MODULE 3-2 MILESTONE TWO

ENHANCEMENT ONE: SOFTWARE DESIGN AND ENGINEERING

CS-499 Computer Science Capstone

Zhraa Al-Bayaa

September 28, 2024

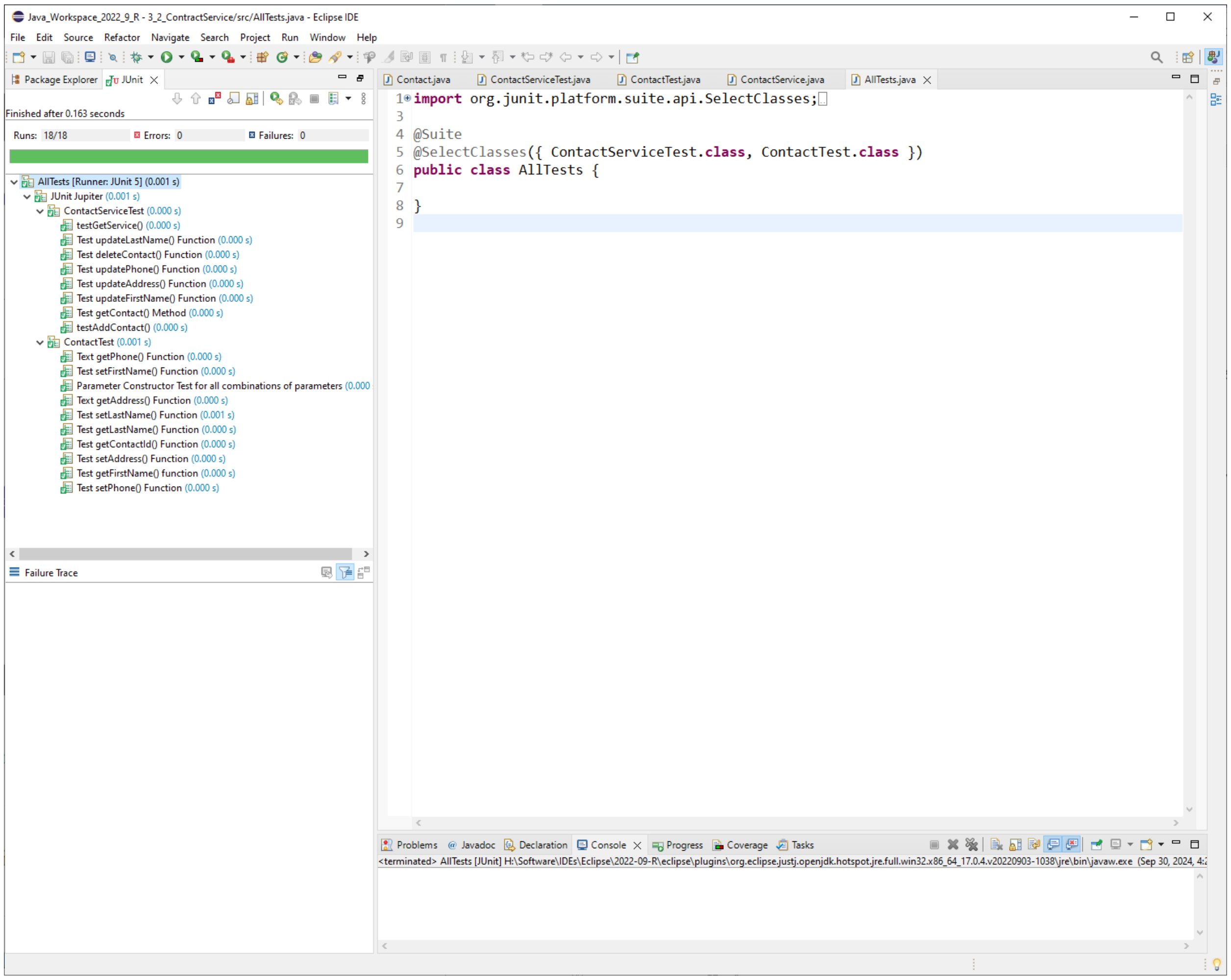
This paper will present the enhancements for artifact one related to the Software Design and Engineering category for the Computer Science Capstone course. The paper entails the details of the artifact chosen for this milestone during the module 2 code review project and is included in the ePortfolio. It also covers the details of the processes and steps taken to create and enhance the artifact. The outcome of this enhancement process will display the outcomes and learnings acquired while completing the enhancements.

Prompt

The artifact selected for this software design and engineering category is related to a Project completed during a course taken at (Southern New Hampshire University) CS320. This project was related to creating a Java-based Contact class along with a repository class to provide crude operations to manage the in-memory contact repository and service. The project also involved creating unit tests for both these classes Contact.java and ContactService.java. The application intends to manage contacts and uses Java HashMap for an efficient in-memory data repository and provides robust unit tests to validate the data input and intended functionality to improve the quality assurance of the artifact. To create this application, java's JDK version 18 was used in combination with a famous Eclipse IDE. The Junit Test version 5 is used to test the boundary conditions, functionality, and data input validations. The Entity classes is used following the best coding standards using data encapsulation, constants, and robust functions to create and use Contact objects. The repository class used HashMap which is one of Java's fastest collection frameworks with efficient insertion, retrieval, and deletion procedure runtimes.

The selected artifact mainly consists of two Java classes and their associated Junit tests to validate their functionality. This application doesn't involve user interaction which is why there is not Text-based Interfaced or Graphical User Interface. The artifact was completed with best Java Programming practices for secure coding and following the standard Java Convections and naming schemes. The application is developed following the Software Design and development frameworks based on requirement analysis concluded from the given problem statement and requirements during the course module. The analysis is transformed into the Java Pojo class and its associated collection or repository class that exposes the functionality to manage the contact within the computer memory.

Screenshot of Test run after enhanced to Test Suite:



```
Finished after 0.163 seconds

Runs: 18/18    Errors: 0    Failures: 0

AllTests [Runner: JUnit 5] (0.001 s)
  JUnit Jupiter (0.001 s)
    ContactServiceTest (0.000 s)
      testGetService() (0.000 s)
      Test updateLastName() Function (0.000 s)
      Test deleteContact() Function (0.000 s)
      Test updatePhone() Function (0.000 s)
      Test updateAddress() Function (0.000 s)
      Test updateFirstName() Function (0.000 s)
      Test getContact() Method (0.000 s)
      testAddContact() (0.000 s)
    ContactTest (0.001 s)
      Text getPhone() Function (0.000 s)
      Test setFirstName() Function (0.000 s)
      Parameter Constructor Test for all combinations of parameters (0.000
      Text getAddress() Function (0.000 s)
      Test setLastName() Function (0.001 s)
      Test getLastName() Function (0.000 s)
      Test getContactId() Function (0.000 s)
      Test setAddress() Function (0.000 s)
      Test getFirstName() function (0.000 s)
      Test setPhone() Function (0.000 s)
```

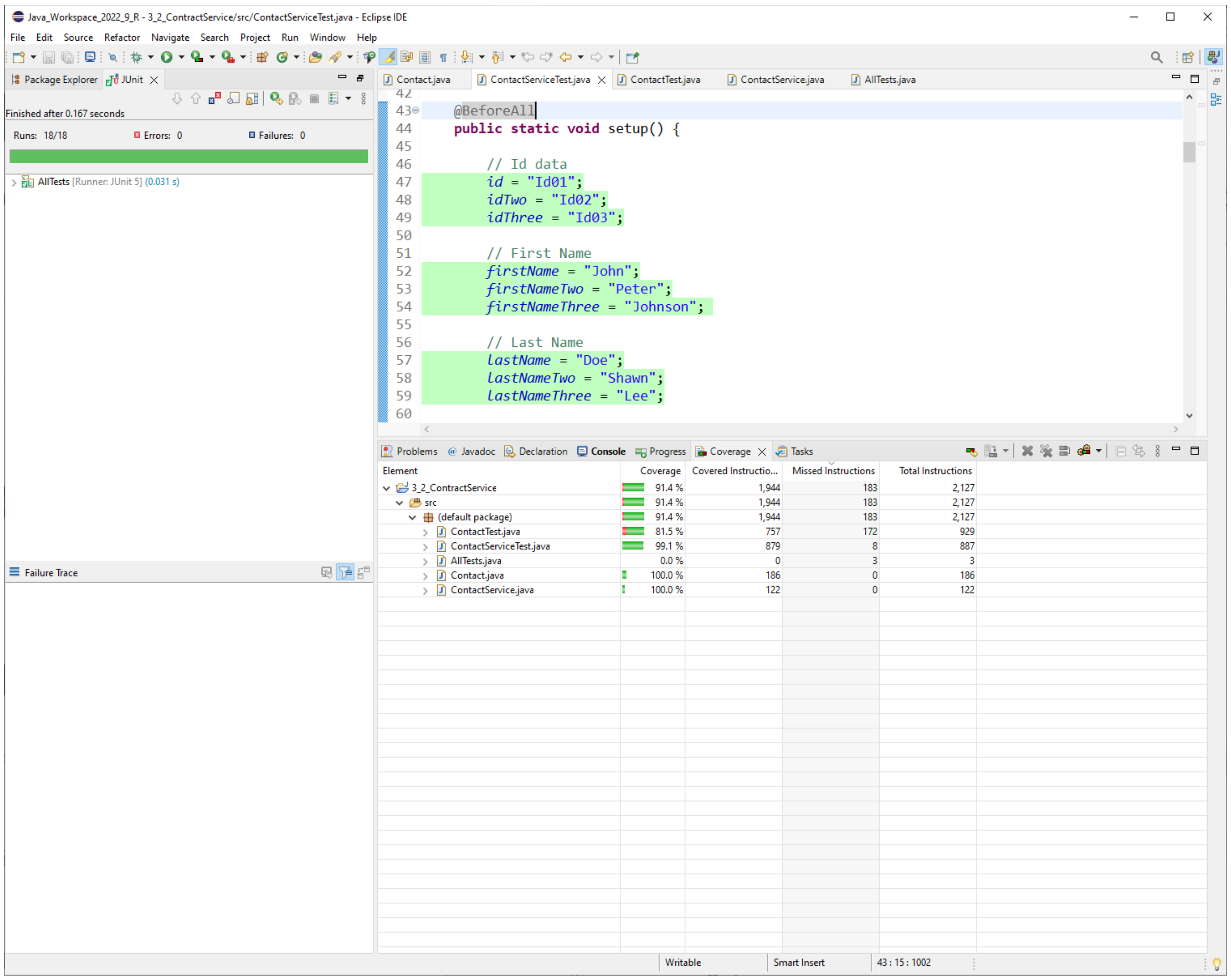
```java
import org.junit.platform.suite.api.SelectClasses;

@Suite
@SelectClasses({ ContactServiceTest.class, ContactTest.class })
public class AllTests {

}
```

```
<terminated> AllTests [JUnit] H:\Software\IDEs\Eclipse\2022-09-R\eclipse\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_17.0.4.v20220903-1038\jre\bin\javaw.exe (Sep 30, 2024, 4:2
```

Code Coverage



```
Finished after 0.167 seconds
Runs: 18/18     Errors: 0     Failures: 0
AllTests [Runner: JUnit 5] (0.031 s)
```

```java
    @BeforeAll
    public static void setup() {

        // Id data
        id = "Id01";
        idTwo = "Id02";
        idThree = "Id03";

        // First Name
        firstName = "John";
        firstNameTwo = "Peter";
        firstNameThree = "Johnson";

        // Last Name
        lastName = "Doe";
        lastNameTwo = "Shawn";
        lastNameThree = "Lee";

```

Element	Coverage	Covered Instructio...	Missed Instructions	Total Instructions
3_2_ContractService	91.4 %	1,944	183	2,127
src	91.4 %	1,944	183	2,127
(default package)	91.4 %	1,944	183	2,127
ContactTest.java	81.5 %	757	172	929
ContactServiceTest.java	99.1 %	879	8	887
AllTests.java	0.0 %	0	3	3
Contact.java	100.0 %	186	0	186
ContactService.java	100.0 %	122	0	122

The creation and enhancement of this application showcase the skills I learned during this CS320 course to learn about software development and quality assurance by creating automated unit tests to test and validate the functionality and correctness of the application. Dynamic application testing offers various advantages to achieve the robustness, correctness, performance, efficiency, and improved quality assurance of the application being developed. Unit testing and quality assurance bring many competitive advantages to the software applications that undergo rigorous testing such as saving maintenance costs, consistent performance as expected, enhanced brand reputation, improved efficiency and operational performance, reduced bugs, and achieving compliance with the regulatory bodies. Testing functionality and boundary conditions through Unit testing ensure that an application will always produce the intended results for a given dataset.

During the development and enhancement of this artifact, I followed the best coding standards and programming practices while closely adhering to the software design and development phases. I used an efficient data structure to manage the contact and wrapped it into a repository class classed ContactService. Java's HashMap is one of its fastest data structures concerning its efficiency it has time and space complexity of O (N) and operations O (1) times at average and O(N) in the worst-case scenario. These runtimes prove that the chosen data structure is fast enough to service the requested CRUD operating for this in-memory contact repository.