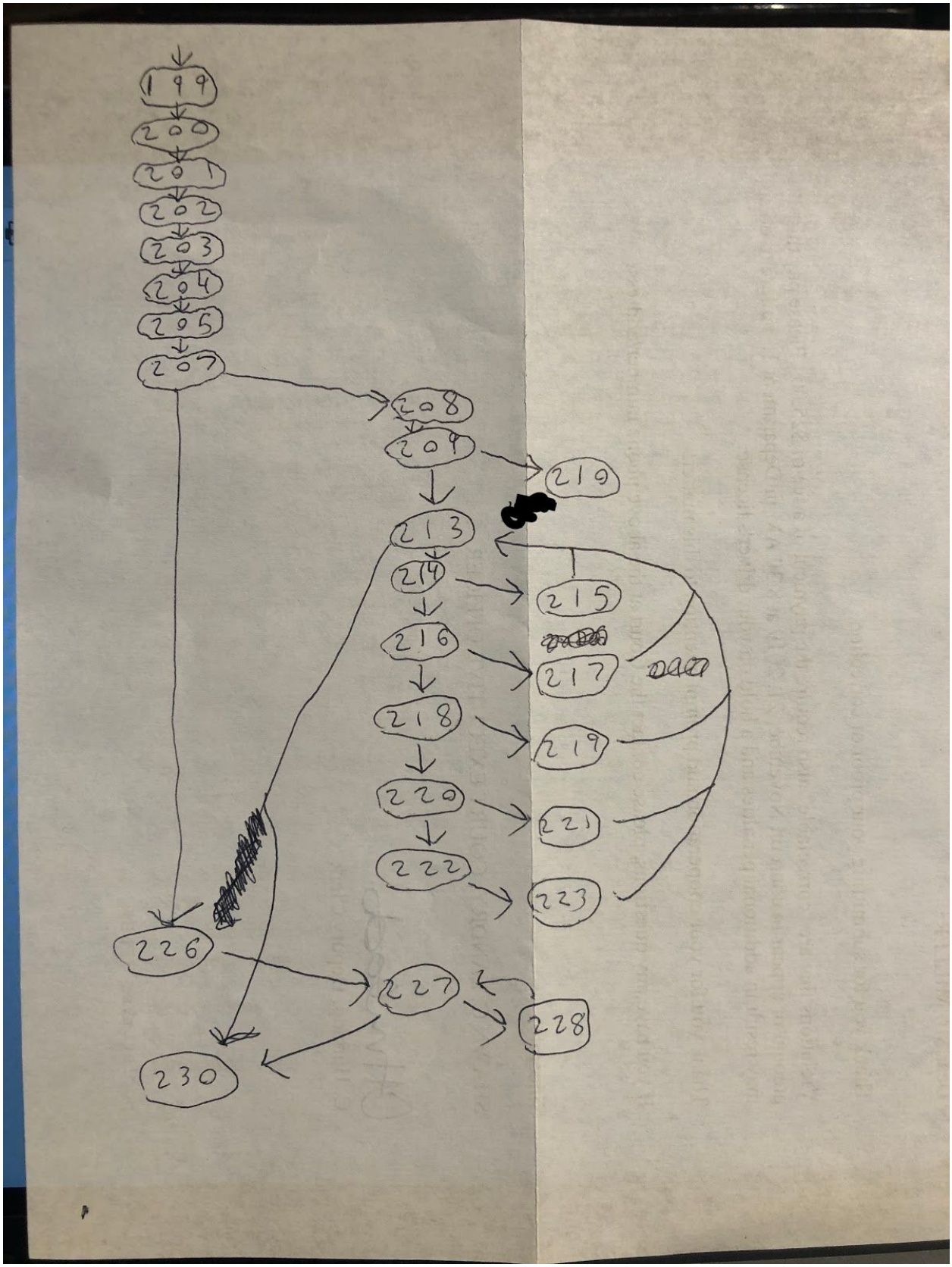
Marcus Miller
SER 316 Assignment 3 Whitebox Testing

Task 3 Step 2.2.b

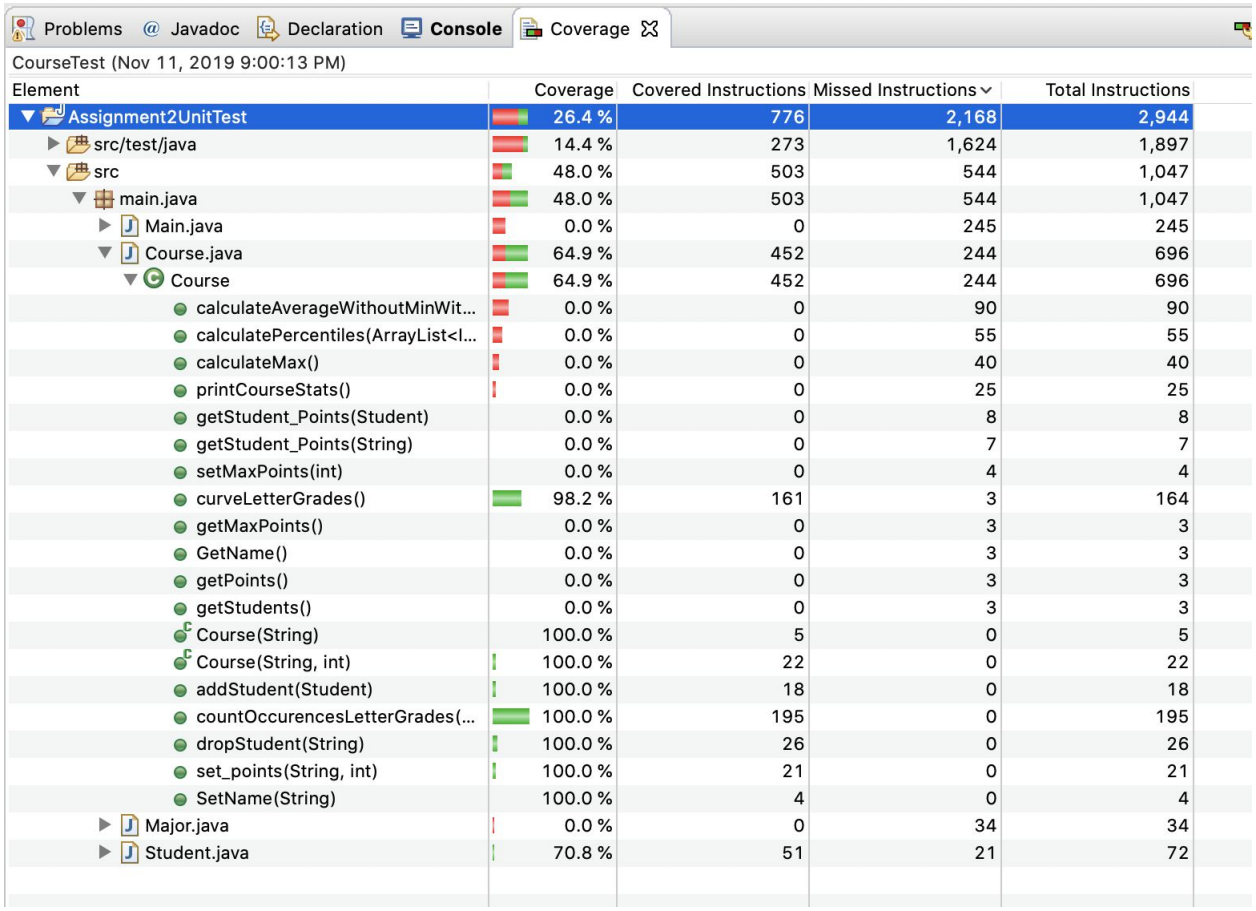



Step 2.2.c.i Node coverage

Test Paths =
[199,200,201,202,203,204,205,207,208,209,213,214,215,213,214,216,217,213,214,216,218,219,213,214,216,218,220,221,213,214,216,218,220,222,223,213,230],
[199,200,201,202,203,204,205,207,226,227,228,227,230],
[199,200,201,202,203,204,205,207,208,209,210]

Step 2.2.c.i Edge coverage
Test Paths =
[199,200,201,202,203,204,205,207,208,209,213,214,215,213,214,216,217,213,214,216,218,219,213,214,216,218,220,221,213,214,216,218,220,222,223,213,230],
[199,200,201,202,203,204,205,207,226,227,228,227,230],
[199,200,201,202,203,204,205,207,208,209,210]

Task 3 Step 2.7.c.i




| Element | Coverage | Covered Instructions | Missed Instructions | Total Instructions |
|---|---|---|---|---|
| Assignment2UnitTest | 26.4 % | 776 | 2,168 | 2,944 |
| src/test/java | 14.4 % | 273 | 1,624 | 1,897 |
| src | 48.0 % | 503 | 544 | 1,047 |
| main.java | 48.0 % | 503 | 544 | 1,047 |
| Main.java | 0.0 % | 0 | 245 | 245 |
| Course.java | 64.9 % | 452 | 244 | 696 |
| Course | 64.9 % | 452 | 244 | 696 |
| calculateAverageWithoutMinWit... | 0.0 % | 0 | 90 | 90 |
| calculatePercentiles(ArrayList<I... | 0.0 % | 0 | 55 | 55 |
| calculateMax() | 0.0 % | 0 | 40 | 40 |
| printCourseStats() | 0.0 % | 0 | 25 | 25 |
| getStudent_Points(Student) | 0.0 % | 0 | 8 | 8 |
| getStudent_Points(String) | 0.0 % | 0 | 7 | 7 |
| setMaxPoints(int) | 0.0 % | 0 | 4 | 4 |
| curveLetterGrades() | 98.2 % | 161 | 3 | 164 |
| getMaxPoints() | 0.0 % | 0 | 3 | 3 |
| GetName() | 0.0 % | 0 | 3 | 3 |
| getPoints() | 0.0 % | 0 | 3 | 3 |
| getStudents() | 0.0 % | 0 | 3 | 3 |
| Course(String) | 100.0 % | 5 | 0 | 5 |
| Course(String, int) | 100.0 % | 22 | 0 | 22 |
| addStudent(Student) | 100.0 % | 18 | 0 | 18 |
| countOccurencesLetterGrades(... | 100.0 % | 195 | 0 | 195 |
| dropStudent(String) | 100.0 % | 26 | 0 | 26 |
| set_points(String, int) | 100.0 % | 21 | 0 | 21 |
| SetName(String) | 100.0 % | 4 | 0 | 4 |
| Major.java | 0.0 % | 0 | 34 | 34 |
| Student.java | 70.8 % | 51 | 21 | 72 |

Task 3 Step 2.7.c.ii
The overall coverage for the program is 26.4%.
Task 3Step 2.7.c.iii
The code coverage for Course.java with your tests from CourseTest.java is 64.9%.

Task 3 Step 2.7.c.v

I was able to reach 100% node coverage on dropStudent, addStudent, and countOccurencesLetterGrades. While only reaching 98.2% node coverage for curveLetterGrades.

```
   */
  public Map<String, String> curveLetterGrades() throws NullPointerException { //TODO verify no side effect with points.
      int curve = 0;
      int OUTOF = 100;
      int maxPoints = Integer.MIN_VALUE;
      int A = 89, B = 79, C = 59, D = 35;
      //Determine curve
      for (Map.Entry<String,Integer> entry : points.entrySet()) {
          if (entry.getValue() >= 0 && entry.getValue() > maxPoints) {
              maxPoints = entry.getValue();
              if (maxPoints >= OUTOF) {
                  curve = 0;
              }
              else {
                  curve = OUTOF - maxPoints;
              }
          }
      }
      //clone points and add curve
      Map<String, Integer> pointsWithCurve = new HashMap<String, Integer>();
      for (Map.Entry<String,Integer> entry : points.entrySet()) {
          pointsWithCurve.put(entry.getKey(), entry.getValue() + curve);
      }
      //convert curvedPoints into <string,string> hashMap
      Map<String, String> gradesWithCurveString = new HashMap<String, String>();
      for (Map.Entry<String,Integer> entry : pointsWithCurve.entrySet()) {
          if (entry.getValue() > A) {
              gradesWithCurveString.put(entry.getKey(), "A");
          }
          else if (entry.getValue() > B) {
              gradesWithCurveString.put(entry.getKey(), "B");
          }
          else if (entry.getValue() > C) {
              gradesWithCurveString.put(entry.getKey(), "C");
          }
          else if (entry.getValue() > D) {
              gradesWithCurveString.put(entry.getKey(), "D");
          }
          else {
              gradesWithCurveString.put(entry.getKey(), "F");
          }
      }
      return gradesWithCurveString;
  }
```

```
    // REACH at least 95% Code Coverage (assign 3)
    // drop a student from course.
    public boolean dropStudent(String asurite) {
        boolean removeFromPoints = points.remove(asurite) != null;
        boolean removeFromStudents = students.remove(new Student(asurite, null));
        /*SER316-start*/
        return removeFromPoints && removeFromStudents;
        //return removeFromPoints == removeFromStudents;
        //We want both points and the student to be removed. The test case above
        //fails if neither were removed.
        /*SER316-start*/
    }
```

```java
    // REACH at least 95% Code coverage  (assign 3)
    // Students should only be added when they are not yet in the course (names (asurite member) needs to be unique)
    ArrayList<Student> students  = new ArrayList<Student>();
    public boolean addStudent(Student s) {
        if(students != null && points.putIfAbsent(s.getAsurite(), -1) == null){
            /*SER316-start*/
            //students.add(s); removed so we don't add a student twice
            /*SER316-end*/
            return students.add(s);

        }
        return false;
    }

public Map<String, String> curveLetterGrades() throws NullPointerException { //TODO verify no side effect with points.
    int curve = 0;
    int OUTOF = 100;
    int maxPoints = Integer.MIN_VALUE;
    int A = 89, B = 79, C = 59, D = 35;
    //Determine curve
    for (Map.Entry<String,Integer> entry : points.entrySet()) {
        if (entry.getValue() >= 0 && entry.getValue() > maxPoints) {
            maxPoints = entry.getValue();
            if (maxPoints >= OUTOF) {
                curve = 0;
            }
            else {
                curve = OUTOF - maxPoints;
            }
        }
    }
    //clone points and add curve
    Map<String, Integer> pointsWithCurve = new HashMap<String, Integer>();
    for (Map.Entry<String,Integer> entry : points.entrySet()) {
        pointsWithCurve.put(entry.getKey(), entry.getValue() + curve);
    }
    //convert curvedPoints into <string,string> hashMap
    Map<String, String> gradesWithCurveString = new HashMap<String, String>();
    for (Map.Entry<String,Integer> entry : pointsWithCurve.entrySet()) {
        if (entry.getValue() > A) {
            gradesWithCurveString.put(entry.getKey(), "A");
        }
        else if (entry.getValue() > B) {
            gradesWithCurveString.put(entry.getKey(), "B");
        }
        else if (entry.getValue() > C) {
            gradesWithCurveString.put(entry.getKey(), "C");
        }
        else if (entry.getValue() > D) {
            gradesWithCurveString.put(entry.getKey(), "D");
        }
        else {
            gradesWithCurveString.put(entry.getKey(), "F");
        }
    }
    return gradesWithCurveString;
}
```

## Task 4 Step 2