Lab: Object-Oriented Design Patterns

Lab Overview

In this lab you will be creating several small java projects to assess your knowledge and understanding of Object-Oriented Design Patterns.

While performing this lab, incorporate the principles and practices that you've learned from prior lessons in this course; for this Lab, be sure to consider

- Object-Oriented Programming
- Object-Oriented Class Structure
- SOLID Principles
- Unit Testing

It is highly recommended that you **design your solution 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.

Submission Instructions

Connect to CodeGrade prior to doing any work, to ensure your changes are recognized and processed.

Create a new feature branch in your TEKbootcamp repository called "designpatterns". Commit and push frequently. When your work is done and ready for review, merge the feature branch into the develop branch.

Following instructions for CodeGrade submission:

1. Click on the "Launch exercise" button on the exercise page at TEKsystems Academy.
2. In the new CodeGrade window, click on the "Connect Git" button.
3. Select GitHub as your Git host. Connect your TEKbootcamp repository to the exercise*.

After selecting a repository, this repository will be cloned to CodeGrade as a first submission. After this, you can start to use Git like you usually would. Every time you push to the develop branch, it will automatically result in a new submission in CodeGrade. Pushes made after the submission deadline will not be taken into consideration.

* You will only have to log in and authorize CodeGrade once; after that, it will be available for all your other assignments inside CodeGrade too.

WARNING: Be sure to connect the proper repository to the exercise. You will not be able to connect to another one immediately using the CodeGrade connection. You can undo your Git connection with CodeGrade by revoking access to the External Tool "CodeGrade" in your GitHub account only.

Grading Expectations

A solution that meets all 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. 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.

Plagiarism Warning

This assignment 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 assignment. 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 are permissible)
- You must be able to explain how all functionality included in your project works.
- List ALL tutorials or aids used, either in your README file, and/or as code comments. Include:
    - The name of the tutorial,
    - Which section(s) you used, and
    - The functionality you learned from the tutorial.
- Failure to identify tutorials/Q&A responses will result in significant reduction of your grade.
- If you work with other boot camp members – or instructional team members – to find solutions, that is acceptable, but must also be clearly identified

This exercise expects that you will copy code from one exercise to another – this does not count as plagiarism (unless it’s not your code) and you are not required to re-write those copied files from scratch (but you will need to modify them).

Estimated Duration

15-20 hours

Project 1: Design Patterns

Project Overview

This project consists of multiple exercises, with each one focused on a single design pattern. The use of additional design patterns where/if appropriate is recommended, but not required – if you do include other patterns, provide your rationale in the READ_ME for this assignment. Each exercise should also have a main method (to display the behavior) and unit tests (to verify correct behavior/error-handling).

Project Setup Instructions

NOTE: you may need to add a JAVA_HOME variable to your User environmental variables - check online for instructions specific to your machine

- Using IntelliJ, create a Maven Project:
  - Project SDK - 11
  - Create from archetype – leave unchecked
  - Press Next
  - Name – ood_exercises
  - Location - [student repository root]/ood_exercises
  - GroupId - com.teksystems.bootcamp
  - ArtifactId: ood_exercises
  - Version - 0.1-SNAPSHOT): 1.0-SNAPSHOT
  - Leave the remaining settings at their defaults and 'Finish' creating the project
- In IntelliJ, open the 'ood_exercises" folder

Exercise 1: Abstract Factory

Files Location: *Under the com.teksystems.bootcamp.ood_exercises folder/namespace add a new folder named "factories", and appropriate files to provide the required functionality.*

Instructions: In Santa's workshop, children who have been marked as "naughty" are given a chunk of coal, and nice children are given a toy. Given a list of children (the Naughty/Nice list), your application should provide the appropriate type of gift. Create the following base classes and interfaces, as needed:

- An Elf interface, and 2 types of elves that implement it (ToyElf and CoalElf)
- An Abstract Factory to decide which elf to create for each child on the naughty/nice list
- It is up to you to decide how to implement and use the Naughty/Nice list

Exercise 2: Façade

Files Location: *Under the com.teksystems.bootcamp.ood_exercises folder/namespace add a new folder named "facade", and appropriate files to provide the required functionality.*

Instructions: Use the Façade pattern to create a class that represents a user making an online purchase, by orchestrating the steps of the process when working with the various backend systems (represented by single classes). This exercise is intended to be abstract and not necessarily represent a real-world system.

Create appropriate classes and interfaces to allow your façade to:

- Accept a purchase request from the user
- Verify with the Inventory system to see if the items are in stock
- Interact with the Billing system to generate a bill
- Pass information to the Payment system to make the purchase
- Send address details to the Shipping system
- Inform the user that the purchase is complete (or failed)

Exercise 3: Template Method

Files Location: *Under the com.teksystems.bootcamp.ood_exercises folder/namespace add a new folder named "template", and appropriate files to provide the required functionality.*

Instructions: Find or create a 1-3 paragraph (approximately) short story and create a Template class that will change out certain parts of speech, Mad Libs-style (https://en.wikipedia.org/wiki/Mad_Libs). As an example, in the Template class, you might have a public tellMeAStoryMethod, which returns "when I cross the " + getPartOfCity() + " I always look " + getNumber() + " ways." This class will have an abstract method for every part of speech that will be replaced, and that the derived classes will override to provide that information.

Your exercise should include:

- The Template base class with abstract methods for the various fill-in-the-blank parts of the story
- A derived class that overrides the template's methods and provides the original/proper parts of speech (in the example above, "street" and "both"), and always returns the same story
- A second derived class that overrides the template's methods and provides randomly selected parts of speech every time one of the methods is called (in the example above, it might randomly pick from "Street", "Road", "Park", Interstate", "Financial District", etc. for the first abstract method, and randomly pick a number for the second).