CS-567 ADVANCED SOFTWARE ASSURANCE PROJECT

Jai Venkata Arun Akula

Northern Arizona University

Email: Ja3726@nau.edu

INTRODUCTION

This is a **School Management System** in Java, which has capability to manage students, teachers, courses, and semesters. It enables the administrators to define and maintain organizational units like students, teachers, courses and semesters; and perform activities including registering students for courses as well as allocating courses to teachers. The system provides tracking of course enrollments and timely grades for students by individual semesters and also form teachers' grade entry. There is the student enrollment for the courses that he or she has taken as well as the grade of the course They are given; and on the other hand, teachers can see the courses they have been given to manage. Another related function is also provided by the system; admission or withdrawal from a course. It acts as a basic framework for handling records and work processes in school institutions.

TOOLS AND DEPENDENCIES USED

JAVA JRE 23, JDK 23; Apache Maven; Chocolatey; Junit Jupiter 5; PIT plugins and Randoop.

My project runs on Java JRE 23 and JDK 23, which supports compatibility as the development tool and assured operational efficiency. Achieving automation and dependency management is via Apache Maven other tools like Chocolatey to measure the code line. JUnit Jupiter 5 is used for unit testing and PIT plugins and Randoop used for mutation and random testing.

IDE USED

IntelliJ IDEA 2024.

I used IntelliJ IDEA 2024 as the development environment of choice for this project, it includes strong coding support and overall code indexing system. Its exceptional capabilities of debugging and integration improve on the productivity and efficiency of the development procedure. It also provides a feature of seamless integration with Maven, JUnit and other tools used in the respective project.

PROJECT STRUCTURE OVERVIEW

The SchoolManagementSystem project reflects the principles of modularity for working with students, teachers, courses, enrollments, and semesters in a school context. Here is a list with the major classes and methods found in more detail in the official documentation of the framework.

1. Main Application Class

- **Purpose:** This class serves as the entry point for the system. It initializes the SchoolManagementSystem and communicate with various components through test cases.
- **Main Methods:**
  - public static void main(String[] args): Setting up the overall framework of the system accompanied by the instant sample data input, enrolment, grading and so on, data output.

2. Student Class

- **Purpose:** This class will represent a student with personal details of students and enrollments in courses.
- **Main Attributes:**
  - String name, int age, String studentId, List<Enrollment> enrollments.
- **Main Methods:**
  - enroll (Course course, String semester); drop Course (Course course, String semester); display Courses();display Grades ().

3. Teacher Class

- **Purpose:** This class represents a teacher responsible for teaching courses and assigning grades.
- **Main Attributes:**
  - String name, String teacherId, List<Course> courses
- **Main Methods:**
  - assignCourse(Course course); displayCourses(); assignGrade(Student student, Course course, String semester, double grade)

4. Course Class

- **Purpose:** This class represents a course offered by the school, including teacher and enrolled students.
- **Main Attributes:**
  - String courseName, String courseCode, Teacher teacher, Map<String, List<Student>> enrolledStudentsPerSemester
- **Main Methods:**
  - setTeacher(Teacher teacher); addStudent(Student student, String semester); removeStudent(Student student, String semester); displayStudents(String semester).

5. Enrollment Class

- **Purpose:** This class represents the enrollment of a student in a specific course for a semester.
- **Main Attributes:**
  - Course course, String semester, Double grade.
- **Main Methods:**
  - getGrade(), setGrade(Double grade); getCourse(), getSemester().

6. Semester Class

- **Purpose:** This class represents a specific semester and its associated courses.
- **Main Attributes:**
  - String semesterName, Map<String, Course> courses
- **Main Methods:**
  - addCourse(Course course); displayCourses().

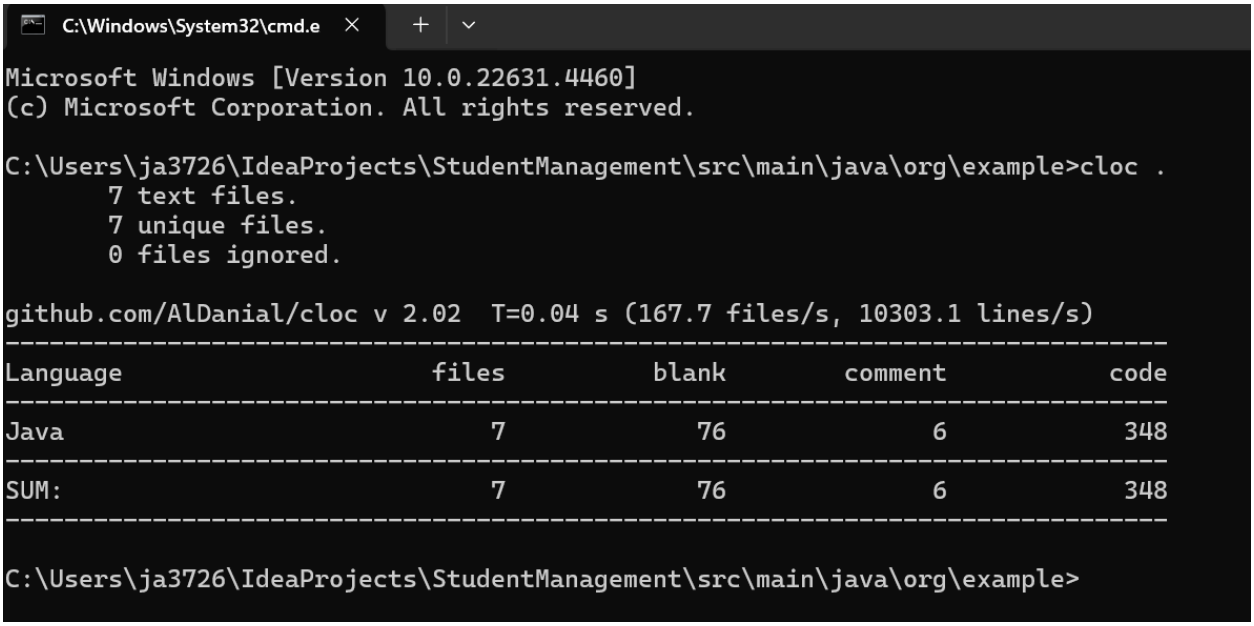
7. SchoolManagementSystem Class

- **Purpose:** This class serves as the core controller to manage students, teachers, courses, semesters, and their interactions.
- **Main Attributes:**
  - Map<String, Student> students, Map<String, Teacher> teachers, Map<String, Course> courses, Map<String, Semester> semesters
- **Main Methods:**
  - addStudent(String name, int age, String studentId); addTeacher(String name, String teacherId); addCourse(String courseName, String courseCode); addSemester(String semesterName); assignTeacherToCourse(String teacherId, String courseCode); enrollStudentInCourse(String studentId, String courseCode, String semester).

CODE MEASURING

Command used: Cloc .

This command was ran in the directory where all the files of the class are present.



The code I chose have 348 lines of actual code.

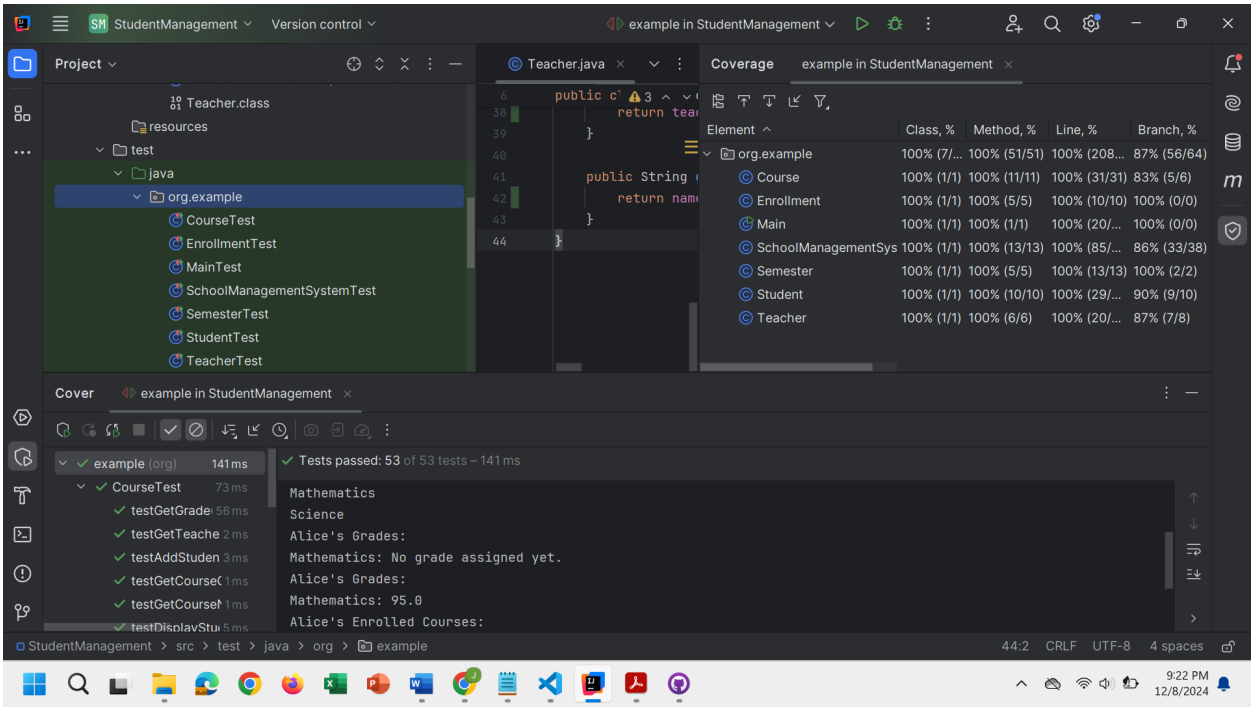
BASIC TEST

Command Used: javac Main.java

```
"C:\Program Files\Java\jdk-23\bin\java.exe" "-javaagent:C:\Users\ja3726\AppData\Local\JetBrains\IntelliJ IDEA 2024.3\lib\idea_rt.jar=53398:
Student added: Alice
Student added: Bob
Teacher added: Dr. Smith
Teacher added: Prof. Johnson
Course added: Mathematics
Course added: Science
Semester added: Fall 2024
Semester added: Spring 2025
Dr. Smith assigned to Mathematics
Prof. Johnson assigned to Science
Alice enrolled in Mathematics for Fall 2024
Bob enrolled in Science for Fall 2024
Grade for Alice in Mathematics (Fall 2024) set to 90.0
Grade for Bob in Science (Fall 2024) set to 85.0
Alice's Enrolled Courses:
Mathematics (Fall 2024)
Dr. Smith's Courses:
Mathematics
Students enrolled in Mathematics (Fall 2024):
Alice
Alice
```

The basic code run will give the output of my SchoolManagementystem with data of the students, teachers and their grades.

CODE COVERAGE REPORT WITH JUNITS



MUTATION TESTING

Mutational testing determines the efficiency of the unit test by applying changes (mutant) to your code in form of bugs and then seeing if the tests will identify them. Mutation analysis is a process of making changes to the original code then running a test suite and observing the results Changes that have failed after a mutation are effective Changes that passed indicate that there is something wrong with the test suite.

In this project, I used a tool called PITtest to perform mutation testing. Here's how it's set up and applied:

Configuring Pitest:

In order to include the PITtest plugin in the pom.xml file it is added as a dependency. This makes it easier to be integrated with Maven which is the build tool used in this project.

Running Mutation Tests:

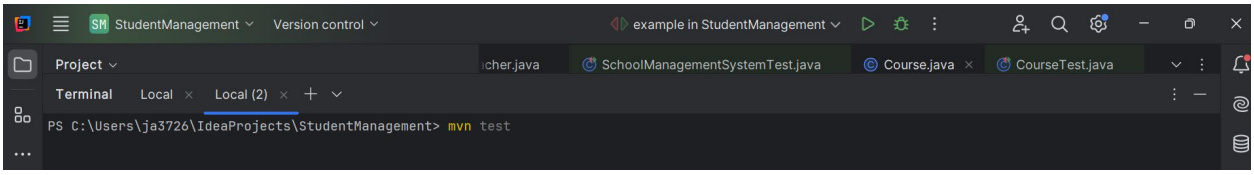
Once the unit tests are ready, I run the mutation tests using the command mvn pitest:mutationCoverage. Pitest then adjusted the code (mutations) and reran the tests to determine if they can detect the changes I have made.

Reviewing the Results:

This test generated a pit report which will show the methods covered, lines covered, the mutation coverage and overall test score.

Command Used: mvn test; mvn clean install

This command will download all the dependencies and plugins that are required for the tests.



Running Pit tests

```
stderr  : 9:27:13?PM PIT >> FINE : Mutation MutationIdentifier [location=Location [clazz=org.example.Main, method=main, methodDesc=([Ljava/lang/Stri
ng;)V], indexes=[103], mutator=org.pitest.mutationtest.engine.gregor.mutators.VoidMethodCallMutator] detected = SURVIVED
stderr  : 9:27:13?PM PIT >> FINE : processed mutation in 35 ms.
stderr  : 9:27:13?PM PIT >> FINE : Running mutation MutationDetails [id=MutationIdentifier [location=Location [clazz=org.example.Main, method=main,
methodDesc=([Ljava/lang/String;)V], indexes=[13], mutator=org.pitest.mutationtest.engine.gregor.mutators.VoidMethodCallMutator], filename=Main.java,
 block=[1], lineNumber=8, description=removed call to org/example/SchoolManagementSystem::addStudent, testsInOrder=[org.example.MainTest.[engine:jun
it-jupiter]/[class:org.example.MainTest]/[method:testMainMethod()]]]
stderr  : 9:27:13?PM PIT >> FINE : mutating method main
stderr  : 9:27:13?PM PIT >> FINE : 1 relevant test for main
stderr  : 9:27:13?PM PIT >> FINE : replaced class with mutant in 19 ms
stderr  : 9:27:13?PM PIT >> FINE : Running 1 units
9:27:13?PM PIT >> FINE : MutationIdentifier [location=Location [clazz=org.example.Main, method=main, methodDesc=([Ljava/lang/String;)V], indexes=[13
], mutator=org.pitest.mutationtest.engine.gregor.mutators.VoidMethodCallMutator] KILLED by [org.example.MainTest.[engine:junit-jupiter]/[class:org.e
xample.MainTest]/[method:testMainMethod()]]
stderr  : 9:27:13?PM PIT >> FINE : Mutation MutationIdentifier [location=Location [clazz=org.example.Main, method=main, methodDesc=([Ljava/lang/Stri
ng;)V], indexes=[13], mutator=org.pitest.mutationtest.engine.gregor.mutators.VoidMethodCallMutator] detected = KILLED by [org.example.MainTest.[engi
ne:junit-jupiter]/[class:org.example.MainTest]/[method:testMainMethod()]]
stderr  : 9:27:13?PM PIT >> FINE : processed mutation in 39 ms.
stderr  : 9:27:13?PM PIT >> FINE : Running mutation MutationDetails [id=MutationIdentifier [location=Location [clazz=org.example.Main, method=main,
methodDesc=([Ljava/lang/String;)V], indexes=[44], mutator=org.pitest.mutationtest.engine.gregor.mutators.VoidMethodCallMutator], filename=Main.java,
 block=[6], lineNumber=17, description=removed call to org/example/SchoolManagementSystem::addCourse, testsInOrder=[org.example.MainTest.[engine:jun
it-jupiter]/[class:org.example.MainTest]/[method:testMainMethod()]]]
stderr  : 9:27:13?PM PIT >> FINE : mutating method main
stderr  : 9:27:13?PM PIT >> FINE : 1 relevant test for main
dentManagement > src > main > java > org > example > © Course                    8:14  CRLF  UTF-8  4 spaces
```

```
stderr  : 9:27:27?PM PIT >> FINE : mutating method getGrade
stderr  : 9:27:27?PM PIT >> FINE : 12 relevant test for getGrade
stderr  : 9:27:27?PM PIT >> FINE : replaced class with mutant in 25 ms
stderr  : 9:27:27?PM PIT >> FINE : Running 1 units
stderr  : 9:27:27?PM PIT >> FINE : Mutation MutationIdentifier [location=Location [clazz=org.example.Enrollment, method=getGrade, methodDesc=()Ljava
/lang/Double;], indexes=[5], mutator=org.pitest.mutationtest.engine.gregor.mutators.returns.EmptyObjectReturnValsMutator] detected = KILLED by [org.
example.EnrollmentTest.[engine:junit-jupiter]/[class:org.example.EnrollmentTest]/[method:testGetGrade_initiallyNull()]]
stderr  : 9:27:27?PM PIT >> FINE : processed mutation in 46 ms.9:27:27?PM PIT >> FINE : MutationIdentifier [location=Location [clazz=org.example.Enr
ollment, method=getGrade, methodDesc=()Ljava/lang/Double;], indexes=[5], mutator=org.pitest.mutationtest.engine.gregor.mutators.returns.EmptyObjectR
eturnValsMutator] KILLED by [org.example.EnrollmentTest.[engine:junit-jupiter]/[class:org.example.EnrollmentTest]/[method:testGetGrade_initiallyNull
()]]

9:27:27?PM PIT >> FINE : Exit code was - OK
9:27:27?PM PIT >> FINE : Minion exited ok
dentManagement > src > main > java > org > example > © Course                    8:14  CRLF  UTF-8  4 spaces
```

```
================================================================================
> pre-scan for mutations : < 1 second
> scan classpath : < 1 second
> coverage and dependency analysis : 10 seconds
> build mutation tests : < 1 second
> run mutation analysis : 33 seconds
--------------------------------------------------------------------------------
> Total  : 44 seconds
--------------------------------------------------------------------------------
================================================================================
- Statistics
================================================================================
>> Line Coverage (for mutated classes only): 208/209 (100%)
>> Generated 107 mutations Killed 75 (70%)
>> Mutations with no coverage 0. Test strength 70%
>> Ran 4715 tests (44.07 tests per mutation)
Enhanced functionality available at https://www.arcmutate.com/
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time:  53.477 s
[INFO] Finished at: 2024-12-08T21:27:27-07:00
[INFO] ------------------------------------------------------------------------
PS C:\Users\ja3726\IdeaProjects\StudentManagement>
udentManagement > src > main > java > org > example > © Course                    8:14  CRLF  UTF-8  4 spaces
```

Pit report

Pit Test Coverage Report

Project Summary

Number of Classes	Line Coverage		Mutation Coverage		Test Strength	
7	100%	208/209	70%	75/107	70%	75/107

Breakdown by Package

Name	Number of Classes	Line Coverage		Mutation Coverage		Test Strength	
org.example	7	100%	208/209	70%	75/107	70%	75/107

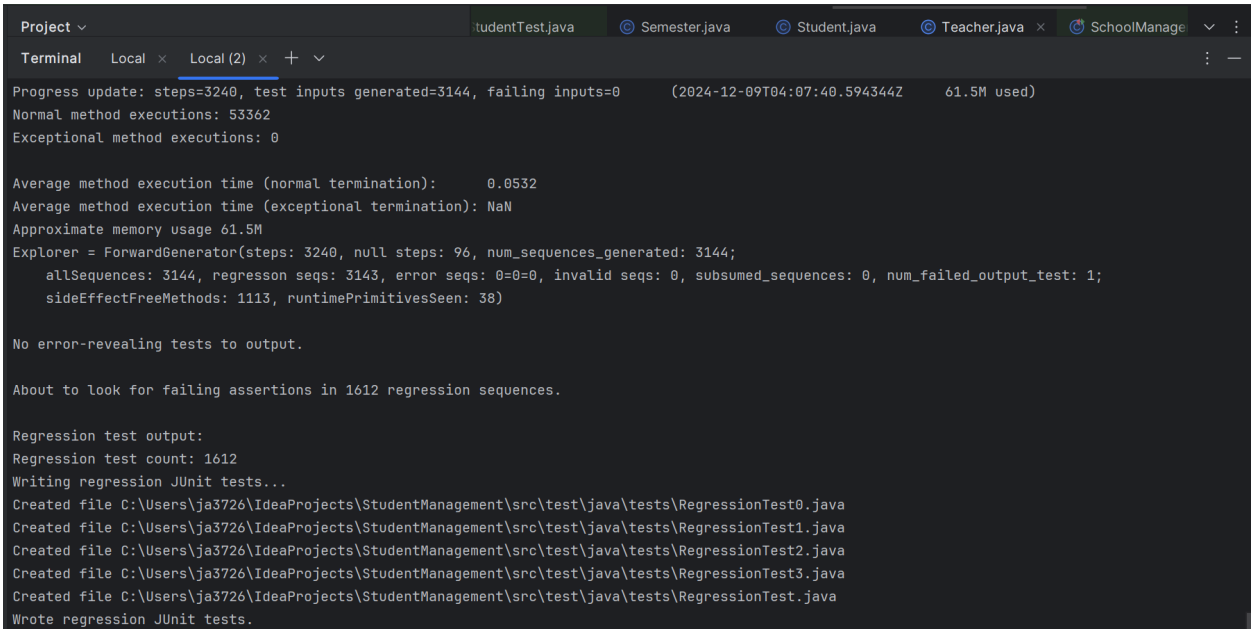
RANDOM TEST CASES GENERATION

Tool Used: Randoop

Command used:

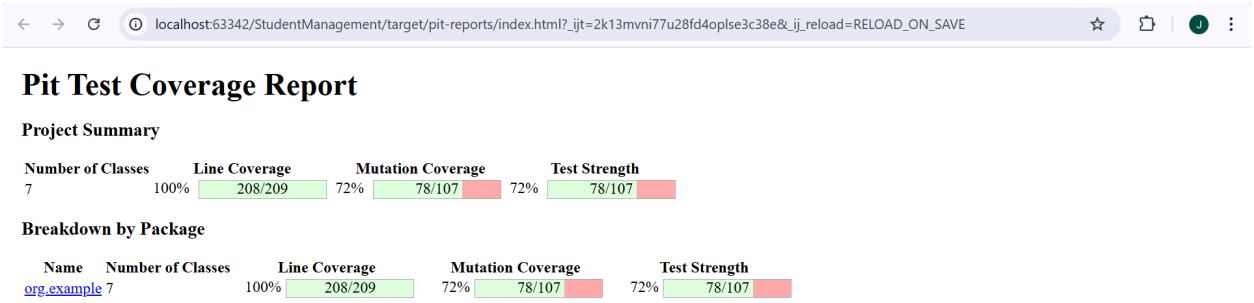
```
PS C:\Users\ja3726\IdeaProjects\StudentManagement> java -cp "C:\Users\ja3726\IdeaProjects\StudentManagement\randoop-all-4.3.3.jar;C:\Users\ja3726\IdeaProjects\StudentManagement\target\classes" randoop.main.Main gentests --testclass=org.example.SchoolManagementSystem --junit-output-dir=src/test/java/tests
```

This command will generate tests as regression and following is the output:



Total 3144 regression test have been generated and it ran for approx 60sec. This test has executed 53,362 methods in total. These tests have used 61.5MB of memory. This test has covered 100% of the lines with mutation coverage of 72%. The Test Strength I got was 72% after the regression testing with Junits. The test score was same all the time after running the tests for multiple times.

Then I ran mvn clean install to generate the pit report.



After implementing regression testing the mutation coverage was increased to 72% from initial 70%. This shows that the automation testing plays an important role in testing industry to improve the quality of the projects and significantly help in increasing the quality of the product.

My project: Software Assurance Project