Assignment 2
Object-oriented program development

Read Section 1 to understand the requirements, Section 2 to understand the programming tasks that you need to carry out, and Section 3 to know the assessment details and submission requirements.

1. Description

In this assignment, you will develop an object-oriented program in Java that obtains user data input from the standard input to create objects and display a report to the standard output. The program also provides the option to store the report in a text file.

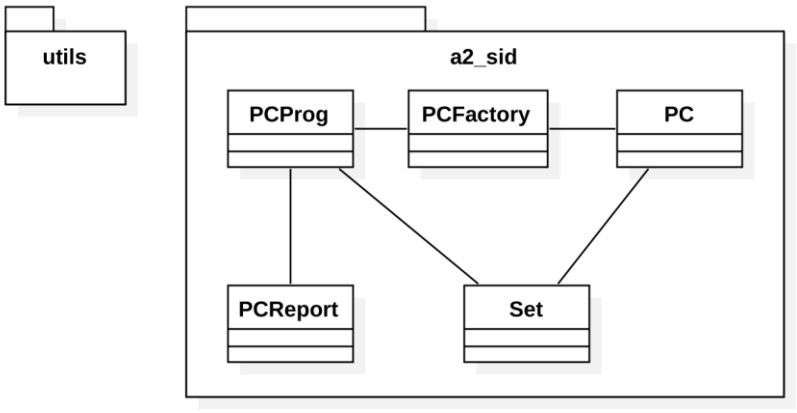

Figure 1: Program structure.

The program, named `PCProg`, processes data about personal computers. A personal computer (PC) is described in terms of the following attributes: `model`, `year`, `manufacturer`, and `comps` (short for components). The table below describes the domain constraints of these attributes. Note that attribute `comps` has the type `Set`. For this program, components are simply strings that denote their names.

Attributes	type	mutable	optional	length	min	max
model	String	T	F	20	-	-
year	Integer	F	F	-	1984	-
manufacturer	String	F	F	15	-	-
comps	Set<String>	T	F	-	-	-

For example, a Lenovo Thinkpad X1 Carbon PC (model 'Thinkpad X1 Carbon'), which has the following components: AMD Ryzen 5, 8GB DDR4, 512GB SSD, and NVIDIA MX450 are represented by the tuple:

```
PC<Thinkpad X1 Carbon, 2022, Lenovo, Set {AMD Ryzen 5, 8GB DDR4, 512GB SSD, NVIDIA MX450}>
```

To create a PC object, the program first prompts the user for the required data values in the following order: `model`, `year`, `manufacturer`, and `components`. The number of components may vary from 0 to many. When the user wants to stop adding components, (s)he enters a blank line. Please note that `comps` is a set that doesn't store duplicate elements.

Once the data values have been obtained, the program creates a new PC object and adds it to a set. This set is an object of the Set class. The program then asks the user if (s)he wishes to continue (the answer to which must be either "**Y**" or "**N**"). If the user answers "**Y**", the program repeats to create the next PC object. If, however, the user responds with "**N**", the program proceeds to display a report about

the PC objects.

To display a report, the program first generates it and then presents the result on a table that looks like the one shown in **Listing 1** below. The report title is displayed in the middle of the top banner. All but the first column corresponds to the PC attributes; the rows are data about the PC objects. Thus, the second column corresponds to the attribute `model`, the third corresponds to the `year`, and the fourth corresponds to the `manufacturer`. The last column lists the string representations of the components of the PC objects. The first column sequentially displays the row numbers. Note that the widths of the second and fourth columns are the lengths of the corresponding attributes. The widths of the first and third columns are 3 and 6, respectively. The fifth column is unrestricted in width. The cell values are properly aligned with the columns and are displayed right-justified, except for the fifth column (to display components), which is left-aligned. To help you imagine, the widths of columns 1 to 4 are marked using a blue background in **Listing 1**. The horizontal lines (as displayed by lines of hyphens in the report) are exactly 99 characters in width. The cell values need not be wrapped. Further, the boundary between two adjacent cells on the same row is exactly one space (␣) apart.

Listing 1: A tabular report for PCs

```
---------------------------------------------------------------------------------------------------
                                        PCPROG REPORT
---------------------------------------------------------------------------------------------------
  1   Thinkpad X1 Carbon   2022          Lenovo [AMD Ryzen 5, 8GB DDR4, 512GB SSD, NVIDIA MX450]
  2   Thinkpad X1 Carbon   2022         Toshiba [AMD Ryzen 5, 16GB DDR4, 512GB SSD, 512GB HDD, ...]
---------------------------------------------------------------------------------------------------
```

Immediately below the report is a prompt for whether the user wishes to **save the report** to a text file. If the user answers "Y", then the program saves the report text to a file named `pcs.txt`, which is located in the same directory as the program's. Otherwise, the program ends.

2. Task requirements

Complete the following tasks:

1. Create a package named `a2_sid`, as shown in **Figure 1**, where `sid` is your student ID. For example, if your student ID is 123456789, then the package name is `a2_123456789`.

You will need to use this package to store all the Java class(es) that you create for the program.

Copy to this package two classes (`PCProg` and Set) that are provided in the attached zip file. Fix the import statements in these classes to match your package name. The subsequent tasks will explain what you need to do with these classes.

IMPORTANT:

a) Failure to name the package as described above will result in an invalid program.
b) You must use the necessary utility classes in the `utils` package. This package should be created as another top-level package, as shown in **Figure 1**. You must not create package `utils` as a sub-package of your package. In addition, you must not include package utils as part of your submission.
c) You must use JDK version 8.

2. Specify and implement class `PC`.

   Note:

(a) `PC` must contain the essential state and behaviour spaces.

(b) `PC` must appropriately use `Set` in their design.

(c) You must implement `PC.toString()` such that the outputs look like the example shown in Section 1.

(d) You must override the `equals()` method for `PC`, which determines the equality of two `PC` objects based on their states.

3. Specify and implement class `PCFactory` that has a factory method for creating PCs. This class must also be a singleton. You must strictly follow the relevant design pattern solutions.

4. Specify and implement class `PCReport` which contains a single operation:

```
displayReport(PC[] objs): String
```

Note the following:

a) The report format must be as specified in **Listing 1**.
b) This class has no instance variables.

5. A partially completed class named `PCProg` is provided for testing your program. Move it into your `a2_sid` package so that you can run it. Class `PCProg` contains the `main` method and some operations for performing the tasks highlighted in Section 1:

a) Attribute `objs` is typed `Set<PC>` and used to record PC objects.
b) `createObjects`: uses `PCFactory` to create a new `PC` object and record it in `objs`. Method `createObjects` should not invoke the `PC` constructor directly.
c) `getObjects`: return the recorded PC objects.
d) `displayReport`: uses `PCReport` to generate and display the tabular report about PC objects.
e) `saveReport`: save the report text to a file.
f) `main`: create a new `PCProg` object and run its operations.

Note:

- The following procedures are completed and provided for you: `displayReport`, `saveReport`, and `main`. You must not change them.
- In order to make the program work, you need to specify and implement the operations `createObjects` and `getObjects`.

3. Submission requirements

Create a zip file containing **just the folder for the required package**. You must name the file as follows: `a2_sid.zip`, where `sid` is your student ID. Submit your file to the designated submission box.

IMPORTANT: Failure to name the file as described above will result in an invalid program. In particular, ONLY the **ZIP** format is accepted. Other formats (e.g., RAR) are NOT accepted.