Middle East Technical University Department of Computer Engineering

CENG 443

Introduction to Object-Oriented Programming Languages and Systems

Spring 2022-2023

Lab 1 Quiz

Due date: May 2, 2023, Tuesday, 20:00

1 Introduction

Keywords: *Object-Oriented Programming, Error Handling*

In the Lab Quiz, you will extend an existing codebase using object-oriented concepts and design principles.

2 Overview

The assignment builds upon the preliminary. You will extend the example preliminary solution on ODTUClass for a number of new entities. You should restructure the code such that the code needs to be duplicated for functionality should be encapsulated in a reusable manner. In other words, instead of copy-pasting, you should use tools like inheritance and composition.

To give you an intuition, you can find a video of a sample run on ODTUClass page of the quiz.

3 Simulation Entities

To remember the roles of already existing entities, you can consult the preliminary text. Apart from **SimulationRunner Class**, you can change existing classes however you want. However, these changes must be non-intrusive and comply with object-oriented concepts and principles e.g. merging all game logic to a single class is not a valid solution. Below you can find the explanation of new entities and expected behaviors of the parts you need to redesign.

3.1 Display Class

In addition to drawing already defined entities, it should also draw salespeople and the supplier repeatedly also.

3.2 Common Class

You should define a few parameters for new entities here. You can find the parameters with their values below:

- **supplierMovementSpeed = 3** (speed of the supplier)
- **salespersonNo = 5** (number of salespeople)
- **salespersonStockStorageMin = 2** (minimum size of storage available for a salesperson)
- **salespersonStockStorageMax = 5** (maximum number of storage available for a salesperson)
- **salespersonMovementSpeed = 3** (speed of salespeople)
- **salespersonMoveInterval = 1000** (number of steps for a salesperson without a sale before moving to a new location)

Note that all parameters listed above are integers.

The class should also spawn multiple salespeople and single a supplier as necessary with random initial positions like previous entities and remove them when they leave the area like customers.

It should not spawn more than one supplier at any given time.

It should not replenish stocks of every store automatically, since it is the supplier's job now.

3.3 Customer Class

You should modify the customer class to interact with travelling salespeople:

- If a customer comes across a salesperson that sells first product in the shopping list, he should buy them from the salesperson.
- Salespeople are not stores. So when planning according to their strategies, customers should only consider stores e.g. even a salesperson sells the cheapest food, customers with the cheapest first strategy should go to the store that sells the cheapest food not to that salesperson.
- After buying from a salesperson, a customer might need to reevaluate his target e.g. a customer with the closest store strategy and with a shopping list starts with food, electronics and food, starts to move the closest shop sells food. After encountering a salesperson sells food, the customer should change his target to the closest store sells electronics.

3.4 Supplier

The supplier replenishes stocks of all stores by travelling. Traversing stores should be in the closest one first basis. After visiting each store exactly one time, the supplier should leave the area like customers.

The supplier should be drawn as green and nothing should be printed additionally.

3.5 Salesperson

You can think salespeople as moving stores. Like a store, a salesperson has a single product with a limited number of stocks. But unlike a store, whenever a salesperson out of stock, he does not replenish but leave the area like customers. Also, since customers do not consider salespeople when setting a target as explained previous section, the price of the product is irrelevant for salespeople.

A salesperson can have one of the following strategies:

- **Random:** As the name suggests, the salesperson moves to a random location.
- **Middle Ground:** The salesperson moves to middle of all customers.

For both strategies, when moving to the location, if a customer buys an item, salesperson selects a new location to move. Also when reached to the location, the salesperson stays there until,

- Either selling a product
- Some time passes which is defined by **salespersonMoveInterval**.

Salespeople should be drawn as blue and their product types, their stocks and their strategies (**R** for random and **M** for middle ground) should be printed.

4 Rules

- **Upload** your code to ODTUClass before time is up. Otherwise you will get **zero points**.
- Design and implement your solution according to object-oriented concepts and design principles.
- Whenever possible, you **should** reuse parts of the existing code.
- When reusing the code, you **should not** copy-paste existing code and utilize tools like inheritance or composition. For that you can define new classes and/or change hierarchy of the classes.
- Ensure that you handled erroneous situations.
- Make sure to document your additions and modifications of the code via comments.
- Use Java 8 or a newer version

Failing to comply the rules above will cause you to lose significant number of points.