

## EECS 2311 – Group 5 Code Smell Fixes

### Code Smell 1

Method: persistence.layer.mainScraper.navigateToCourseList()

Code Smell Type: Long Method

Fix: Extract method


### Description

This method webscrapes courses from the YorkU course website.

In its implementation, towards the very end, it exports the data it webscraped into a JSON file.

The chunk of code that performs the export functionality can be extracted into a new method and used.

### Image of code smell before fix



```
138 WebElement courseCodeTitle = driver2.findElement(By.xpath("/html/body/table/tbody/tr[2]/td[2]/tab
139 WebElement courseDescriptionElement = driver2.findElement(By.xpath("/html/body/table/tbody/tr[2]/t
140
141 String combined = courseCodeTitle.getText();
142 String code = combined.split(regex: "[ ]")[0];
143 String description = courseDescriptionElement.getText();
144 String name = combined.split(regex: "[ ]")[1];
145
146 Course newCourse = new Course(code, name, description, setPrerequisites(description, name));
147 courseList.add(newCourse);
148 }
149 }
150 catch (Exception e)
151 {
152     System.out.println("Error encountered while navigating a course");
153 }
154 }
155
156 for (Course course : courseList)
157 {
158     System.out.println(course.getCode() + " | " + course.getName());
159 }
160 }
161 }
162 GsonBuilder builder = new GsonBuilder();
163 Gson gson = builder.create();
164 try
165 {
166     FileWriter writer = new FileWriter("courses.json");
167     writer.write(gson.toJson(courseList));
168     writer.close();
169 }
170 catch (Exception e)
171 {
172     System.out.println(e.getMessage());
173 }
174
175 driver.close();
176 driver2.close();
177 }
178
179 //Get course list from the database
180 public static ArrayList<Course> getCourseList()
181 {
182     //...
183 }
184
185 public static ArrayList<Course> getCourseListFromJSON()
186 {
187     Gson gson = new Gson();
188     //...
189 }
```

Image of code after fix (Extraction Method)

```

137         driver2.get(courseLink);
138         WebElement courseCodeTitle = driver2.findElement(By.xpath("/html/body/table/tbody/tr[2]/td[2]/table/tbody/tr[2]/td[2]/table/tbody/tr[2]/td[2]"));
139         WebElement courseDescriptionElement = driver2.findElement(By.xpath("/html/body/table/tbody/tr[2]/td[2]/table/tbody/tr[2]/td[2]/table/tbody/tr[2]/td[2]"));
140
141         String combined = courseCodeTitle.getText();
142         String code = combined.split(" ")[0];
143         String description = courseDescriptionElement.getText();
144         String name = combined.split(" ")[1];
145
146         Course newCourse = new Course(code, name, description, setPrerequisites(description, name));
147         courseList.add(newCourse);
148     }
149 }
150 catch (Exception e)
151 {
152     System.out.println("Error encountered while navigating a course");
153 }
154 }
155
156
157 for (Course course : courseList)
158 {
159     System.out.println(course.getCode() + " | " + course.getName());
160 }
161 }
162
163 exportArrayListToJSON(courseList, filename: "courses.json");
164
165 driver.close();
166 driver2.close();
167 }
168
169 private static void exportArrayListToJSON(ArrayList<Course> courses, String filename)
170 {
171     GsonBuilder builder = new GsonBuilder();
172     Gson gson = builder.create();
173     try
174     {
175         FileWriter writer = new FileWriter(filename);
176         writer.write(gson.toJson(courses));
177         writer.close();
178     }
179     catch (Exception e)
180     {
181         System.out.println(e.getMessage());
182     }
183 }

```

## Code Smell 2

Method: presentationlayer.gpaCalculatorController

Code Smell Type: Duplicated Code

Fix: Extract method

Multiple methods, specifically the calculate method, addCourses method, and fillCourses method had several lines where their code was the same which made the methods longer than they needed to be.

To fix this I extracted a method from the duplicate code and made the other methods call it when they needed its functionality. After implementing the fix the methods were much shorter and more readable

Before fix

```
private void CalculateGPA(ActionEvent event){
    try{
        for ( int i=0; i < gradePointFields.size(); i++){
            if (coursesAdded.indexOf(courseNames.get(i).getText()) != -1){
                continue;
            }
            else {
                coursesAdded.add(courseNames.get(i).getText());
                GradePoints = GradePoints +
                    Integer.parseInt(gradePointFields.get(i).getText()) *
                    Integer.parseInt(creditWeightFields.get(i).getText());
                TotalCredits = TotalCredits + Integer.parseInt(creditWeightFields.get(i).getText());
                courses++;
                textAreaCourses.appendText(courses + ". " + courseNames.get(i).getText() + " Weight: " + creditWeightFields.get(i).getText()
                    + " Grade: " + grades.get(i).getValue() + " Point Value: " + gradePoints.get(i).getText() + ".0\n");
            }
            // textAreaCourses.getText().contains(null); use this to check if a course name has already been added
        }
    }
    catch (Exception e) {
```

```

        + " Grade: " + grades.get(i).getValue() + " Point Value: " + gradePoints.get(i).getText() + ".0\n");
    }
    // textAreaCourses.getText().contains(null); use this to check if a course name has already been added

}

}catch (Exception e) {

}

gpa = (double)GradePoints/((double)TotalCredits;
gpa = Math.round(gpa*100.0)/100.0;
textFieldGPA.setText(""+gpa);
textFieldCredits.setText(""+TotalCredits);
}

```

After fix

```

private void CalculateGPA(ActionEvent event){
    int GradePoints=0;
    int TotalCredits=0;
    fillCoursesAdded();
    for (CompletedCourse completedCourse : completed) {
        GradePoints = GradePoints + completedCourse.gradeValue * completedCourse.creditWeight;
        TotalCredits = TotalCredits + completedCourse.creditWeight;
    }
    gpa = (double)GradePoints/((double)TotalCredits;
    gpa = Math.round(gpa*100.0)/100.0;
    textFieldGPA.setText(""+gpa);
    textFieldCredits.setText(""+TotalCredits);
}

```

## Code Smell 3

**Class:** scheduleGenerator.java

**Code Smells:** Duplicate Code, Dead Code

**Fix:** Implemented generateYear() method

- Duplicate Code
  - The code to build the schedule of courses for each year of study is duplicated 4 times (for each year) with very minor changes, it would be better/more efficient to extract this code into a separate method and pass the LinkedHashMap, majorCourses, and year of study as parameters. This way we would eliminate the need to repeat almost the same code 4 times.
- Dead Code
  - The main method in the class is commented out, which can be very confusing to other developers working on the project. This can be removed from the class without affecting the code or project whatsoever.

## Code Before Fix:

```
        case "electrical":
            majorCourses = elecCourses;
            break;
        case "computer":
            majorCourses = compCourses;
            break;
        default:
            break;
    }

    System.out.println("\nFirst Year Courses:");
    firstYear = new ArrayList<>(majorCourses.get(1).size());
    for(int i=0; i<majorCourses.get(1).size(); i++) {
        firstYear.add(majorCourses.get(1).get(i));
        firstYear.removeAll(coursesTaken);
    }
    System.out.println(firstYear);

    System.out.println("\nSecond Year Courses:");
    secondYear = new ArrayList<>(majorCourses.get(2).size());
    for(int i=0; i<majorCourses.get(2).size(); i++) {
        secondYear.add(majorCourses.get(2).get(i));
        secondYear.removeAll(coursesTaken);
    }
    System.out.println(secondYear);

    System.out.println("\nThird Year Courses:");
    thirdYear = new ArrayList<>(majorCourses.get(3).size());
    for(int i=0; i<majorCourses.get(3).size(); i++) {
        thirdYear.add(majorCourses.get(3).get(i));
        thirdYear.removeAll(coursesTaken);
    }
    System.out.println(thirdYear);

    System.out.println("\nFourth Year Courses:");
    fourthYear = new ArrayList<>(majorCourses.get(4).size());
    for(int i=0; i<majorCourses.get(4).size(); i++) {
        fourthYear.add(majorCourses.get(4).get(i));
        fourthYear.removeAll(coursesTaken);
    }
    System.out.println(fourthYear);
}
```

## Code After Fix:

```
switch(major) {
    case "civil":
        majorCourses = civilCourses;
        break;
    case "mechanical":
        majorCourses = mechCourses;
        break;
    case "electrical":
        majorCourses = elecCourses;
        break;
    case "computer":
        majorCourses = compCourses;
        break;
    case "software":
        majorCourses = softCourses;
        break;
    case "space":
        majorCourses = spaceCourses;
        break;
    default:
        break;
}

UserProfile instance = UserProfileInstance.getUserProfile();
ArrayList<Course> profileTaken = instance.getRegularCourses();
coursesTaken.addAll(profileTaken);

firstYear = generateYear(majorCourses, coursesTaken, year 1);
secondYear = generateYear(majorCourses, coursesTaken, year 2);
thirdYear = generateYear(majorCourses, coursesTaken, year 3);
fourthYear = generateYear(majorCourses, coursesTaken, year 4);
}

4 usages  Ali Saheb
public static ArrayList<Course> generateYear(LinkedHashMap<Integer, ArrayList<Course>> majorCourses, ArrayList<Course> coursesTaken, int year) {
    ArrayList<Course> listYear = new ArrayList<>(majorCourses.get(year).size());
    for(int i=0; i<majorCourses.get(year).size(); i++) {
        listYear.add(majorCourses.get(year).get(i));
    }
    listYear.removeAll(coursesTaken);
    return listYear;
}
```

(also removed dead code such as main method and other random lines - not shown in screenshots)

## Code Smell 4

Method: `presentation.layer.userProfileController.addCourseHandler()` and  
`presentation.layer.userProfileController.deleteCourseHandler()`

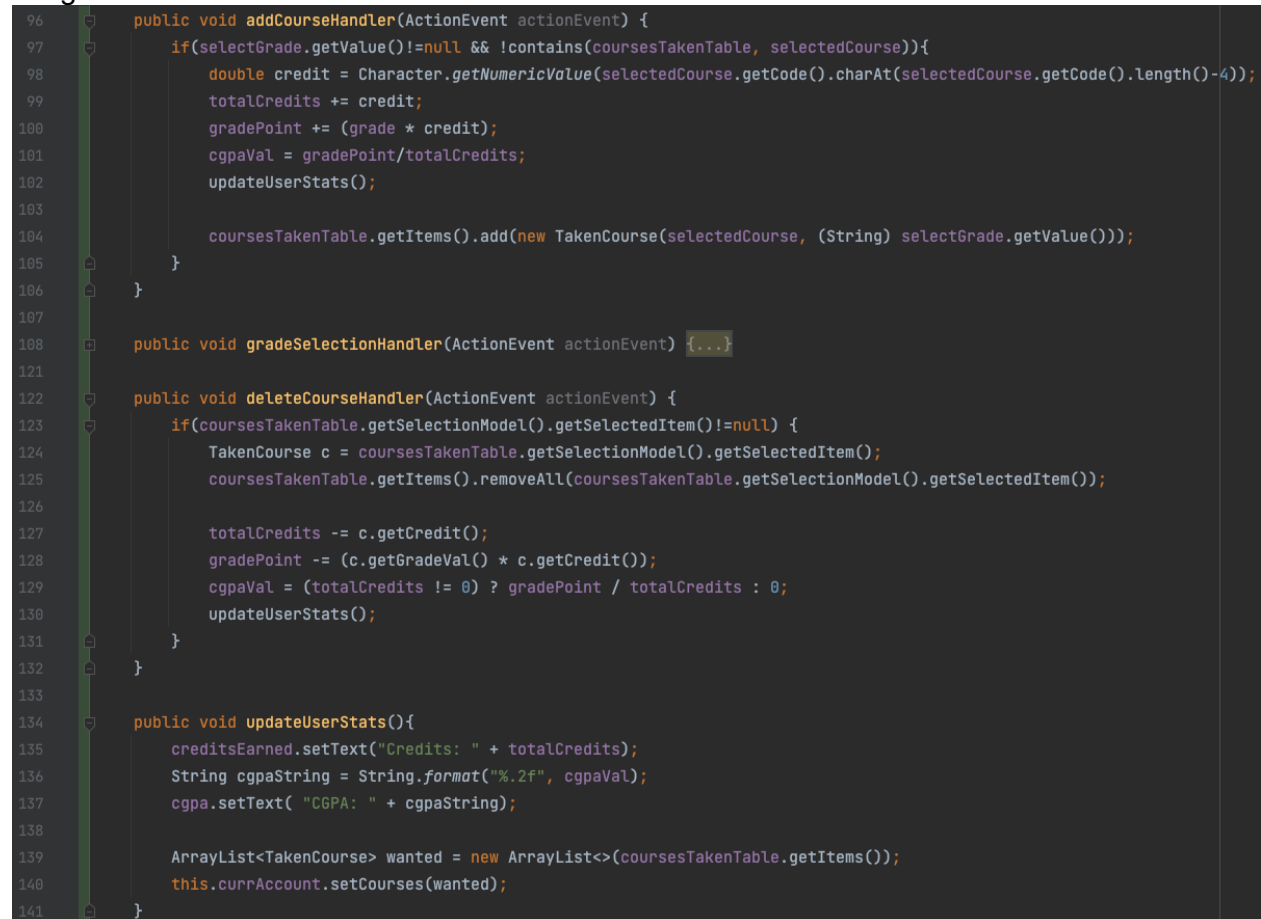
Code Smell Type: Duplicated code

Fix: Extract method

### Description:

These methods are responsible for adding/removing the courses to/from the “Courses Taken” table. We must update the GPA and credit values when a particular course is added or removed. Before the fix, each method updates these values individually. After the fix, the update of values is extracted into `updateUserStats()` which other methods call on addition/deletion of the course.

Image of code smell before the fix:



```
96 public void addCourseHandler(ActionEvent actionEvent) {
97     if(selectGrade.getValue()!=null && !contains(coursesTakenTable, selectedCourse)){
98         double credit = Character.getNumericValue(selectedCourse.getCode().charAt(selectedCourse.getCode().length()-4));
99         totalCredits += credit;
100         gradePoint += (grade * credit);
101         cgpaVal = gradePoint/totalCredits;
102         updateUserStats();
103
104         coursesTakenTable.getItems().add(new TakenCourse(selectedCourse, (String) selectGrade.getValue()));
105     }
106 }
107
108 public void gradeSelectionHandler(ActionEvent actionEvent) {...}
109
110
111
112 public void deleteCourseHandler(ActionEvent actionEvent) {
113     if(coursesTakenTable.getSelectionModel().getSelectedItem()!=null) {
114         TakenCourse c = coursesTakenTable.getSelectionModel().getSelectedItem();
115         coursesTakenTable.getItems().removeAll(coursesTakenTable.getSelectionModel().getSelectedItem());
116
117         totalCredits -= c.getCredit();
118         gradePoint -= (c.getGradeVal() * c.getCredit());
119         cgpaVal = (totalCredits != 0) ? gradePoint / totalCredits : 0;
120         updateUserStats();
121     }
122 }
123
124
125
126
127
128
129
130
131
132
133
134 public void updateUserStats(){
135     creditsEarned.setText("Credits: " + totalCredits);
136     String cgpaString = String.format("%.2f", cgpaVal);
137     cgpa.setText("CGPA: " + cgpaString);
138
139     ArrayList<TakenCourse> wanted = new ArrayList<>(coursesTakenTable.getItems());
140     this.currAccount.setCourses(wanted);
141 }
```



Image of code after the fix (Extraction Method):

```
96     public void addCourseHandler(ActionEvent actionEvent) {
97         if(selectGrade.getValue()!=null && !contains(coursesTakenTable, selectedCourse)){
98             coursesTakenTable.getItems().add(new TakenCourse(selectedCourse, (String) selectGrade.getValue()));
99             updateUserStats();
100         }
101     }
102
103     public void gradeSelectionHandler(ActionEvent actionEvent) {...}
104
105
106
107     public void deleteCourseHandler(ActionEvent actionEvent) {
108         if(coursesTakenTable.getSelectionModel().getSelectedItem()!=null) {
109             coursesTakenTable.getItems().removeAll(coursesTakenTable.getSelectionModel().getSelectedItem());
110             updateUserStats();
111         }
112     }
113
114
115
116
117     public void updateUserStats(){
118         totalCredits = 0;
119         gradePoint = cgpaVal = 0;
120         for(TakenCourse tc:coursesTakenTable.getItems()){
121             totalCredits += tc.getCredit();
122             gradePoint += tc.getGradeVal() * tc.getCredit();
123         }
124         cgpaVal = (totalCredits != 0) ? gradePoint / totalCredits : 0;
125
126         creditsEarned.setText("Credits: " + totalCredits);
127         String cgpaString = String.format("%.2f", cgpaVal);
128         cgpa.setText( "CGPA: " + cgpaString);
129
130         ArrayList<TakenCourse> wanted = new ArrayList<>(coursesTakenTable.getItems());
131         this.currAccount.setCourses(wanted);
132     }
133 }
```