CSE105 Coursework 3 2018

Design and code a garden design application in Java. It should graphically display a 'garden' in a JFrame window.

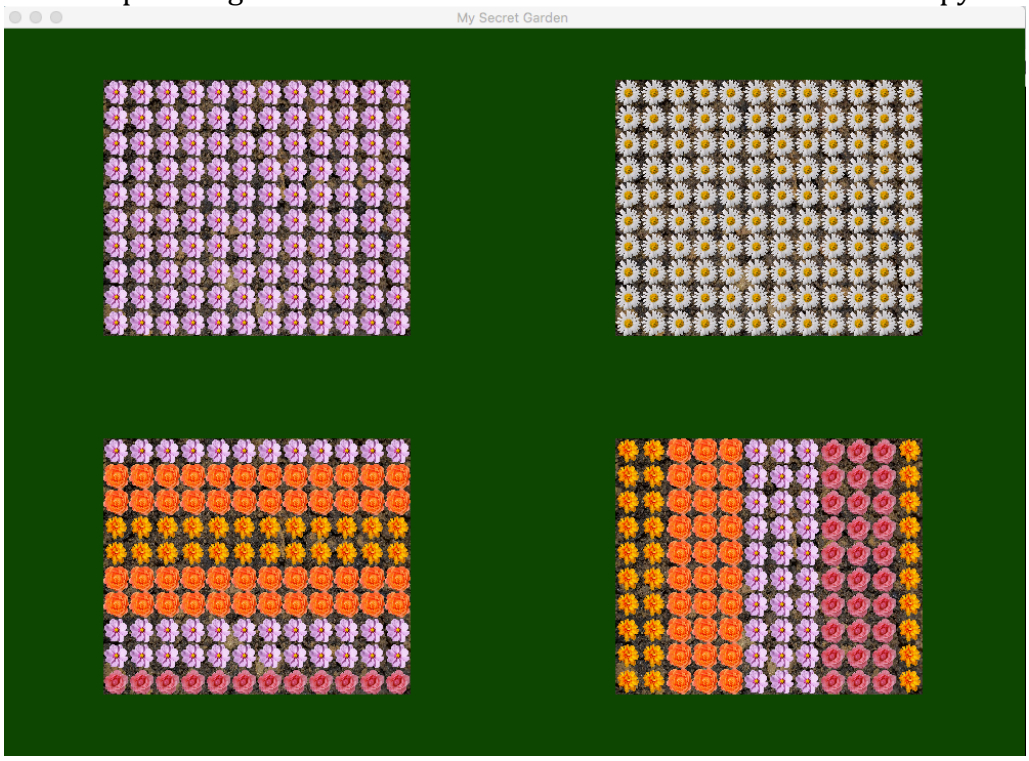
- The 'garden' is contained in a JFrame. It should contain other components for graphics, painting etc.
- It can have a blank background, or pretty grass/a texture.
- It will contain flowerbeds.

Flowerbeds display rows of flowers. They can be all the same, or mixed.

- If mixed, they could be in patterns, or random. The example below shows 2 plain flowerbeds at the top, and 2 flowerbed patterns below
- It is possible to achieve a maximum of 80% without displaying flowers in patterns (ie plain beds with only one type of flower per bed). This would be a good approach for students who have lower coding skills.

You should pay attention to applying principles of OOP as taught this semester, particularly class design and structure. You may re-use and re-purpose code and classes developed and used (by you) in the Labs this semester.

An example of a garden with 4 flowerbeds is shown below. Do not copy it!



The application is controlled via a standard command-line menu system, which allows for the following functionality:

- Add flowerbed
- Remove flowerbed
- Save and exit

Resources

Your application will contain a directory (folder) containing a number of suitable graphic files of flowers (up to 10). These flowers should be suitable in terms of size, background-transparency etc.

Your submitted application MUST include a text file which will load a garden on startup. The garden must contain at least four flowerbeds containing different flowers. If you do not include this, 40 points are immediately deducted.

Task Requirements

It should have the following functionality from a menu input system:

- **Flowerbed Display (10%)**

This is a JFrame window, containing a number of components (panels). Each component is a flowerbed displaying flowers, as shown in the example screenshot above. The number of rows and columns depends on the size of the flower images. When flowerbeds are added and removed, the display should be updated.

- **Add Flowerbed (20%)**

The user must be able to add a flowerbed to the display. They should be able to specify:

- Flowerbed size, setting the height and width of the flowerbed.
- Flowerbed location, setting the position on screen (x and y co-ordinates).
- Type of flower (choosing from a menu of flower image file names, which are in a folder in your application project folder).

To get full marks, they should also be able to display flowers in a pattern (as shown in the lower flowerbeds in the screenshot above).

The user should be able to choose the type of pattern (eg squares or stripes)
And also the details of the pattern, eg:

- Horizontal or vertical stripes?
- How many flowers wide are the stripes?
- How many different colours or flowers for the stripes?

Or similar details for squares etc.

- **Remove Flowerbed (5%)**

Using a command line menu, the user can remove flowerbeds from the display.

How this is achieved is up to you. Top marks will go to solutions which identify the flowerbeds available for removal in the clearest way.

- **Save and exit (5%)**

The user should be able to select an exit option in the menu. This will automatically save the current state of the garden to a text file and exit the program. The file format **must** be a text (.txt) file.

- **Load file on startup (5%)**

When the program is started, garden data should be automatically loaded from a text file, and the display and flowerbeds should be restored to the state of the program when it was last exited.

- **Menu Interface (10%)**

This must be entirely menu driven, displayed through the console window and with keyboard input and all user input must be fully validated. The menu must be clear and usable, with prompts to the user so that it is obvious how to use the program. The flower type menu options should be a list of the flower images in the directory.

- **Input Validation and Exception handling (15%)**

The user input must be fully validated. That means you should check for (and recover from) user input of:

- The wrong data type (eg float or string instead of int)
- The wrong data range (eg negative or very large numbers for location or width)
- Blank input

If any exceptions are thrown as a result of wrong input (eg an int is required, and a String or blank is input) the exception must be handled so that the application doesn't crash, and the user can recover the situation.

There should also be further validation that just the data type. For example, bounds checking:

- Are the x, y values visible on screen?
- Are the size values within a reasonable range?

- **Design Document and Code Quality (30%)**

A design document should be submitted. This should be a word document that gives a simple UML diagram detailing the classes (their names, attributes, methods), and also the relationships between them.

In addition, the code should follow clear conventions for variable names, comments, indentation, and should be easily readable.

Core Task requirements (required to achieve 40%+)

- The submitted file must be a valid NetBeans project folder, including all resources.

- The project must compile and run in Netbeans with no further configuration and debugging.

- **The application you submit MUST include a text file to load a garden on startup. The garden must contain at least four flowerbeds containing different flowers. If you do not include this, 40 points are immediately deducted.**

- The submission instructions must be followed exactly.

Submission

You must submit your work to the submissions drop-box on the CSE105 ICE page before 8.00pm on 14th December 2018.

You must submit:

- One design document (a Word document) with a simple UML diagram detailing the classes and the relationships between them.

- One .txt file for **each** .java class file. These documents must NOT be in a ZIP or RAR archive. The file name must be the class name. Each file must have your name/student number in a comment at the top.

- One ZIP or RAR archive of your **entire Netbeans project folder**. Make sure this contains **ALL** the resources your application needs to run, **including your flower image resources and a txt file to load a garden on startup**.

- If your program does not compile and run from this zip file only, you cannot pass (See Core Task Requirements). The zip file name must start with your student number.

This assignment is individual work. Plagiarism (e.g. copying materials from other sources without proper acknowledgement) is a serious academic offence. Plagiarism and collusion will not be tolerated and will be dealt with in accordance with the University Code of Practice on Academic Integrity. Individual students may be invited to explain parts of their code in person, and if they fail to demonstrate an understanding of the code, no credit will be given for that part of the code.