Will Loughlin and Sophia Chaltas
OOAD Project 6

Semester Project: Gradebook

Status Summary

Work Done: Written description of the work done in the first week of your project and (in the case of multi-person teams) the breakdown of work across team members.

- https://github.com/WillLoughlin/GradeBook
- School, User, Teacher, Student, Class, and Logger classes created and implemented with command line back end menu to test functionality.
- Saving and Loading completed for Users and Classes in Users.csv and Classes.csv.
- Functionality for:
    - Add/Remove Teachers, Students, and Classes
    - Add/Remove Students from Class
    - Set/Change Teacher of Class
    - Basic UI development
    - Automatic Saving/Loading for Classes and Users

Changes or Issues Encountered: Has anything changed so far in your approach to the project from the initial design in Project 5?

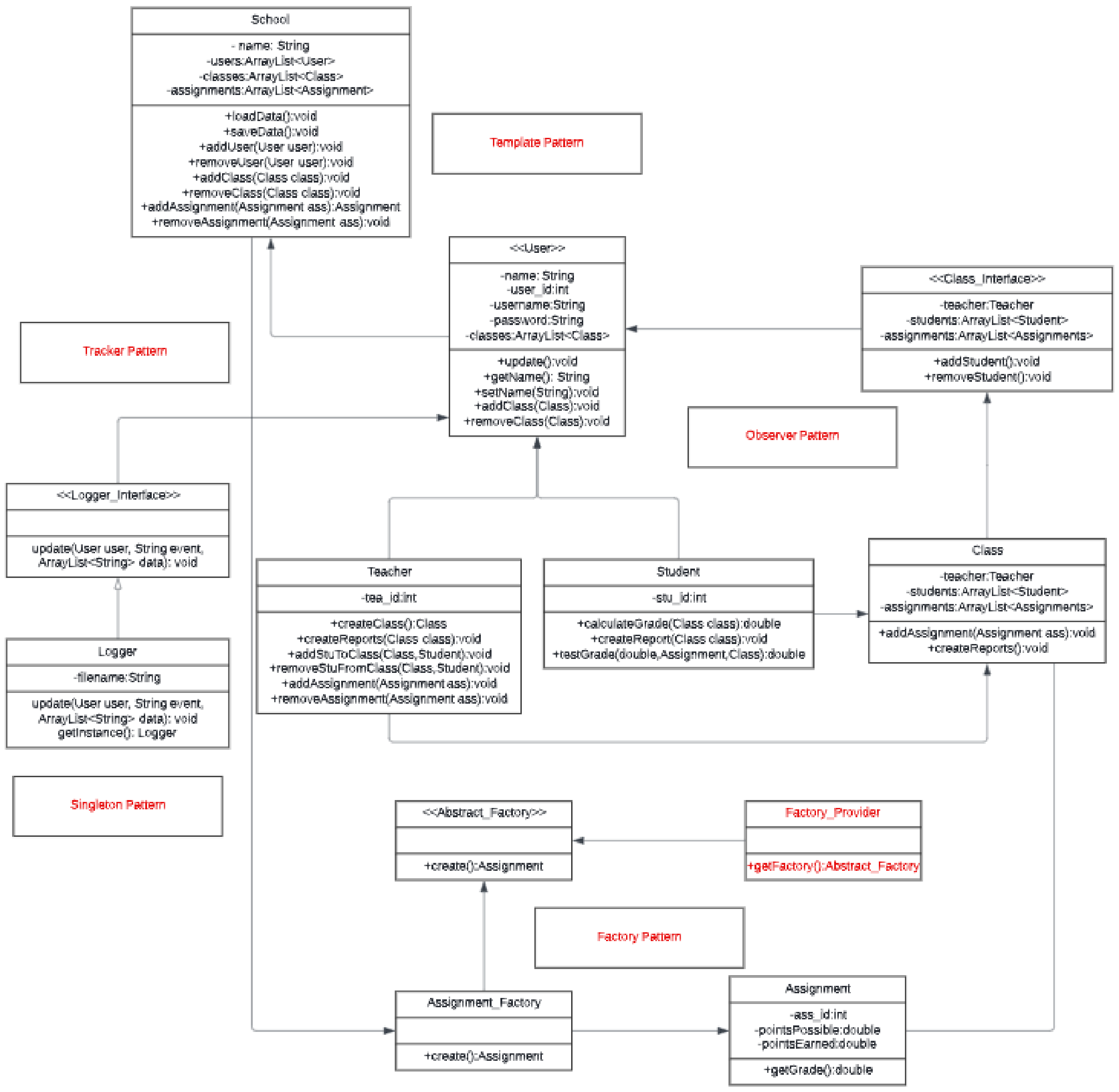
- The user interface Vaadin is a little more difficult to implement than anticipated but it will still work. It takes time to learn because every time you make a change and refresh it takes a minute or two to reload the frontend.
- Other than a steep learning curve with the user interface the project is moving along as planned and there shouldn’t be any issues getting it done on time.
- The overall class structure is the same as the proposal, other than added variables and functions there are no significant changes.

Patterns: Now that you have more of your system implemented, please describe the use of design patterns so far in your prototype and how they are helping you or your design.

- Template
  - We use the template pattern where we define the User as a skeleton of patterns to be implemented by either Students or Teachers.
  - Students and Teachers have similar needs but with slightly different implementations. The template pattern is effective here because it is helpful for them to have the same function names but they cannot be implemented in the same way.
- Tracker
  - The tracker pattern is implemented with the Logger class that tracks actions taken on the school.
  - These actions are stored in the Logger.txt file and mark events like the creation and removal of students, classes, and any other significant events.
  - This kind of tracker is helpful to understand what has been happening to the school with multiple users.
- Singleton
  - We use the singleton pattern with the tracker to ensure that there is only one tracker active at a time.
  - This is done with a private constructor and a getInstance method to access the logger's methods.
- Observer
  - The observer pattern is used to model the relationship between students and classes.
  - Students are added to classes as observers that are updated with changes to the class.
  - The student observers can be added and removed from any class at any time.
- Abstract Factory
  - The abstract factory pattern is used to create assignments in our project.
  - Assignment creation is very important as it stores the main information used in our GradeBook.
  - We use a factory to ensure that this creation goes smoothly.

Class Diagram

The class diagram did not change much since we planned well. We moved the implementation of the abstract factory to the school and updated the classes to demonstrate how we used the pattern.


School
- name: String
-users:ArrayList<User>
-classes:ArrayList<Class>
-assignments:ArrayList<Assignment>
+loadData():void
+saveData():void
+addUser(User user):void
+removeUser(User user):void
+addClass(Class class):void
+removeClass(Class class):void
+addAssignment(Assignment ass):Assignment
+removeAssignment(Assignment ass):void
Template Pattern
<<User>>
-name: String
-user_id:int
-username:String
-password:String
-classes:ArrayList<Class>
+update():void
+getName(): String
+setName(String):void
+addClass(Class):void
+removeClass(Class):void
<<Class_Interface>>
-teacher:Teacher
-students:ArrayList<Student>
-assignments:ArrayList<Assignments>
+addStudent():void
+removeStudent():void
Tracker Pattern
Observer Pattern
<<Logger_Interface>>
update(User user, String event, ArrayList<String> data): void
Teacher
-tea_id:int
+createClass():Class
+createReports(Class class):void
+addStuToClass(Class,Student):void
+removeStuFromClass(Class,Student):void
+addAssignment(Assignment ass):void
+removeAssignment(Assignment ass):void
Student
-stu_id:int
+calculateGrade(Class class):double
+createReport(Class class):void
+testGrade(double,Assignment,Class):double
Class
-teacher:Teacher
-students:ArrayList<Student>
-assignments:ArrayList<Assignments>
+addAssignment(Assignment ass):void
+createReports():void
Logger
-filename:String
update(User user, String event, ArrayList<String> data): void
getInstance(): Logger
Singleton Pattern
<<Abstract_Factory>>
+create():Assignment
Factory_Provider
+getFactory():Abstract_Factory
Factory Pattern
Assignment_Factory
+create():Assignment
Assignment
-ass_id:int
-pointsPossible:double
-pointsEarned:double
+getGrade():double

https://lucid.app/lucidchart/b0ba452c-46df-4b30-b79f-27d96fa941c0/edit?invitationId=inv_4a6ed4b0-c859-4f3a-876e-66634cfc10e8&page=0_0#

Plan for Next Iteration

The backend of our project is fully complete at this point so all we have left to do is create a front end for users to interact with the stored data. We will need to create a login/register page, a teacher portal and a student portal. Teachers and students will need to be able to check grades and teachers will need to be able to create and edit assignments. Once we have a finished front end our project will be completed. This will be done using Vaadin components and will be accessible via localhost.