Champlain College Degree Planner

Authored by: Mason Lee and Abigail Gehlbach

December 4, 2024

CHAMPLAIN COLLEGE

About the Project

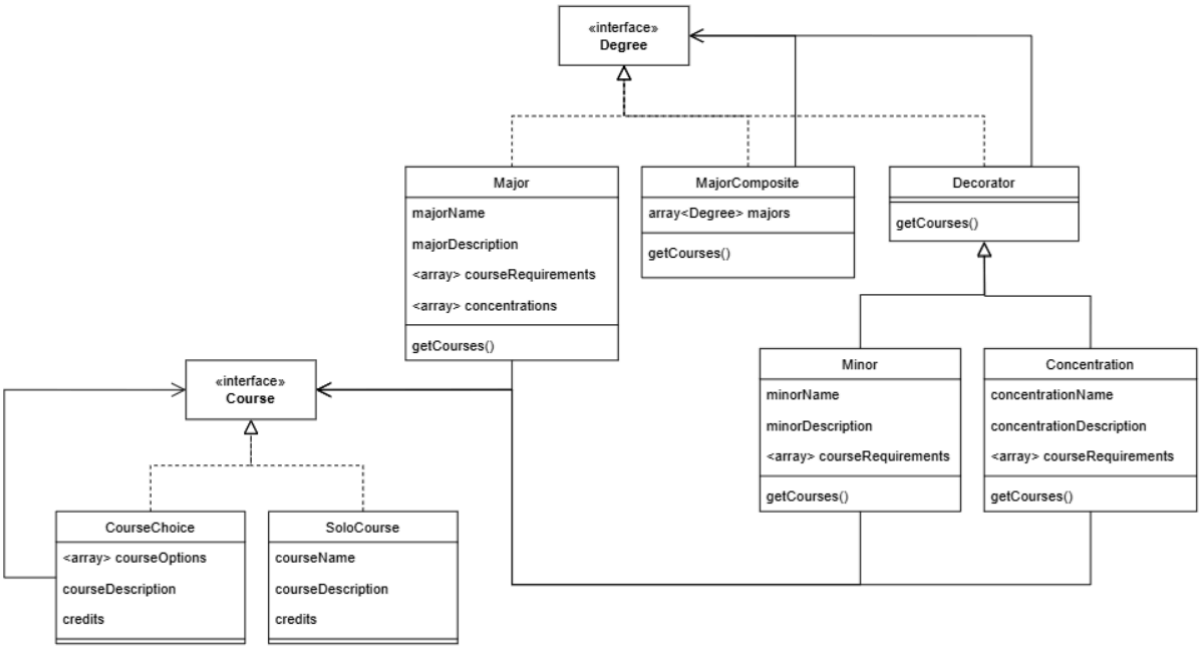
This project is a degree planner for students to explore possible different degrees by adding majors, minors, and concentrations to a degree. The inspiration behind the project is that Champlain does not have a good application to view required courses for majors, minors, and concentrations. We want students who are curious about what they can do with their degree to have an easy way of viewing what a course schedule could look like.

Use Cases

The degree planner could be used if a student wants to add a second major. This application allows them to build a degree in the degree planner to see what classes they would need to take to complete the degree. Similarly, if a student wishes to see if they could fit a minor into their degree, they can add a minor to see what classes they would need to complete the degree.

Let's go through an example of a student who wants to customize their degree. Let's say, the student is a Computer Science major, and they want to add a Creative Media major. Using our program, the student would create a new degree, add a computer science major to the degree, and then add a creative media major to the degree. Using our program they will likely end up noticing that there is not enough overlap between the Computer Science major and the Creative Media major. Maybe this student is determined to have a double major, so they delete the Creative Media major from their degree plan and they add a Math major. They can then see that maybe they can fit the Math major in their degree and still graduate on time! Similarly, the student can add minors and concentrations to their degree to see if they can fit them into their degree. This can be used as a way to explore what classes the student would like to take to see if these degree modifications are a good fit for them.

Design Patterns Used



This project utilized three design patterns: composite, decorator, and strategy. The composite pattern was used twice, once in major composite so that we can wrap two majors in one degree, and once in course choice so that we can wrap multiple solo course options together in a course choice object. The project also uses the decorator design pattern to add majors and concentrations to degrees. Lastly, the strategy pattern was used in courses so that solo course and course choice can be a course added to majors and minors. While the design appears simple when it is all put together, it took a long time to design. It is easy to design a complicated project structure which is not readable and hard to modify; it is difficult to design a project which uses less code and is easy to extend.

Code Excerpts

This is an example of how one could use our code using our test drive file. First, you would create the modifiers to your degree. In this example, the test student wants to have a Computer Science major, with a concentration in Mobile Development, and a Data Science minor. This is how the user creates those modifiers:

```
Major compSci = new Major();
Minor dataScience = new Minor();
Concentration mobileDev = new Concentration();
```

Next, the user would create new courses that they would add to the degree. In this example, the courses that would be required for the major, minor, and concentration are listed below. This is how the user would create new courses:

```
// Create new courses
SoloCourse introProgramming = new SoloCourse("Intro to Programmi
SoloCourse dataViz = new SoloCourse("Data Visualization", "This
SoloCourse designPatterns = new SoloCourse("Software Design Patt
SoloCourse serverSideWeb = new SoloCourse("Server Side Web Devel
SoloCourse refactoring = new SoloCourse("Software Refactoring",
SoloCourse ios = new SoloCourse("iOS App Development", "This cou
SoloCourse android = new SoloCourse("Android App Development", "

// Creating a composite course and adding course choices.
CourseChoice csi3xx = new CourseChoice("CSI-3XX", "Students may
csi3xx.addCourse(designPatterns);
csi3xx.addCourse(serverSideWeb);
csi3xx.addCourse(refactoring);
```

The next step would be to take the courses the user has created, and add them to the degree modifiers. We can see in this example that we are adding intro to programming to both Computer Science and Data Science. Note that even though the course is required for the major and the minor, it will not be counted twice, because when compiling the list of required courses we use a set to remove duplicates.

```
// Adding courses to degree modifiers
compSci.addCourse(introProgramming);
compSci.addCourse(csi3xx);
dataScience.addCourse(dataViz);
dataScience.addCourse(introProgramming);
mobileDev.addCourse(ios);
mobileDev.addCourse(android);
```

Next, we take the degree modifiers that we've built, and add them to the degree. In this example we are simply adding the major, minor, and concentration to our custom degree.

```
// Add degree modifiers to myDegree and print out degree requirements
MajorComposite myDegree = new MajorComposite();
myDegree.addDegreeModifier(compSci);
myDegree.addDegreeModifier(dataScience);
myDegree.addDegreeModifier(mobileDev);
```

Finally, the last step is to call the getCourses method in majorComposite, and loop through the required courses and print them out!

```
System.out.println("Required courses: ");
for(Course course : myDegree.getCourses()) {
    System.out.println(course);
}
```

Conclusion

The degree planner project accomplishes what the group set out to do. It allows a student to create a degree and add majors, minors, and concentrations to that degree. The student can print out all the course requirements for the degree while omitting duplicate courses. We really enjoyed our conversation while designing our UML where we used design language to effectively communicate ideas together. It was surprising to both of us to see that our codebase was fairly small compared to other projects, supporting the idea that good design leads to readable and modifiable code.