CS-665: Software Designs and Patterns

Class Project

This document should not be disseminated outside the purview of its intended purpose.

You Choose (20 points)

GitHub Project Template Link
https://github.com/edorsini/cs-665-project-template

Application Description

One of the key requirements for this course is the final project, which counts for 20% of the grade and must be completed individually. This project presents an opportunity for you to demonstrate your software design skills by solving a problem that interests you and presents a suitable challenge.

During the course, we have covered a variety of design patterns, including the Strategy, Factory Method, Abstract Factory, Singleton, Prototype, Command, Observer, State, Template, Facade, Decorator, Composite, Adapter, Proxy, Iterator, and Mediator patterns. However, there are many additional design patterns that can be used in Java programming, as listed on the website Java Design Patterns.

Boston University Metropolitan College
Ed Orsini | edorsini@bu.edu

The final project provides you with the opportunity to showcase your understanding and use of design patterns in solving a real-world problem.

Your main objective for the class project is to expand your knowledge of design patterns. You should choose a design pattern that was not covered in our class, study it on your own, develop a unique use case scenario for it, implement the pattern in your preferred programming language, and create a presentation that showcases the design pattern and your scenario.

It is highly recommended to utilize a combination of design patterns in your project, as this mirrors the way design patterns are utilized in real-world software projects.

To get started, you should review the list of design patterns and select one that you find interesting. Some examples of important patterns that were not covered in depth in our class include the Bridge, Builder, Callback, Delegation, and Thread Pool patterns.

Please note that your project idea should not be taken from existing books or websites and should be a unique project based on your own original ideas.

Also note that the implementation of a graphical user interface is not necessary. To demonstrate the functionality of your implementation, you should implement unit tests.

Tasks

Implementation Description (2 points)

In your implementation of this application, it is important to consider software design principles. This section outlines the main software design concepts and their goals.

For example:

- Explain the level of flexibility in your implementation, including how new object types can be easily added or removed in the future.
- Discuss the simplicity and understandability of your implementation, ensuring that it is easy for others to read and maintain.
- Describe how you have avoided duplicated code and why it is important.
- If applicable, mention any design patterns you have used and explain why they were chosen.

We recommend that you write this description in a `README.md` file using MarkDown format (https://spec.commonmark.org/current/) and add the file to the root folder of your project. This should be done after completing the other tasks in this assignment.

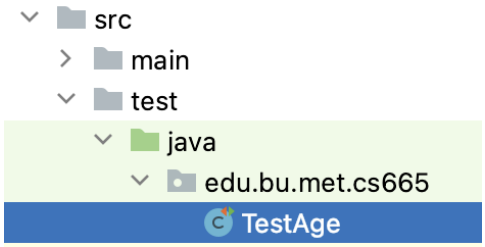
UML Class Diagram (5 points)

Develop a class model for your application, consisting of 5 to 8 of the most crucial classes, that encompasses the features of the described use case scenario. Only include essential and non-trivial methods.

Java Solution (13 points)

- Follow the project template given to implement your project.
- Submit a zip file that includes the implementation package, with a `README.md` file explaining how to compile and run the implementation. Ensure that the zip file includes all subdirectories of the project, excluding any binary files. The zip file should not exceed 10MB in size and should only contain source files, not generated binaries.
- Provide clear and thorough documentation within the code. It is best to write the documentation as the code is being implemented, rather than postponing it for later.
- Adhere to the Google Java Style Guide (https://google.github.io/styleguide/javaguide.html).
- Ensure that the solution can be compiled using the **`mvn compile`** command.
- Implement JUnit tests to verify the functionality of the implementation. **A minimum of 3-5 JUnit tests required.**

The example below is found in the project template as an example in the following file:



```java
@Test
public void testSetFirstName() {
    // Given: a student object with the following first name and last name
    Person student = new Person("John", "Doe");

    // When: the student's first name is changed via the following setter method
    student.setFirstName("Bob");

    // Then: we confirm the expected result is the same as the value obtained from
    // the getter method
    assertEquals( expected: "Bob", student.getFirstName());
}
```

Submission

When you have completed your assignment:

1. Ensure that you have the latest version of your code saved on your computer.
2. Compile all results from the three tasks into a single document, such as a PDF file for the UML diagrams.
3. Zip all of your code and the document together into one .zip file. Remember to remove any binary files, which are usually found in the bin/ or target/ folders, as they can significantly increase the size of your zip file.
4. Verify that you have correctly uploaded the zip file. To do this, download the file, unzip it, and confirm that the contents are correct and that the file is not damaged. Please note that we will only be able to evaluate the zip file uploaded to the blackboard, and any incorrect or damaged files cannot be evaluated.

After completing your assignment, you can download a ZIP file of your repository using the green download button on GitHub. Make sure to upload this ZIP file to Blackboard. It's important to note that we will be grading both the final ZIP file uploaded to Blackboard and the history of your GitHub repository. Both versions should match. The purpose of having a ZIP file on Blackboard is to provide an archived copy of your assignment.

Grading

Your solution should be a standalone program that can be compiled and executed following the instructions provided in the `README.md` file. It's recommended to utilize the provided project template and utilize build tools like Maven to integrate your implementation. If your program satisfies all the required functionality, compiles, and runs successfully, you will receive full points. Grading will be based on the following evaluation criteria, and points will be deducted for each task accordingly.

- Your UML diagram will be missing important components such as Interfaces/Classes, which will result in a 5% reduction for each missing component.
- To compile your solution, we will use the "**`mvn clean compile`**" command after downloading, unzipping, and running the command on your project. Your code must compile using Java JDK 1.8 or else it will result in a 10% grade deduction for the implementation task.
- If your code includes functionality bugs, a 10% deduction will be applied for each bug found.
- Your submission should include a README.md file that clearly explains your conceptual solution, the steps to compile and execute the code. Failure to include such a file or not providing all requested information will result in a 10% reduction of points.
- Your program must implement the requested functionalities, and if it does not, a 10% deduction will be applied for each missing functionality.
- We will use jplag (https://github.com/jplag/jplag) to programmatically check for plagiarism. Any solutions that are found to be an exact duplicate of someone else's will not be accepted, and we will contact you regarding the issue.

Late Work

Late work will not be accepted. We understand that exceptions can be made in extreme circumstances with proper documentation. For instance, if you provide a doctor/dentist note that verifies you were unable to meet the deadline due to illness, an extension may be granted.

Academic Misconduct in Programming

In a programming course like ours, it's crucial to understand the line between acceptable collaboration and academic misconduct. Our policy on collaboration and communication with classmates is straightforward: you may not share or receive code through any means, including visually, electronically, verbally, or otherwise. Any other forms of collaboration are permitted.

When it comes to communication with individuals who are not classmates, TAs, or the instructor, it is strictly prohibited. This includes posting questions or seeking assistance on programming forums such as StackOverflow.

When using external resources such as the web or Google, a "two-line rule" applies. You may search for information and access any web pages you need, but you may not incorporate more than two lines of code from an external source into your assignment in any form. Even if you alter the code, such as by changing variable names, it remains a violation to use more than two lines of code obtained from an external source.

It is important to properly cite your sources by adding a comment to your code that includes the URL(s) consulted during the construction of your solution. This not only helps to ensure academic integrity but also aids in later recollection of your thought process.

Boston University Metropolitan College

Ed Orsini | edorsini@bu.edu