Cape Peninsula University of Technology

Applications Development Practice 3 (ADP372S)

Assignment 1

Due Date: Friday, 28th March 2025 at 23:59

Lecturers: Radford Burger, Temuso Makhurane, Kruben Naidoo & Israel Tchouya'A Ngoko

Section A: Overview

The aim of the assessment will be to ensure that the learner has installed the appropriate IDE on his/her pc and demonstrates the ability to use **Git** and **Github** to collaborate on the same project with other members of the team.
Furthermore, the learners will be assessed on the implementation of the **Maven** build in their project and also show how **Test Driven Development** (TDD) is applied in the project solution. This assignment forms the base of capstone project in ADP372S that the group will present at the end of the year. The capstone project is an enterprise application coded in Java using the best practices taught in class.

In this assignment, the group will articulate their domain problem as a UML class diagram which will be saved in the Readme.md file of the project solution. Thereafter each team member will implement a portion of the domain using the concepts of Domain Driven Design described thus far.

The following sections discuss the requirements in more detail.

Section B: Setting up the project and collaborating on Github

1. Each team member will setup/create your GIT account on **Github.com**.
2. The team leader (is also a team member) will then:
   a. set up a **maven project** in Java using **IntelliJ** (the basecode must include the necessary pom.xml and .gitignore files etc), then upload/push from your local to your **Github.com** repository (remote) and;
   b. invite team members as collaborators.
   c. the team member sets up **milestones** and **issues** for the team members in the tasks described below.
3. Each of the team members will then accept the collaboration invitation and fork the project onto their Github.com account. In addition, the team member will clone the project onto their local.
4. Each member (including the team leader) may create a branch (use your student number as the branch name) and use this to experiment with completing tasks but that may be merged into the master branch of their project. This may include model classes or test classes that demonstrate the (TDD) test fixtures as appropriate.
5. Each member will be responsible for coding at least one model class, and the corresponding factory class. Furthermore, the team member will demonstrate their understanding of TDD, by coding the factory test class.
6. Another milestone is for the team member to implement a repository and repository test for that class as discussed in your lessons. The group should adhere to good design principles taught for example, code to abstraction not to concretion, and so forth.
7. The team member must remember that if you are working on your branch, merge from your branch into the master, and send a pull request from your master to the team leader.
8. Thereafter the team leader will add the member's contributions to the master branch and ensure that the pull-request is error-free.
9. Each member must ensure to get the updated version of the project after every milestone is completed (be in sync with the team lead) and ultimately before submission of the group assignment.

Hint: Kindly refer to the notes and video recordings posted on Blackboard.

Section C: UML

The group will choose an appropriate problem domain to model. The diagram should depict all the Domains (or Entities) and their relationships. Remember to include the relationships and multiplicity as well. The class diagram can be constructed at a high- level view (as such include the attributes of the entity but not the methods for now). This problem should **NOT** be the same problem as the one you are working on in Project 3 (PRT362S). You may use StarUML (or any other modelling tool) to construct your uml class diagram. The UML class diagram must be saved in your Readme.md file of your project. More details are discussed below.

Section D: Implementing the UML domain problem using IntelliJ

In the real world, it is known that the success of a project is usually the result of what is known as careful planning, which goes hand in hand with the talent, and collaboration of a project's team members. This goes further with the formation of team and group projects. Consequently, when a company forms these teams, the members of the group may behave in whatever specific way, but working in individual parts. This is normal, however, it is in the best interest of the management to pay attention to **team roles** and **responsibilities. This is not done to micromanage the members, however,** to ensure the team works in a very effective manner. As a team, you have all come together and select an approved **design problem** in a **problem domain.** You have also designed a **domain model (UML class diagram)** for your design problem depicting **your domains (entities) and their relationships.**

Section D1: DDD- Domain

You need to write code for your entities in the domain model using Builder Pattern. Remember to choose your attributes and their corresponding datatypes carefully. Domain (or Entity) is one of the building blocks of Domain Driven Design.

Activities

1. **Pre-coding Design Activities**
   The group must design their Domain model using StarUML or any tool of your choice.

2. **Group Lead Tasks:**
   - Create a maven application using IntelliJ that will act as the code base for your project.
   - Create **one entity** as shown on your domain model using the **Builder Pattern.** Remember entities must be in a package called **domain.**
   - You also need to **create a Github repository** for your project and **push the code base** to it.
   - Add your group members as collaborators on the GITHUB repository and make sure they accept.
   - Suggestion: Create a Milestone with a due date: **Friday, 14th March 2025**, and title: **Entity Milestone.**
   - Create issues (creation of entities using Builder Pattern) and assign them to all group members, also assign the issues to the Entity Milestone. Each group member must be responsible for at least one entity.
   - Merge correct, error-free pull requests from group members.

3. **Group Members Tasks:**
   - Accept the collaboration invitation sent by your group leader on Github.
   - Fork the Github repository created by your **group lead** onto your Github account, and then make a **clone** of your **remote** on your **local** repo.
   - Solve the issue (in this case for example, create the code for an entity using Builder Pattern) assigned to you by your Group Lead.
   - Push your code to your upstream.
   - Send your code (using pull request) to the Group Lead.

As soon as the group lead has updated the GitHub repository, get the updated version (typically a sync will happen after the milestone due date and the team lead gives you the go ahead when all team members completed their tasks).

Section D2: DDD- Factory

The second coding part of the practical assignment is the factory for your entities. **Factory** is another building block of Domain Driven Design.

Activities

1. **Group Lead Tasks:**
    - Create a **factory** package under your root package.
    - Create one factory class for an entity. **Use TDD**.
    - Update your repository on GITHUB with the new code.
    - Suggestion: Create a Milestone with a due date: **Monday, 17th March 2025**, and title: **Factory Milestone.**
    - Create issues (creation of factory classes) and assign them to all group members, also assign the issues to the **Factory Milestone. Remember,** each group member was responsible for at least ONE entity, and therefore would create the related Factory class for each entity.
    - Merge correct, error-free pull requests from group members.

2. **Group Members Tasks:**
    - Get the updated GITHUB repository from your group lead.
    - Solve the issue assigned to you by your group lead. Use TDD.
    - Send (using pull request) your code to the group lead.
    - As soon as the group lead has updated the Github repository with all team member tasks merged, get the updated version.

Section D3: DDD- Repository

Your next practical coding task is the repository. Repository is another building block of Domain Driven Design.

Activities

1. **Group Lead Tasks:**
    - Create a **repository** package under your root package.
    - Create the main generic repository interface called **IRepository** (with the create, read, update and delete methods) in the root of the repository package.
    - Then create a Repository interface with its implementation class (~~in the **impl** package~~) for an entity. **Use TDD**.
    - Update your repository on GITHUB with the new code.
    - Suggestion: Create a Milestone with due date: Sunday, **23th March 2025**, and title: **Repository Milestone.**
    - Create issues (creation of repository interface and implementation classes) and assign them to all group members, also add the issues to the **Repository Milestone.** Each group member must be responsible for repository for their related entities.
    - Merge correct, error-free pull requests from group members.

2. **Group Members Tasks:**
    - Get the updated GITHUB repository from your group lead.
    - Solve the issue assigned to you by your group lead. Use TDD.
    - Make sure your repository interface and its corresponding implementation are in the right packages.
    - Send (using pull request) your code to the group lead.
    - As soon as the group lead has updated the repository with all team member tasks, get the updated version.

All group members must submit their copy (origin) of the GITHUB repository to Blackboard. That is the repository on your GITHUB account, which must be in sync with your group lead's GITHUB repository.

Section E: Test Driven Development

Each team member will have test classes (in the appropriate Test Packages) for the Domain/Entity they implemented. These test fixtures will be for testing the Factory and Repository.
All group members must submit **their copy (origin) of the GITHUB repository**. **Note:** That is the repository on your **GITHUB account**, which must be in **sync** with your **Group Lead's GITHUB repository on/before Friday, 28th March 2025.**

Section F: Submission

- You should complete the assignment in a group of 5-6 members. You may choose your own group BUT from within your own class group (for example, 3D). These groups will be set on Blackboard by your Lecturer. The Group Lead should send an email to your Lecturer only with each of the member details as soon as possible.
- The **Author** (group member name + student number) can be included in the comments of each Java source file in the **IntelliJ** project where applicable.
- This is a coding project and as such the members of the group must demonstrate that they have contributed equally to the coding of the application and that the work was distributed fairly. *Marks may be awarded differently for each member based on their contributions.*
- Please note that all students are required to submit the **link to your GitHub repository in the comments section of the upload.** This is a **group submission** so only ONE group member (preferably Group Lead) will do the upload and in the comments list the all the team member names, student numbers, and links to their Github repo.
- **The due date for this assignment is Friday 28th March 2025 at 23h59**.

Further note: all members should add appropriate comments to their program code for example the following sample comments should be placed at the top of your source code file,

```
/* Employee.java
   Employee model class
   Author: Ross Barth (217149073)
   Date: 04 March 2025
 */
```