Stack and Queue Assessment September 2024

Palettes are loaded into two vans which is only wide enough for one palette at a time. A van can hold 8 palettes at any one time maximum.

The vans are loaded from a pick list (**from data on a file**) at the depot and taken to the shop. Assume the storeman reads the file data and retrieves a palette and adds it to a van using a fork lift truck.

Write a program that simulates this action (using a stack). All error checking should be included in your solution.

Print out the Stack (i.e. the contents of the vans) at various stages to check the program works as advertised. Supply the following printouts, when Vans are empty, when one van is loaded, when all the items have been added and when both vans are full or file contents are complete.

Show that your program takes care of when both vans are full and an attempt to load more is attempted.

Stack and Queue Assessment September 2024

Check the system work for 0 palettes, 1 palette, 2 palettes, 15 palettes, 16 palettes more than 16. Relevant messages should be reported back to the users. A van should not leave for the shop unless it is full.

After each file has been processed print out the following :

Status of the vans (leave for the shop or waiting for more deliveries)

The van contents and the order in which they will be unloaded.

When the vans gets to the shop the contents are unloaded onto a conveyer belt. *All items from the vans are placed on the conveyor belt* ***before any transfer*** *to the shop store room is undertaken*. The shop conveyor belts can hold up to 16 items.

Once the conveyor belt is loaded the van drivers receives the storeman signature confirming delivery. The van drivers drive away. The storeman then starts the process of transferring the contents of the conveyor belt to the storeroom.

The storeman checks each item as it comes off the conveyer belt and places it in the first piece of available floor space which is stored as an Array List. As a final check the Array List is check against the original file to make sure all items were delivered. The contents of the Array List should be sent to a file called “ShopContents.txt”.

Supply printouts of the conveyor belt before the van arrives (.i.e. when it is empty), when the van has unloaded all the contents on to the conveyor belt. After half the contents of the belt have been removed. When all the items have been removed from the conveyor belt.

Note the Array list should be updated and displayed in synchronization with the conveyor belt being emptied.

Stack and Queue Assessment September 2024

There are 3 types of palette that are delivered to the shop.

These contain Food , Toys and pet food.

When the conveyer belt is unloaded there are 3 area to store the items Bay 1 Pet Food , Bay 2 Food and Bay3 Toys. Each bay is an arraylist.

The excess material from the previous delivery has been removed and put out for sale. Therefore you can assume the bays are empty before you load the new delivery.

After each loading bay is filled report the number of items stored in BAY1, BAY2 and BAY3

Create a solution implementing the **Stack** and **Queue** Abstract Data Types from the Java util library for the above scenario. Read the data from a file

Use a String to hold the item ID.

Test the program with the following two data runs. Include a test runs of your own.

Data run 1

Pod one F11, F34, P22, T56, F16, T77, P12, F41, T22.

Pod two P19, F39, T92, T36, P36, T75, P15, F48, F88.

Data run 2

Pod one T11, T34, P22, F56, T77, P12, F41, T11.

Pod two EMPTY

A printout to show the stack and queue at various stages.

- Show the state of the stack and queue before the vans are loaded.
- Show the state of the stack and queue after the stack has been unloaded.
- Show the state of the stack and queue after the loading bay is used.

Stack and Queue Assessment September 2024

You now have to use the following files instead of the Java util library files.

- **ListNode.java**
- **ListQueue.java**
- **ListStack.java**
- **ArrayQueue.java**
- **ArrayStack.java**
- **Queue.java**
- **Stack.java**
- **UnderFlowException.java**

Using the files above remove the java stack and queue library files and replace these files instead. Your program should still run with minimum modification. Retest to ensure that the program still works.

Supply printout of the stack code queue code used to implement the solution. Print the files that you have written not the ones supplied.

A printout to show the stack and queue at various stages.