Capstone 2

IMPORTANT: Make all of your changes for this capstone in a single feature branch in your student repository, and merge it into develop when you are ready to turn in your assignment.

Contents

Project Setup

Using IntelliJ, create a Maven Project:

- Project SDK - 11
- Create from archetype - maven-archetype-quickstart
- Name – capstone2
- Location - [student repository root]/capstone2
- GroupId - com.teksystems.bootcamp.capstone2
- ArtifactId: capstone2
- Version - 0.1-SNAPSHOT): 1.0-SNAPSHOT
- Leave the remaining settings at their defaults and ‘Finish’ creating the project

In IntelliJ, open the ‘capstone2’ folder

CAPSTONE EXERCISE

Prompt

Create a Point-of-sale application for a restaurant, fast food, carryout, delivery, etc., that allows the user to:

- Create an order for a customer
- Customize the order and components of the order
- Provide a "receipt" (as console output)
- Create additional orders after a receipt is provided
- Pre-defined menu items (ex: Double Burger, Veggie Burger, etc.)
- Combo Meals
- Different pricing for pre-defined items and combos, versus building them "from scratch"
- Retrieve and display previous receipts based on the order number
- The final order price must account for tax
- Users must be able to order at least:
    - Entrees
    - Toppings for the entrees
    - Drinks
    - Sides

WARNING

The capstone project is a demonstration of your understanding of the topics covered to-date in the boot camp. You may need to reference tutorials or Q&A sites such as StackOverflow to complete parts of the project. It is acceptable to use tutorials/Q&As to learn how to add specific functionality to your project, but you cannot simply copy a tutorial repository, follow a single tutorial from start to finish, or copy the code from a Q&A site to develop your application. See guidelines below for using tutorials:

- Do not fork a tutorial repository. This would be considered plagiarism and result in an automatic grade of zero and potential removal from the boot camp.
- Do not simply follow the tutorial from start to finish.
- Do not copy code from a Q&A site and then change it for your application (a few lines of code is permissible)
- You must be able to explain how all functionality included in your project works.
- ***<u>List ALL tutorials or aids used, either in your Read Me file, and/or as code comments. Include:</u>***
    - ***<u>The name of the tutorial,</u>***
    - ***<u>Which section(s) you used, and</u>***
    - ***<u>The functionality you learned from the tutorial.</u>***
- Failure to identify tutorials/Q&A responses will result in significant reduction of your grade.

- If you work with other bootcamp members – or instructional team members – to find solutions, that is acceptable, but must also be clearly identified

We recommend that you commit and push your code to your remote repository at least every few hours. Failure to do so may result in a loss of your code if anything happens to your work machine. If this happens, you will not be given extra time or resources to allow you to complete your project.

Instructions

MVP

Note: It is highly recommended to **design your application first**, before starting to code, creating user stories, etc. This doesn't have to be reflective of your final product but will help you think through how to fit all the pieces together.

Under your capstone 2 namespace, add appropriate classes to create the application described below. Both the core logic and front end should be written in Java and present the user with a command-line application. The application should have the following features/ functionality:

- The application is usable, with working logic and user interface
- Clear separation between the application logic, and the UI (e.g.: the UI should call a class/classes that perform the actual logic, and the UI should only handle inputs and outputs
- Understandable UI interaction, which guides the user through the process
- Follows basic Object-Oriented patterns, practices, and principles
- 50% unit test coverage for the application's logic and classes
- Error/ exception handling and validation of user input (where/ when applicable)

When you submit your project, include your design documents in a top-level folder; these documents should include both your UI wireframe, and your design documents for the point-of-sale application logic. They can consist of:

- digital files
- A document with URLs and descriptions, and/ or
- (pictures of) hand-drawn work

When creating your application, commit and push often, and label "important" commits – at a minimum, this should include the MVP which has the basic functionality, and the final version that you will present at the end of the capstone development period.

When presenting, ensure you complete the presentation in the allotted time; you should also be able to show that you understand the approach you used, and provide insight about the decisions you made

POST-MVP

A solution that meets all of the requirements in this lab constitutes a grade of Proficient (3/5). To earn a higher grade, your solution must also apply the code quality concepts you've learned up to this point. For example, extremely well-organized code, following SOLID principles, clear understanding and implementation of OOP principles. Adding additional functionality that does not improve the quality of the code and only serves to make your assignment look more impressive, will not help to raise your grade.

- Sufficiently detailed documentation, identifying the intended features and functionality, that provides a clear understanding of the intended implementation
- Intuitive UI functionality/prompts
- Well-organized code, use of naming conventions and coding standards and practices
- 80% unit test coverage for the application’s logic and classes
- Good use of design patterns where appropriate
- Well-rehearsed and clear presentation, with very clear responses to questions and feedback