Project 1 - Factory

CmpE 250, Data Structures and Algorithms, Fall 2022

Instructor: Özlem Durmaz İncel
TAs: Suzan Ece Ada, Barış Yamansavaşçılar
SAs: Batuhan Çelik, Bahadır Gezer, Zeynep Buse Aydın

Due: 21/10/2022, 23:55 Sharp

1 Introduction

In *Rumeli Hisarüstü*, a factory called *1-A Factory* produces essential products. Every product has their respective value. In the factory, individual units called holders are responsible for handling the products in the factory line. Each holder is coupled with the previous and the next holder to create a product line. You are tasked with the organization of this product line.

2 Details

You will be given two classes `Holder`, and `Product`; and an interface, `Factory`. You need to write two new classes `FactoryImpl`, and `Project1`.

2.1 `FactoryImpl`

- This class should implement the `Factory` interface. The overridden methods should work the way they are described in the `Javadoc`.
- This class has three mandatory fields. These fields should be correctly modified inside the overriden methods.

```
private Holder first
private Holder last
private Integer size
```

- You are free to add helper methods.

2.2 `Project1`

- The main method should be implemented here. This class is the entry point of your program.
- You are required to read from the input file and write to the output file. These file paths will be given in the program arguments.
- Each input command will be given in a single line. You have to read the input file and write to the output file line by line.

3 Input & Output

3.1 Input

- **Add First -** Adds a new product at the beginning of the factory line.

AF	product_{id}	product_{value}

- **Add Last -** Adds a new product to the end of the factory line.

AL	product_{id}	product_{value}

- **Add -** Adds a new product to the given `index` of the factory line. Prints `"Index out of bounds."` if the index is out of bounds.

A	index	product_{id}	product_{value}

- **Remove First -** Removes the first product from the factory line and prints it. Prints `"Factory is empty."` if the there is no product in the factory.

RF

- **Remove Last -** Removes the last product from the factory line and prints it. Prints `"Factory is empty."` if the there is no product in the factory.

RL

- **Remove Index -** Removes the product in the given `index` and prints it. Prints `"Index out of bounds."` if the index is out of bounds.

RI	index

- **Remove Product -** Removes the first occurrence of the product with the given `value` and prints it. Prints `"Product not found."` if the product is not in the factory line.

RI	product_{value}

- **Find -** Prints the product with the given $\texttt{product}_{id}$. Prints `"Product not found."` if the product is not in the factory line.

F	product_{id}

- **Get -** Prints the product in the given `index`. Prints `"Index out of bounds."` if the index is out of bounds.

G	index

- **Update -** Updates the `value` of the product with the given $\texttt{product}_{id}$ to $\texttt{product}_{value}$. Prints `"Product not found."` if the product is not in the factory line.

U	product_{id}	product_{value}

- **Filter Duplicates -** Removes products such that after the filtering process there is only a single occurrence of each product in the factory line. In the context of this method, duplicate products are products with equal `value` fields, the `id` fields are of course unique.

FD

- **Reverse -** Reverses the factory line.

R

- **Print -** Prints the factory line.

P

Input file path will be given as the first program argument.

3.2 Output

- **Product -** Products will be printed in the format below.

(product_{id}, product_{value})

- **Factory Line -** Factory line will be printed in the format below.

{ product_1, product_2, ... , product_n }

Output file path will be given as the second program argument.

4 Submission

Your project will be graded automatically. So it's important that you carefully follow the submission instructions. First, all 5 of your source files, and nothing more, should be collected under `Project1/src`. Then, you should zip the `Project1` folder and rename it to `p1_<student`$_{id}$`>.zip` (e.g., `p1_2020400999.zip`). This zip file will be submitted through moodle.

Your program must be runnable through the terminal. We will compile your code with the `javac` command. The target version is 17. Hence, it is imperative that your project structure is correct, otherwise you will not be able to get any points from the automatic grading system. The compiled program will be run using `java` command. Your program should be able to run with full-path arguments.

5 Grading

Grading of this project is based on the automatic compilation and run and the success of your code in test cases. If your code compiles and runs with no error, then you will get 10% of the project grade. 40% of the grade will come from unit tests. We will test each method implementation for the methods in the `Factory` interface. Each method implementation will have equal weight. The rest of your grade will be the sum of collected points from each test case. Each test case will have equal weight. Maximum project grade is 100%.

Input	Corresponding Output Line	Explanation
P	{}	Initially the factory line is empty.
RF	Factory is empty.	Cannot remove from the front since the factory is empty.
RL	Factory is empty.	Same as above, but now it's the end.
AF 3 9		`(3,9)` is added to the front.
A 0 2 4		`(2,4)` is inserted to `index` 0.
AL 5 25		`(5,25)` is added to the end.
A 0 1 1		`(1,1)` is inserted to `index` 0.
A 3 4 16		`(4,16)` is inserted to `index` 3.
AL 6 36		(6,36) is added to the end.
AF 7 30		`(7,30)` is added to the front.
P	{(7, 30),(1, 1),(2, 4),(3, 9), (4, 16),(5, 25),(6, 36)}	Factory line is printed.
F 3	(3, 9)	Product with `id` 3 is found and printed.
F 20	Product not found.	Product with `id` 20 does not exist.
U 4 9	(4, 16)	`value` of the product with `id` 4 is updated to 9. The previous product is printed.
U 13 21	Product not found.	Product with `id` 13 does not exist.
G 0	(7, 30)	Product with `index` 0 is printed.
G 17	Index out of bounds.	The factory line has 7 products so `index` 17 is out of bounds.
U 1 36	(1, 1)	`value` of the product with `id` 1 is updated to 36. The previous product is printed.
A 21 21 21	Index out of bounds.	There is no product at `index` 21.
FD	2	All duplicates are removed from the factory line. In this case it was `(4,9)` and `(6,36)`. Prints the number of duplicates removed.
P	{(7, 30),(1, 36),(2, 4), (3, 9),(5, 25)}	Prints the factory line.
R	{(5, 25),(3, 9),(2, 4), (1, 36),(7, 30)}	Reverses the factory line. Prints the new version.
RP 9	(3, 9)	Removes the first product with `value` 9.
RP 13	Product not found.	No product has `value` 13.
RL	(7, 30)	Removes and prints the last product.
RI 2	(1, 36)	Removes and prints the product at `index` 2.
RF	(5, 25)	Removes and prints the first product.
RI 4	Index out of bounds.	The factory line has a single product so `index` 4 is out of bounds.

Table 1: Line by line analysis of example input

6 Warnings

- All source codes are checked automatically for similarity with other submissions and exercises from previous years. Make sure you write and submit your own code. Any sign of cheating will be penalized by at least -50 points at first attempt and F grade in case of recurrence.
- Make sure you document your code with necessary inline comments and use meaningful variable names. Do not over-comment, or make your variable names unnecessarily long. This is very important for

partial grading.

- Do not make changes in `Holder.java`, `Product.java`, and `Factory.java`.
- Do not add any files to the source code except `Holder.java`, `Product.java`, `Factory.java`, `FactoryImpl.java`, and `Project1.java`.
- Make sure that each method in the `Factory` interface does the operations it is supposed to do. These methods will be individually unit tested. For example, if there is only one product in the factory and either `RF` or `RL` is called, then both `first` and `last` fields should be set to `null`. So basically the three mandatory fields should always be correctly set.
- Make sure that the white-spaces in your output is correct. You can disregard the ones at the end of the line.