Computer Science 2A

Practical Assignment Bonus 1

Assignment date:	2024-05-18
Deadline	2024-05-24 13h00
Marks: 100	

This practical assignment must be uploaded to eve.uj.ac.za **before** 2024-05-24 13h00. Late[1] or incorrect submissions **will not be accepted**, and will therefore not be marked. You are **not allowed to collaborate** with any other student.

Good coding practices include a proper coding convention and a good use of documentation. Marks will be deducted if these are not present. Every submission **must** include a batch file unless stated otherwise.

The **reminder page** includes details for submission. Please ensure that **ALL** submissions follow the guidelines. The reminder page can be found on the last page of this practical.

This practical aims to familiarise you with the ***Abstract Factory Design Pattern***.

The **Utopian Artificial Intelligence Forum**[2] is hosting their annual simulation challenge. You have been tasked by your team to create factories to be used in the project that your team is building.

Since you are **only dealing with factories**, your implementation will **not** include a complex GUI or work with files.

Your team has already done some boilerplate code to ensure that the whole team is on the same page when developing the simulation. You have been provided with the `SIM.jar` file and accompanying documentation in the `provided_javadoc` folder that you need to familiarize yourself with.

In the `JAR` file, your team has provided you with two enumerations:

- `EOperatorType`
  - MANNED
  - UNMANNED
- `EVehicleType`
  - AERIAL
  - TERRESTRIAL
  - AQUATIC

[1] Alternate arrangements for exceptional circumstances will been posted on eve.

[2] Disclaimer - This series of problem statements are a work of fiction. Names, characters, businesses, places, events, and incidents are either the products of the author's imagination or used in a fictitious manner. Any resemblance to actual persons, living or dead, or actual events is purely coincidental.

They have also provided you with all of the **Abstract Factory Design Pattern** interfaces that you must use. The interfaces are detailed in the provided javadoc. Your classes must therefore be compatible with all of these:

- Abstract Product Interfaces: `IAerialVehicle`, `ITerrestrialVehicle`, and `IAquaticVehicle`
- Abstract Factory Interface: `IOperatorFactory`

Your team lead has also provided you with a `OperatorPicker` class in the `JAR` file which will be used in the simulation to select the operator that will be used. The `OperatorPicker` is responsible for returning an instance of the correct factory to be used to create vehicles with operators.

You need to create a `Java` application with the following:

- Create classes for each operator and vehicle type that will implement the product interfaces (e.g., `MannedAerialVehicle`, `UnmannedAquaticVehicle`, etc.[3]).
  - The implementation of each vehicle method will simply print out the movement type and operator[4]. The output should follow this structure: "`<EOperatorType> <EVehicleType> is <Movement Type>.`". For example, the Manned Terrestrial Vehicle would output in the console on a newline: `MANNED Terrestrial Vehicle is driving`.
  - Your model classes must be placed in the `acsse.csc2a.sim.model` package.
- Create Concrete Factory classes for each operator type that will implement the `IOperatorFactory` interface.[5]
  - Each factory must make the correct vehicle type (e.g., `MannedFactory` makes `MannedAerialVehicle`, `MannedAquaticVehicle`, etc.).
  - Your Concrete Factory classes must be placed in the `acsse.csc2a.sim.concrete` package.
- `Main`: Test that your classes work by using the provided `Main`.[6]
  - You have been instructed to **NOT make any changes to the** `Main` because your classes must be compatible with the rest of your team's work.[7]

Marksheet

1. Correct Implementation of the Factory Design Pattern (Note this is an all or nothing practical.`[100]` If you do not provide the correct output, you will not get the marks.)

NB

Submissions which **do not compile** will be capped at 40%!

Practical marks are awarded subject to the student's ability to explain the concepts and decisions

[3] There are 6 product classes in total.

[4] See the notes in the provided_javadoc to see what each method should print to the standard output stream.

[5] Your class names must be compatible with the `OperatorPicker` class.

[6] By examining the imports and code in the Main, you should be able to understand more about the expected usage of the program.

[7] We may replace your Main, so your classes should all be implemented (not just the ones used in the Main class).

made in preparing the practical assignment solution. (Inability to explain code = inability to be given marks.)

Execution marks are awarded for a correctly functioning application and not for having related code.

Reminder

Your submission must follow the naming convention below.

`SURNAME_INITIALS_STUDENTNUMBER_SUBJECTCODE_YEAR_PRACTICALNUMBER`

Example

Surname	Berners-Lee	**Module Code**	CSC02A2
Initials	TJ	**Current Year**	2024
Student number	209912345	**Practical number**	PBonus 1

`Berners-Lee_TJ_209912345_ CSC02A2_2024_PBonus 1`

Your submission must include the following folders:

Folder	State	Purpose
`bin`	*Required*	Should be empty at submission but will contain runnable binaries when your submission is compiled.
`docs`	*Required*	Contains the batch file to compile your solution, UML diagrams, and any additional documentation files. All files must be in **PDF** format. Your details must be included at the top of any **PDF** files submitted. **Do not include generated JavaDoc.**
`src`	*Required*	Contains all relevant source code. Source code must be places in relevant sub-packages! Your details must be included at the top of the source code.
`data`	*Optional*	Contains all data files needed to run your solution.
`lib`	*Optional*	Contains all libraries needed to compile and run your solution.

NB

Every submission **must** include a batch file that contains commands which will:

- Compile your Java application source code.
- Compile the associated application JavaDoc.
- Run the application.

Do not include generated JavaDoc in your submission. All of the classes/methods which were created/updated need to have JavaDoc comments.

Multiple uploads

Note that only **one** submission is marked. If you already have submitted once and want to upload a newer version then submit a newer file with the same name as the uploaded file in order to overwrite it.