CSE 102L Computer Programming Laboratory – Exercise 9

Due Date: 23:59 Sunday June 11th

Disclosure

You will submit your file to an assignment that is given through MS teams. Your filename should be *Ex9_yourStudentNumber.java*. Submissions made after the deadline will not be accepted, be sure to submit your work before the due date and make sure to click turn in button. Your code will be automatically controlled, so be sure to have the same class, method, variable names as described here. Failure to do so may result in you receiving 0 from this exercise. **All classes should be written to a single Java file. In a single java file there can only be single public class. Do NOT use Inner Classes. Be careful naming your file. If your editor inserts the file into a package, remove that line from the file but do NOT delete the import statements.**

Write set of classes according to the following specifications. Declare all attributes as **private** if not requested otherwise and use camelCase formatting for attributes.

YOU WILL CONTINUE FROM YOUR PREVIOUS EXERCISE (Exercise-8) (obviously change the name to Ex9)

Interfaces

1. Common <T>

- Methods:
    - isEmpty(): Boolean
    - peek(): T
    - size(): int

2. Stack <T> - child of Common with type T

- Methods:
    - push(T item): boolean
    - pop(): T

3. Node <T>

- Attributes:
    - int: DEFAULT_CAPACITY = 2
- Methods:
    - setNext(T item)
    - getNext(): T
    - getPriority(): double

4. PriorityQueue <T> - child of Common with type T

- Attributes:
    - int: FLEET_CAPACITY = 3
- Methods:
    - enqueue(T item): Boolean
    - dequeue(): T

5. Package <T> - **same methods with one additional method (just add this new method to interface)**

- Methods
    - Same methods from Exercise-8
    - getPriority(): double – if the class is Matroschka throws UnsupportedOperationException("Not implemented"). Otherwise returns distanceTo / item.price

Classes

1. Box – **Same class from previous exercise except Constructor**

- Attributes:
    - distanceToAddress: int
    - same attributes from previous exercise
- Methods:
    - Constructor that takes *distanceToAddress* in addition to item of type T (this is not a new constructor, change the existing one to this one, keep the no arg constructor)
    - Same methods from previous exercise

2. Container – implements Stack with type Box that can contain **anything**. Also implements Node and Comparable with Container.
   *Lower priority is better, this means if we have two containers with sum of the item prices inside of boxes are the same, the one with the less sum of distanceToAddress will be closer to head in queue, so it will have **priority** to leave the queue over other one. Implement Comparable according to this.*

- Attributes
    - boxes[]: array of Box type that can contain anything
    - top: int
    - size: int
    - priority: double
    - next: Container

- Methods:
    - No-Arg Constructor that instantiates a boxes with *default capacity*, sets *top* to -1, *next* to null, *priority* to 0
    - toString(): String – returns a String in the format:
        - “Container with priority: “ + priority
    - Implement methods from interfaces

3. CargoFleet – implements PriorityQueue with type Container

- Attributes:
    - head: Container
    - size: int
- Methods:
    - No-Arg Constructor that sets *head* to null, *size* to 0
    - Implement methods from interfaces

4. CargoCompany

- Attributes:
    - stack: Container
    - queue: CargoFleet
- Methods:
    - No-Arg constructor that initializes stack and queue objects
    - add(T box): None – given any type T that is a box that can contain **anything**, pushes the box to *container*. If successful, does nothing. If not, enqueues the *container* to queue. If it can enqueue successfully, creates a new *container* and assigns *stack* with newly created *container*, then calls itself with the same box. If enqueue is not successful, calls **ship()** method with queue.
    - ship(CargoFleet fleet): None – while the fleet is not empty, dequeues from fleet and calls **empty()** method with recently dequeued element.
    - empty(Container container): None – while the container is not empty, pops the elements and calls **deliver()** method with recently popped element and display what is returned from **deliver()** method.
    - deliver(T box): Sellable – given any type T that is a box that can contain **anything**, extracts the box’s content and return it
    - **all methods except add method in this class is private**