# Object Oriented Programming

**CSE1100 – Programming Exam**

*A. Zaidman / T. Overklift*

Date: 08-11-2023    Duration: 3 hours    Pages: 6

## Allowed resources

During the exam you are **NOT** allowed to use books or other paper resources. The following resources are available on the exam computer:

- A PDF version of the lecture slides (located on the S-drive of the computer).
- A local (limited) version of Answers EWI.
- The Java API documentation (located on the desktop of the computer).

## Set up your computer

1. Log in to windows using:
   **Username**: `EWI-CSE1100`
   **Password:** `GoodLuck0811!`
2. Once the environment loads you will be asked to provide your **personal NetID and password combination**. Entering these will give you access to a private disk (*P-drive*) where your final exam submission will be stored.
3. Once this is done, please open **My Computer** and navigate to the **Z-drive**, where you will find your exam template. Open the exam template folder, and then open the “OOP Exam Template” IntelliJ file. While IntelliJ starts, you can read the rest of the exam.
4. Once the project has opened you can start implementing your exam in this project! Ensure that you are working in the project on the **Z-drive**, to avoid any issues when handing in!

## Rules

- You are not allowed to use the **internet**.
- **Mobile phones / smartwatches** are not allowed. You must turn them off *and* put them away in your coat / backpack.
- Backpacks remain **on the ground**, not on your desk.
- Attempts at **fraud**, or actual acts of fraud, will be punished by the Board of Examiners.

## About the exam

- Your project (sources *and* tests) **must compile** to get a passing grade. Sometimes IntelliJ will allow you to run part of your code even when some classes fail to compile. We will still see this as a compilation failure. Ensure that **ALL classes you hand in compile.** If your solution does not compile, you will not get any points.
- **Follow the submission instructions in the last part of the exam carefully**.
- **Stop writing code a few minutes before the end** of the exam so you have time to make sure your submission compiles and to hand in the exam.
- The **grading** is explained on the last page of this exam.

## Plant Determination

Eden recently moved into an apartment with a delightful garden full of plants and flowers. However, they don't know all that much about botany, and would like to find out which plants and flowers grow in the garden. Eden asks you, someone with programming experience, to help create a simple app that can help with the determination of plants and flowers they could use. The idea is that once you start the app you can query a plant database and narrow down possible matches based on plant characteristics, such as colour or type of flower.

Eden did some work up front and provides you with a database of different plants and their characteristics. They ask you to develop an application that can read and process the following information:

```
HERB: White mustard; Sinapis alba; (20, 70)
FLOWER DETAILS: yellow; small; 4 petals
EDIBILITY: Yes; strong, pungent, somewhat bitter and slightly sweet
HERB: Basil;  Ocimum basilicum; (30, 150)
FLOWER DETAILS: white; small; 5 petals
EDIBILITY: Yes; savory and sweet with mint and pepper notes
SHRUB: Sweetbriar Rose; Rosa rubiginosa; (200, 300)
FLOWER DETAILS: pink; medium; 5 petals
TREE: Pine; Pinus; (300, 8000)
TREE: Oshima cherry; Prunus speciosa; (400, 1200)
FLOWER DETAILS: white; medium; 5 petals
HERB: Bracken fern; Pteridium aquilinum; (30, 100)
EDIBILITY: No; mixture of asparagus, almonds and kale
SHRUB: King sago; Cycas revoluta; (50, 100)
```

The order of the lines for the a single plant is fixed. However, **herbs**, **shrubs** & **trees** may appear in any order in the input file. For each plant you will first get the name, then the Latin name, followed by a range indicating the typical size range for the plant.
Then, *if* the plant flowers, on the next line you will find information about the flowers of the plant. First the color, then the size, followed by the number of petals the flowers have.
For **herbs** only there is an additional line with information about whether the herb is safe to eat or not and the taste / fragrance notes for that herb.

The file **plants.txt** is available in the template project.

Design and implement a program that:

- **Reads in** the file plants.txt and has classes and structures to store and represent the information in the file.
- To enable user interaction, please provide a **command line interface** (reading from `System.in` and writing to `System.out`).

An **Herb** is characterised by:

- A name
- A Latin name
- A range indicating typical plant size (in cm)
- The colour of the flowers (o*ptional, not all herbs have flowers!)*
- The size of the flowers (o*ptional, not all herbs have flowers!)*
- The number of petals of the flowers (o*ptional, not all herbs have flowers!)*
- An indication of whether the herb is safe to eat.
- Fragrance/taste notes

A **Shrub** is characterised by:

- A name
- A Latin name
- A range indicating typical plant size (in cm)
- The colour of the flowers (o*ptional, not all shrubs have flowers!)*
- The size of the flowers (o*ptional, not all shrubs have flowers!)*
- The number of petals of the flowers (o*ptional, not all shrubs have flowers!)*

A **Tree** is characterised by:

- A name
- A Latin name
- A range indicating typical plant size (in cm)
- The colour of the flowers (o*ptional, not all trees have flowers!)*
- The size of the flowers (o*ptional, not all trees have flowers!)*
- The number of petals of the flowers (o*ptional, not all trees have flowers!)*

## Menu and options that should be implemented:

```
Please make your choice:
1 – Show plants that meet all criteria.
2 – Filter plants by size.
3 – Filter plants by flower colour.
4 - Filter plants by edibility.
5 – Reset all criteria.
6 – Simulate global warming.
7 – Quit the application.
```

**Option 1:**

Show all the plants that meet all the restrictions set by options 2, 3 and 4. Please note that these restrictions can stack on top of each other.
If no restrictions have been set print all plants that have been read from the input file. Make sure this output is in a nicely readable format. Ensure that you also print a clear message in case there are no plants that meet the criteria / there are no plants.

*The output could for instance look like (example has been shortened):*

```
Herb named Basil (latin: Ocimum basilicum), typical size between 30cm and 150cm.
This plant has small, white flowers with 5 petals.
This herb is safe to eat. Savory and sweet with mint and pepper notes.

Tree named Pine (latin: Pinus), typical size between 300cm and 8000cm.
This plant does not have flowers.
```

**Option 2:**

This option should restrict the plants that are printed for option 1 based on a size provided by the user. If the user inputs 150, all plants that have 150 within their typical size range should be printed (the ranges for the plants that have been provided in the file are *inclusive*).

**Option 3:**

This option should restrict the plants that are printed for option 1 based on flower colour. For this option you should show the user all colours that are known **without duplicates** (and have not been filtered out yet) and let them choose a colour from the options. If there are no colours to choose from you should indicate this.

**Option 4:**

This option should restrict the plants that are printed for option 1 based on whether they are edible or not. The user should be able to provide input to indicate whether the results should be filtered on herbs that *are* edible or herbs that *are not* edible. Since only herbs have this property, any plants that are not herbs should be filtered out in both cases.

**Option 5:**

This option should reset any restrictions that have been set through options 2, 3 and 4 and restore the original list of plants. Since reading from the file is an expensive operation, this option should not read from the file again.

**Option 6:**

This option should simulate the fact that the plants in the garden will grow taller if it gets warmer on the earth. Each time this option is used, both the minimum and the maximum in the size range of **all plants** should be increased by 10% (and then rounded down to the nearest int). So, in the case of Basil, the range will change from (30, 150) to (33, 165) after calling option 6 once. Option 6 can be called multiple times to stack the growth.

For full points you **must** implement this feature using the `replaceAll` method which is available for `List`

*You may reset any restrictions applied in options 2, 3 and 4 when going through this process. If option 1 is selected after option 6, it should show all plants with their increased size ranges.*

**Option 7:**

The application stops.

## - Some important things to consider for this assignment:

- The program should **compile .**
- For a good grade, your program should also work well, without exceptions.
- Take care to have a nice programming style, in other words make use of code indentation, whitespaces, logical identifier names, etc. You can check this with **checkstyle**!
- You should provide Javadoc comments. Proper Javadoc comments include a clarification for each parameter as well as a short description of the method.
- Your program should have an **equals() & hashCode()** method for each class that is used in a non-static way.
- Think about your design: consider using *composition*, *inheritance*, *interfaces*, *enums*, *records*, etc. The textual information should be read into a class structure containing the right attributes to store the data (so storing all information in a single `String` is not allowed, because this would hinder further development).
- Ensure your program is properly **unit tested**. Your final project should have at least **80% line coverage overall, *and* at least the same number of test method as non-trivial methods.** Constructors, getters, and setters are considered trivial methods. *Any method that directly interacts with a file or user input (`System.in`) do not have to be tested.*
- The **filename plants.txt should not be hardcoded** in your Java program. Please make sure to let the user provide it when starting the program (either as an explicit question to the user or as a program argument).

## Handing in your work

To hand in your work you must create a **ZIP** file containing your project files. To do so, please go to the desktop and click the **“Exam Uploader”**. There you will have to fill in your student number (e.g. 5678901) and click “Upload Exam Data”. When this process has completed successfully you can verify whether a zip file with your exam has appeared on the **P-drive**.

---

## Good Luck!

## Grade composition

*1.4 points* **Compilation**

- If your solution does not compile your final score = 1.

*1.5 points* **Class Design**

- Proper use of OOP principles. Additionally, there should be a good division of logic between classes & interfaces.

*0.5 points* **equals() & hashCode() implementation**

- Correct implementation of equals() & hashCode() in all classes that are part of your data model (any records are exempt).

*1.6 points* **File reading**

- Being able to read user-specified files, and parsing the information into Objects. *A partially functioning reader may still give some points.*

*1.5 points* **Code style**

- Ensure you have code that is readable. This includes (among others) clear naming, use of whitespaces, length and complexity of methods, Javadoc, etc.

*0.5 points* **User Interface**

- Having a well-working (looping) textual interface (including option 1 which prints the plants in the console). *A partially functioning interface may still give some points.*

*1.2 points* **JUnit tests**

For full points, your project should have at least **80% line coverage overall, *and* at least the same number of test method as non-trivial methods.** Constructors, getters, and setters are considered trivial methods. *Any method that directly interacts with a file or user input (`System.in`) does not have to be tested.* You are expected to test the filtering functionality in your program for full points, so take this into account in your design.

*1.8 points* **Managing plants in the catalogue**

- *0.3 points* for being able to filter plants on a size.
- *0.5 points* for being able to filter plants on the flower colour.
- *0.3 points* for being able to filter plants on whether they are edible or not.
- *0.3 points* for being able to reset all filters and restore the original catalogue of plants.
- *0.4 points* for managing the growth of the plants due to global warming properly.

**There is a 0.5 point penalty if you hardcode the filename.**
**There is a 0.5 point penalty if you do not hand in a proper (zip) archive.**