ASSIGNMENT 4

Submission Date : 07.05.2021
Due Date : 28.05.2021 (23:59)
Programming Languages: Java
Subject : Stack, Priority Queue

1 Introduction

In this assignment, you are going to simulate a vending machine scenario. According to this scenario, you are given several vending parts such as biscuits, chocolates, drinks and so on. In each part of vending machine, there are vending items. You have also token box for store your tokens. There are also different tokens for each part of vending machine.

2 Background

2.1 Stack

A stack is an Abstract Data Type (ADT), commonly used in most programming languages. It is named stack as it behaves like a real-world stack, for example – a deck of cards or a pile of plates, etc. A real-world stack allows operations at one end only. For example, we can place or remove a card or plate from the top of the stack only. Likewise, Stack ADT allows all data operations at one end only. At any given time, we can only access the top element of a stack.

This feature makes it LIFO data structure. LIFO stands for Last-in-first-out. Here, the element which is placed (inserted or added) last, is accessed first. In stack terminology, insertion operation is called PUSH operation and removal operation is called POP operation. The following figure (Figure 1) depicts a stack and its operations.

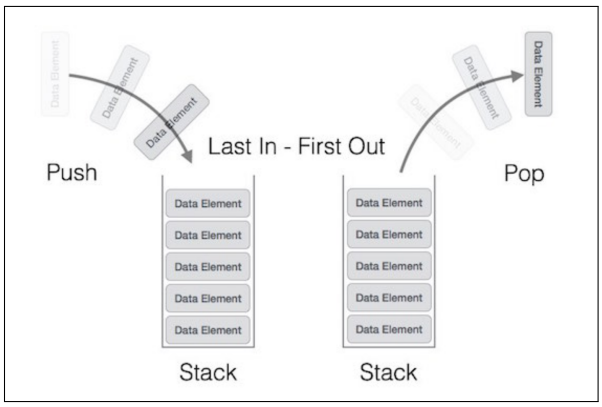

Figure 1: Stack and it's operation

2.2 Queue

Queue is an abstract data structure, somewhat similar to Stacks. Unlike stacks, a queue is open at both its ends. One end is always used to insert data (enqueue) and the other is used to remove data (dequeue). Queue follows First-In-First-Out methodology, i.e., the data item stored first will be accessed first. The following figure (Figure 2) depicts a queue and its operations.

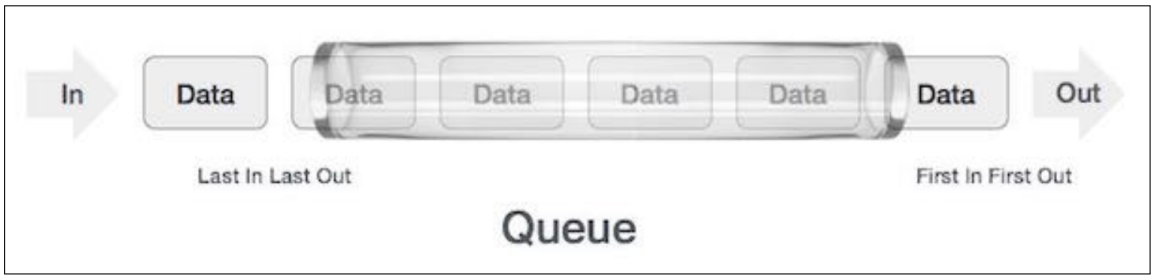

Figure 2: Queue representation

Basic operations are as following:

- **insert / enqueue:**This function defines the operation for adding an element into queue.
- **remove / dequeue:** This function defines the operation for removing an element from queue.

In computer science, a **priority queue** is an abstract data type similar to a regular queue or stack data structure in which each element additionally has a "priority" associated with it. In a priority queue, an element with high priority is served before an element with low priority.

3 Problem Definition

3.1 Scenario

You are expected to complete tasks that are given tasks.txt according to input files and produce an output file. The informations about parts, items, tokens and tasks are in the input files. In parts.txt file, you are given several parts of vending machine. These parts are modeled as stacks that are following the last-in-first-out policy. And each part has items given in items.txt. Each item should be in corresponding part of vending machine. You have also token box for store your tokens which are given in tokens.txt. This box is modeled as priority queue. In a priority queue, an item with high priority is served before an element with low priority. In this assignment, the higher value token has more priority. If two tokens have the same value, they should serve according to the order in which they were enqueued. In our scenario, you have two different tasks that are BUY and PUT. These will be given in tasks.txt. There may be more than one of each in tasks.txt.

3.2 Experiment

Your system will take the input files as parameters that are parts.txt, items.txt, tokens.txt and tasks.txt. The initial status of the parts of the vending machine can be obtain by reading the input from these files. The status of each part should be written into the output file after the tasks are completed.

3.3 Inputs

There will be four input files that your program will take as arguments. These are *parts.txt*, *items.txt*, *tokens* and *tasks.txt* respectively. You are expected to create a program that reads the inputs about vending machine and after completing tasks given in tasks.txt, produces an output file exhibiting the final status of the each parts of vending machine. Details of the files are given below.

parts.txt	items.txt	tokens.txt	tasks.txt
Biscuits Chocolates Drinks Crisps	I1 Biscuits I2 Biscuits I3 Chocolates I4 Biscuits I5 Chocolates I6 Crisps I7 Drinks I8 Chocolates I9 Biscuits I10 Crisps I11 Chocolates I12 Drinks	T1 Biscuits 2 T2 Chocolates 3 T3 Chocolates 2 T4 Drinks 3 T5 Biscuits 4 T6 Crisps 2 T7 Drinks 5	BUY Biscuits,3 Chocolates,2 Crips,1 PUT Biscuits,I13,I14,I15 Chocolates,I16,17 Drinks,I18,I19

3.3.1 Parts

- In each part you are given, items are kept.
- Vending machine parts are modeled as stacks that are following the last-in-first-out policy.

3.3.2 Items

- The initial status of items are given in items.txt according to their corresponding part.
- Each item can be put to the relevant part in the vending machine or bought from the corresponding part.
- Stack policy is applied when putting or buying items.

3.3.3 Tokens

- Each token can be used for a relevant items. For example you can only use biscuit tokens to buy biscuits.
- Each token value corresponds to 1 relevant item. For example, you can take only 2 biscuits by using T1 token.
- The list of tokens in the tokens.txt file may be given to you in mixed order.
- Your token box is modeled as priority queues. When asked for a token, you should use the highest value inside the token box.
- The initial state of token box is given tokens.txt

3.3.4 Tasks

- Your tasks are given in tasks.txt
- You can both buy items from the vending machine and put items into the vending machine. Which action you will do is given in the tasks.txt file.

- A tasks.txt has the following format:

```
<BUY><TAB><PartOfVendingMachine,NumberOfItems><TAB>...<TAB>
<PartOfVendingMachine,NumberOfItems>
<PUT><TAB><PartOfVendingMachine,ItemId,ItemId,...,ItemId><TAB>....<TAB>
<PartOfVendingMachine, ItemId,ItemId,...,ItemId >
```

- During a task you need to pay attention to the order of putting items to vending machine and taking items from vending machine.

3.4 Output

You are expected to produce a output. In this output, status of each part of vending machine should be printed. The format of this file is as follows:

```
<Name of part>
<Item0>
<Item1>
.....
<ItemN>
---------------
<Name of part>
<Item0>
<Item1>
.....
<ItemN>
---------------
<Name of part>
<Item0>
<Item1>
.....
<ItemN>
-----------------
.....
.....
-----------------
Token Box:
<Token0>
<Token1>
.....
<TokenN>
```

3.5 Example Run

- Assume that you are given the files in Section 3.3. Reading the inputs from these files, the initial configuration of the vending machine and your token box as shown in Figure 3.
- As seen in the Figure 3, the items read from the items.txt file are put to the corresponding part in order of reading by using stack principle.
- Tokens read from the tokens.txt file are put to token box according to priority queue. As shown in Figure 3, T1 and T3 have same value. Since T1 is earlier than T3 in the reading order, its priority is high.

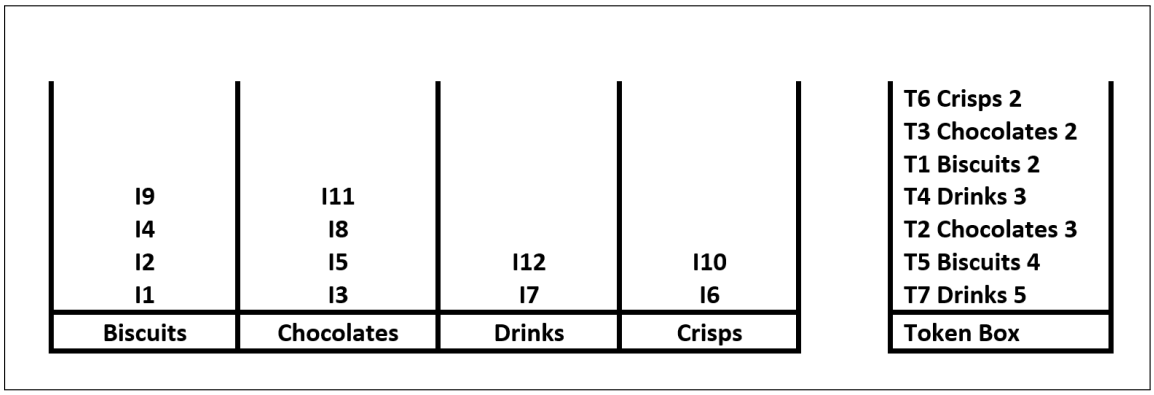

Figure 3: The initial configuration of the parts and token box

- According to task.txt file, you should buy 3 biscuits, 2 chocolates and 1 crisps from the vending machine for the first task. As shown in Figure 3 and Figure 5, putting items to corresponding part and taking items from the part are done according to the stack principle.
- When taking any item from the vending machine, get the corresponding token with the highest priority from the token box. If value of the token you get is greater than or equal to the value given in the task, update the value of the token and insert it back to the queue. If it is small, remove the token from the queue and take the corresponding token from the queue for the remaining value. Repeat this until you provide the value given in the task file. The sum of the corresponding token values of each part will be given higher than the value given in the task file.

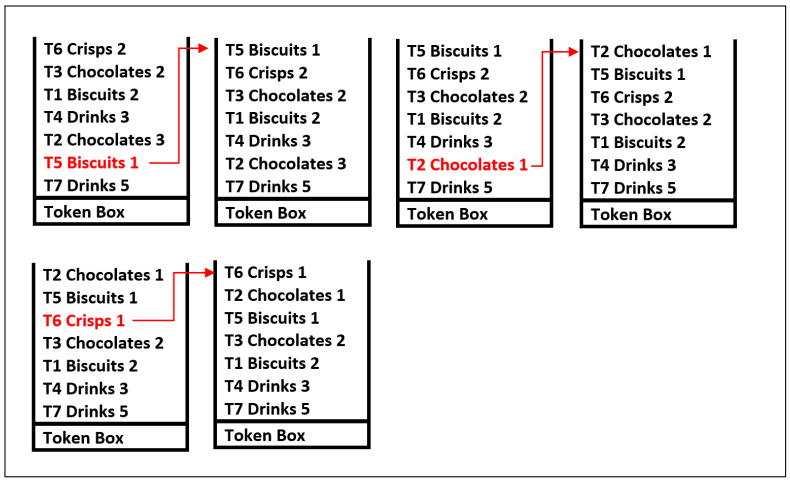

Figure 4: When reading the first task, the status of the token box step by step

- After reading the first task, the status of vending machine as in Figure 5.

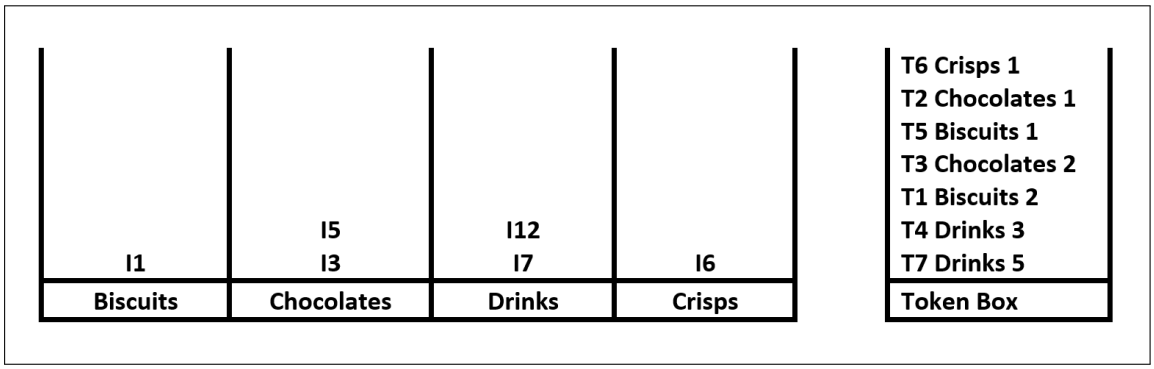

Figure 5: After reading the first task, status of the parts and token box

- For the second task, you should put I13, I14 and I15 into biscuits part of vending machine, put I16 and I17 into the chocolates part of vending machine and lastly put I18 and I19 into the drinks part of vending machine. After reading this task, the status of vending machine as in Figure 6:

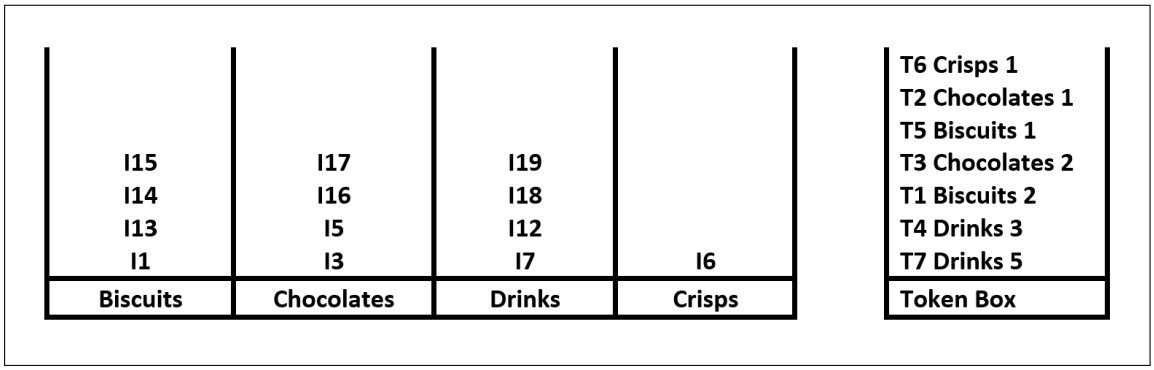

Figure 6: After reading the first task, status of the parts and token box

- After the tasks are completed, the output is shown in Figure 7. Please pay attention to the order of your items when printing the status of each part.

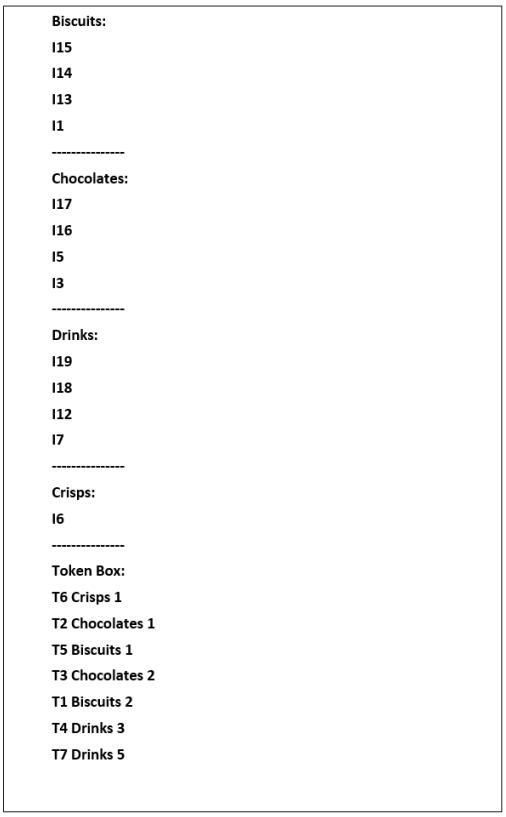

Figure 7: Status of each parts and token box after the tasks are completed

4 Grading and Evaluation

- Your work will be graded over a maximum of 100 points.
- Your total score will be partial according to the grading policy stated below.

Compiled	10p
Taking input files as arguments from command line and reading the file contents	10p
Correct reporting the status of token box after tasks are completed	30p
Correct reporting the status of each part of vending machine after tasks are completed	35p
Code design, clean and readable code, algorithmic perspective, comments	15p

- Your code will be tested with different inputs. And one output file will be expected as output file.
- The assignment must be original, individual work. Downloaded or modified source codes will be considered as cheating.
- We will be using the Measure of Software Similarity (MOSS) to identify cases of possible plagiarism.
- The following usage example shows how your program will be run from the command line.

Usage example:
> javac *.java
>java Main parts.txt items.txt tokens.txt tasks.txt output.txt

Notes

- Do not miss the deadline. Submission will be end at 07/05/2021 23:59, the system will be open until 23:59:59. The problem about submission after 23:59 will not be considered.
- Save all your work until the assignment is graded.
- The assignment must be original, individual work. Duplicate or very similar assignments are both going to be considered as cheating.
- Try to put as small code as possible inside the main method. Create other methods or classes, preferably, and call them in your main method. Main method should be quite comprehensible.
- Write READABLE SOURCE CODE block. Don't forget to write comments of your codes when necessary.
- **For this assignment, you must use your own implementation of stack and queue. In other words, you cannot use any existing stack and queue codes from other sources such as the stack class in the java collections.**
- The names of classes', attributes' and methods' should obey to Java naming convention.
- Do not submit your project without first compiling it on dev machine.
- There will be again 3 days extensions (each day degraded by 10 points) in this project.
- You can ask your questions via Piazza and you are supposed to be aware of everything discussed in Piazza. You will be held responsible for any announcement and explanation made by the advisors.
- General discussion of the problem is allowed, but DO NOT SHARE answers, algorithms, source codes and reports.
- Ignore the cases which are not stated in this assignment and do not ask questions on Piazza for such extreme cases.
- You will use online submission system to submit your experiments. https://submit.cs.hacettepe.edu.tr/ No other submission method (email or etc.) will be accepted. Do not submit any file via e-mail related with this assignment.
- File hierarchy must be zipped before submitted (Not .rar, only .zip files are supported by the system). You must submit your work with the file hierarchy stated below:

<student_id.zip>/(Required)

→src/(Required)

→Main.java(Required)

→Stack.java(required)

→Queue.java(required)

→*.java(optional)

Policy

All work on assignments must be done **individually** unless stated otherwise. You are encouraged to discuss with your classmates about the given assignments, but these discussions should be carried out in an **abstract** way. That is, discussions related to a particular solution to a specific problem (either in actual code or in the pseudocode) **will not be tolerated**. In short, turning in someone else's work (from internet), in whole or in part, as your own will be considered **as a violation of academic integrity**. Please note that the former condition also holds for the material found on the web as everything on the web has been written by someone else.