Name: Constantinescu Iulia-Andreea
Group: 323CC

Implementation

This homework took approximately 10 days. I found it quite appropriate concerning the level of difficulty. The hardest thing for me was the ScoreVisitor class and especially the visit functions.

1. Application architecture

1.1 Catalog Class

- For this class I used Singleton design pattern accordingly to make sure that is the same instance everywhere
- I created addCourse and removeCourse methods to add and to remove courses from the catalog
- The class also contains 3 methods that will help to the ParentPage interface (addObserver, removeObserver and notifyObservers), the first 2 for adding and removing parents from the list of observers(subscribers to the Catalog) and the last one "gives" notification to a parent's students

1.2 User, Student, Parent, Assistant, Teacher Classes

- These Student, Parent, Assistant, Teacher classes are subclasses of abstract class User and it is used to simplify instantiation
- For these classes it is used Factory design pattern to create and return instances of all the users
- Student Class also has 2 methods for setting the parents and a compareTo method
- Parent Class has also a method(update) for updating the notification received

1.3 Grade Class

- This class has getters and setters for all its fields
- Grade Class also implements Comparable(so I also had to do a compareTo method) and Cloneable interfaces(that will help me later at Memento design pattern)
- Method clone returns a new instance of a grade

1.4 Group Class

- Group contains 3 filed, the group id, the assigned assistant and a Comparator

1.5 Course Class

- This class connects almost everything
- It contains getters and setters for all the filed and methods like returning all the students, all the grades from a certain course, adding an assistant or a student to a course, instantiating and adding a group
- The class also has an internal CourseBuilder class to access the private fields of Course
- Course has another internal class named Snapshot that keeps a backup of the old grades if requested and also contains 2 methods(getBackup and undo)

- fullCourse and partialCourse are 2 classes that extends Course and create different representations of course(full course and partial course). They also include a getGraduatedStudents

1.6 Observer Design Pattern

- This design pattern consists of 2 interfaces Observer that is implemented by Parent and Subject by Catalog
- The purpose is that parents would get notifications about student's grades(the field student and grade from class Parent would be updated - update method), all trough method notifyObserver

1.7 Strategy Design pattern

- Strategy is an interface that is implemented by 3 classes(BestPartialScore, BestExamScore, BestTotalScore) that contain getBestStudent method based on the chosen strategy

1.8 Visitor Design Pattern

- This hole design pattern is based on ScoreVisitor class that implements Visitor interface
- ScoreVisitor contains more methods in order to create examScores and partialScores dictionary(getTuple returns a list of triplets <Student, String, Double>) from a dictionary field, buildPartialScores and buildExamScores methods that assemble an element in dictionary
- visit methods are the ones that assign grades in Catalog to students and if a grade is not already there, it will be initialized an added to the list of grades;
  these 2 methods are created with the help of getGrades and getStudents methods which return a list of students or grades from a Tuple(because Tuple is a private internal class) from both partialScores and examScores dictionary
- for a more simple approach, I transformed ScoreVisitor in a Singleton class so that there will be only one instance of ScoreVisitor(one version of the dictionaries)

1.9 Memento Design Pattern

- Memento Design Pattern is used to create a backup of the list of grades from Course
- As mentioned at 1.5, in Course there is an internal class named Snapshot that keeps a backup of the old grades if requested
- getBackup method creates a backup for the grades using clone method
- undo method restores the old grades

1.10 Test Class

- In Test there is the static void main method where the catalog.json is being parsed. The information from the .json file is completed successfully into the Catalog
- Next, the examScores and partialScores dictionaries are being completed; to complete the partialScores the .json is parsed numberCourses * numberAssistants times, at the end, after the right course with the right assistant is found, the element is being created and added to the dictionary. In the same way the examScores is being created but the .json file is parsed only numberCourses times and at the end, if it is the right course, the element is being created and added to the dictionary
- After that, the graphic interfaces are being tested, it is given to each of them a parameter by choice, except the Assistant page because I wanted to test the example if both teacher and

assistant are at the same course, how would the Parent page look like. So, for the Parent page I selected a parent with the student enrolled at that same course as at the teacher and the assistant.

- To see the Parent page with the right notification, first, you need to validate the grades from each Assistant and Teacher page, or at least one. When the grades are validated, they will be also printed
- After testing the graphic interfaces, Strategy and Memento are being tested. The output will be printed.

2. Graphic Interface

2.1 Student Page

- This page has a parameter of type Student
- I started by creating the frame and add a border and a background. I continued by adding an appropriate text, which I placed on a panel. I created another panel, but this time was a null panel, to add it to the previous panel so that the text will not be so close to the button I created after. At last, I added to the frame a photo with the Polytechnic logo. That specific button has the action(Action 1) to open another frame that contains a scrollbar with the list of courses the student attends to. The panel was added to the frame. If course is selected and then the button is pressed(a new button that has Action2), a new frame will be created with the information of the student form that specific course. If a course is not selected, other frame will be created, that will have an explanation text with why there is no information. This frame contains a button that when is pressed, the frame will close.

2.2 AssistantTeacher Page

- This page has 2 parameters, a teacher and an assistant, but one of them will always be null. If the non-null parameter is a teacher, then is a Teacher Page, else an Assistant Page.
- This page's first and second frames are the same as the first 2 frames from Student Page, with one small exception, the button for the courses will lead to a frame with a scrollbar with the courses the teacher or the assistant teach. In the second page, if you forget to select a course, a frame will appear with an explanation text with why there is no information. This frame contains a button that when is pressed, the frame will close. When you select a course, another frame will open with all the students that specific teacher or assistant has, along with their grade. On that frame there is a button that when you press it, it will validate all those grades(add them to Catalog) and then print them. This button was done with Action7(for assistant) and Action8(for teacher), both containing the call of the function visit

2.3 Parent Page

- This page has a parameter of type parent
- The first frame is identical with the other first frames(from the other pages), but this time there is no button for seeing courses, there is a button for seeing notification. The notification is taken from the parent's fields and is updated using method notifyObserver, which is called after the validation of the grades in Page Teacher and/or Page Assistant. If there is no notification, a frame will appear with an explanation text with why there is no information. Once again, this frame contains a button that when is pressed, the frame will close.