CSE1322L Assignment 1 - Fall 2023

Introduction:

As a review of your knowledge of the concepts we learned in 1321, we will be writing the order-tracking part of a Point of Sales system for a fast-food restaurant.

A Point of Sales system is a combination of both software and hardware whose purpose is to assist merchants in making and tracking sales at the location where they are finalized (hence the name). The software involved can be responsible for several tasks, including accepting payments, printing receipts, tracking orders, and managing inventory.

Some restaurants, particularly fast-food restaurants, may have a screen behind the counter which displays which orders are ready and which ones are still being prepared, as shown below.

Orders

Ready
1976
2623
Preparing
2808
3188
6561

We will be implementing such a display.

Requirements

The features described below must be in your program.

- A total of three classes: the driver, Order, and OrderList
- The Order class keeps track of individual orders:
  - Must have 4 fields:
    - A private integer to keep track of the order's id number.
    - A private static integer to keep track of the next available id number. This variable should be initialized with a value of 1. (we'll cover what static variables are for at a future date)
    - A public Boolean called "ready" which keeps track of if an order is ready or not.
    - A public array of strings called "items", which keeps track of what items are in the order.
  - It must have 2 constructors:

        - The default constructor assigns the value of the next available id to the order's id, increments the static integer field by 1, sets "ready" to "false", and initializes "items" to a string array of size 3, with all strings being empty.
        - The overloaded constructor takes in one String array as its parameter, assigns the value of the next available id to the order's id, increments the static integer field by 1, sets "ready" to "false", and assigns the parameter passed to "items".
    - It must have 2 methods:
        - A method called getId() which returns the id of the order
        - An overload of the toString() method, which returns a string in the following format:

```
Order number: {order id}
{Ready/Not Ready}
{Order items, one per line}
```

        - Notice that the curly braces must be replaced with the appropriate information. For example, if an order of id 6295 that is ready and has 1 Hamburger and 1 Drink has its toString() called, it'll look like this:

```
Order number: 6295
Ready
1 hamburger
1 Drink
```

        - Similarly, an order that is has an id of 9329, is not ready, and has 1 apple pie, 3 cookies, and 1 drink will return the following:

```
Order number: 9329
Not Ready
1 Apple pie
3 Cookies
1 Drink
```

- The OrderList class keeps track of sets of orders:
    - It contains a single private field: an array of Orders called "orderList"
    - It contains 6 methods:
        - addOrder(Order order): Takes in a single Order object as a parameter and returns nothing. This method traverses the "orderList" and finds the first empty spot on the array. If it finds one, it inserts the order in that spot. If it does not find it, it must double the size of the array and copy over all the elements to the new array, after which it inserts the order in the first available spot. **The array doubling and the copying of its items must be done manually. You cannot use a built-in or imported method to do this.**

    - removeOrder(int id): Takes in a single integer as a parameter and returns nothing. This method traverses the "orderList" looking for the order whose id matches the one in the parameter. If it finds it, it removes that order from the array. If it doesn't find it, it does nothing.
    - readyOrder(int id): Takes in a single integer as a parameter and returns nothing. This method traverses the "orderList" looking for the order whose id matches the one in the parameter. If it finds it, it sets the "ready" field of that order to "true". If it doesn't find it, it does nothing.
    - sortOrders(): <u>This method must be private</u>. It takes no parameters and returns nothing. It sorts all the orders in "orderList" in ascending order. Empty cells can be placed either at the start or at the end of the array. **You must write the sorting code yourself. Any sorting algorithm will do. You cannot use a built-in or imported method to do this.**
    - printOrder(int id): Takes in a single integer as a parameter and returns a string. This method traverses "orderList" and returns the toString() method of the order whose id matches the one in the parameters. If it doesn't find an order with a matching id, it must return an empty string.
    - printOrders(): takes in no parameters and returns a string. It must first sort all of the Orders inside "orderList" using sortOrders(). Then, it must return all the orders in a string in the format below.

```
Ready
{orders that are ready, in ascending order, one per line}
Preparing
{orders that are not ready, in ascending order, one per line}
```

    - Using the example at the start of the assignment, it would look like this:

```
Ready
1976
2623
Preparing
2808
3188
6561
```

- The driver must contain the following:
    - at least the main method
    - It must declare one OrderList called "restaurantOrders".
    - It must contain a loop which prints the following options:
        - "1- Create order"
        - "2- Delete order"
        - "3- Ready order"
        - "4- Print order"
        - "5- Print all orders"
        - "6- Exit"

    - The loop must then read a number from the user and execute a selection statement with the appropriate option, according to the option the user picked.
        - 1: Creates a string array of size 3. It then asks the user for 3 items for their order, storing them in the array. The array is then used to create a new Order object through its overloaded constructor, after which this newly created Order object is passed to restaurantOrders' addOrder(). It should then print "Order has been added".
        - 2: Requests an order id from the user and passes it to restaurantOrders' removeOrder(). It then prints "Order has been removed", regardless of if any orders were removed.
        - 3: Requests an order id from the user and passes it to restaurantOrders' readyOrder(). It then prints "Order has been set to "Ready"", regardless of if any orders were set to ready.
        - 4: Requests an order id from the user and passes it to restaurantOrders' printOrder(). If printOrder() returns an empty string, print "No order with such id", otherwise print the results of printOrder().
        - 5: Calls restaurantOrders' printOrders().
        - 6: Terminates the loop

Considerations

- This assignment may seem intimidating, but that's just because of the number of things you have to do; the assignment itself isn't very hard, so don't be discouraged.
- Remember that you will get partial credit for partial work. Try to deliver as much of the assignment as you can.
- Static is a special keyword we will see more in depth later. Only one variable in your assignment should be static, and declaring it should look as below:
    - static int nextID = 1;
- Recall that an array of objects always initializes all of its indexes with the value "null".
    - When adding orders, you will know a space is empty because its value will be null: **if(array[0] == null){}**.
    - When removing an order, you will empty the index where that order was by setting it to null: **array[0] = null**;
    - When sorting the orders in the array, you can sort all the null values to the start or to the end of the array: **[null, null, Order, Order]** or **[Order, Order, null, null]** are both acceptable.
    - **Be very careful when reading the contents of the order list**. The order list could contain either order objects or null in any of the indexes at any given moment. If you try to do something with one of the elements of the array thinking that it's an order when it is in fact a null, your program will crash.
- Remember that strings can contain new line characters which, when ready by the printing method, will print an empty line. The first order example given above would look like "Order number: 6295\nReady\n1 hamburger\n1 Drink"

- **Remember that you must write the sorting code yourself (i.e.: you cannot use built-in or imported methods)**. Remember also that we provided code for sorting algorithms in all languages in one of the slides for 1321.
- You may add any other helper methods you believe are necessary, but they won't count towards your grade.

Example: [User input in red]

```
FastFood World Order Tracker

Please select a menu option:
1- Create order
2- Delete order
3- Ready order
4- Print order
5- Print all orders
6- Exit
5

READY
PENDING

Please select a menu option:
1- Create order
2- Delete order
3- Ready order
4- Print order
5- Print all orders
6- Exit
1

Enter item 1: 1 Hamburger
Enter item 2: 1 Medium Fries
Enter item 3: 1 Soda
Order has been added.

Please select a menu option:
1- Create order
2- Delete order
3- Ready order
4- Print order
5- Print all orders
6- Exit
```

```
1

Enter item 1: 2 Small Hambugers
Enter item 2: 2 Small Fries
Enter item 3: 2 Small Orange Juices
Order has been added.

Please select a menu option:
1- Create order
2- Delete order
3- Ready order
4- Print order
5- Print all orders
6- Exit
1

Enter item 1: 1 Apple Pie
Enter item 2:
Enter item 3:
Order has been added.

Please select a menu option:
1- Create order
2- Delete order
3- Ready order
4- Print order
5- Print all orders
6- Exit
5

READY
PENDING
1
2
3

Please select a menu option:
1- Create order
2- Delete order
3- Ready order
4- Print order
5- Print all orders
6- Exit
4
```

```
Please enter order id: 5

No such order found

Please select a menu option:
1- Create order
2- Delete order
3- Ready order
4- Print order
5- Print all orders
6- Exit
4

Please enter order id: 3

Order number: 3
Not Ready
1 Apple Pie

Please select a menu option:
1- Create order
2- Delete order
3- Ready order
4- Print order
5- Print all orders
6- Exit
3

Please enter order id: 3
Order has been set to "Ready".

Please select a menu option:
1- Create order
2- Delete order
3- Ready order
4- Print order
5- Print all orders
6- Exit
5

READY
3
PENDING
```

```
1
2

Please select a menu option:
1- Create order
2- Delete order
3- Ready order
4- Print order
5- Print all orders
6- Exit
2

Please enter order id: 1
Order has been removed.

Please select a menu option:
1- Create order
2- Delete order
3- Ready order
4- Print order
5- Print all orders
6- Exit
1

Enter item 1: 1 Cheese Burger
Enter item 2: 2 Medium Fries
Enter item 3:
Order has been added.

Please select a menu option:
1- Create order
2- Delete order
3- Ready order
4- Print order
5- Print all orders
6- Exit
5

READY
3
PENDING
2
4
```

```
Please select a menu option:
1- Create order
2- Delete order
3- Ready order
4- Print order
5- Print all orders
6- Exit
6

Shutting off...
```

Submitting your answer:

Please follow the posted submission guidelines here:
https://ccse.kennesaw.edu/fye/submissionguidelines.php

Ensure you submit before the deadline listed on the lab schedule for CSE1322L here:
https://ccse.kennesaw.edu/fye/courseschedules.php