homework as a Visual Studio Code Java project. You can use other IDEs as well; however, you must test if it **can be executed** in Visual Studio Code. It is a good idea to check your exported project on another group member's PC.
4. Submit your homework through Microsoft Teams.
5. Your exported Java Project should have the following naming format with your assigned group ID (which will be announced on MS Teams) as given below:

   G05_CENG211_HW3

   Also, the zip folder that your project is in should have the same name.

   G05_CENG211_HW3.zip
6. Please beware that if you do not follow the assignment rules for exporting and naming conventions, you will lose points.
7. Please be informed that your submissions may be anonymously used in software testing and maintenance research studies. Your names and student IDs will be replaced with non-identifying strings. If you do not want your submissions to be used in research studies, please inform the instructor (Dr. Tuğlular) via e-mail.