**Faculty of Computing**

**SE-314: Software Construction**

**Class: BESE 13-A**

# Lab 08: Abstract Data Type - I

**CLO-03:** Design and develop solutions based on Software Construction principles.  
**CLO-04:** Use modern tools such as Eclipse, NetBeans etc. for software construction.

**Date: 11th Nov 2024**

**Instructor: Dr. Mehvish Rashid  
Lab Engineer: Mr. Aftab Farooq**

|  |  |
| --- | --- |
| NAME | MUHAMMAD SAAD UMER |
| CLASS | BESE 13-A |
| CMS | 408485 |

# Lab 08: ADT

# Introduction:

Students will have hands-on experience on designing, testing, and implementing abstract data types. Given a set of specifications, you will write unit tests that check for compliance with the specifications, and then implement code that meets the specifications.

**Material:**

https://ocw.mit.edu/ans7870/6/6.005/s16/psets/ps2/

## Lab Tasks

Solve problem 1-2 of problem set 2 listed on the link. The type we will implement is **Graph<L>:** an abstract data type for mutable weighted directed graphs with labeled vertices.

**Task1:**

Devise, document, and implement tests for Graph<String> .

For now, we’ll only test (and then implement) graphs with String vertex labels. Later, we’ll expand to other kinds of labels.

In order to accommodate running our tests on *multiple implementations* of the Graph

interface, here is the setup:

* The testing strategy and tests for the **static** Graph.empty() method are in **GraphStaticTest.java** . Since the method is static, there will be only one implementation, and we only need to run these tests once. We’ve provided these tests. You are free to change or add to them, but you can leave them as-is for this problem.
* Write your testing strategy and your tests for all the **instance** methods in **GraphInstanceTest.java** . In these tests, you must use the emptyInstance() method to get fresh empty graphs, *not Graph.empty() !* See the provided testInitialVerticesEmpty() for an example.

**Task2:**

Implement weighted directed graphs with String labels — **twice** . Check further details on the link. And provide the following for all the classes in this task:

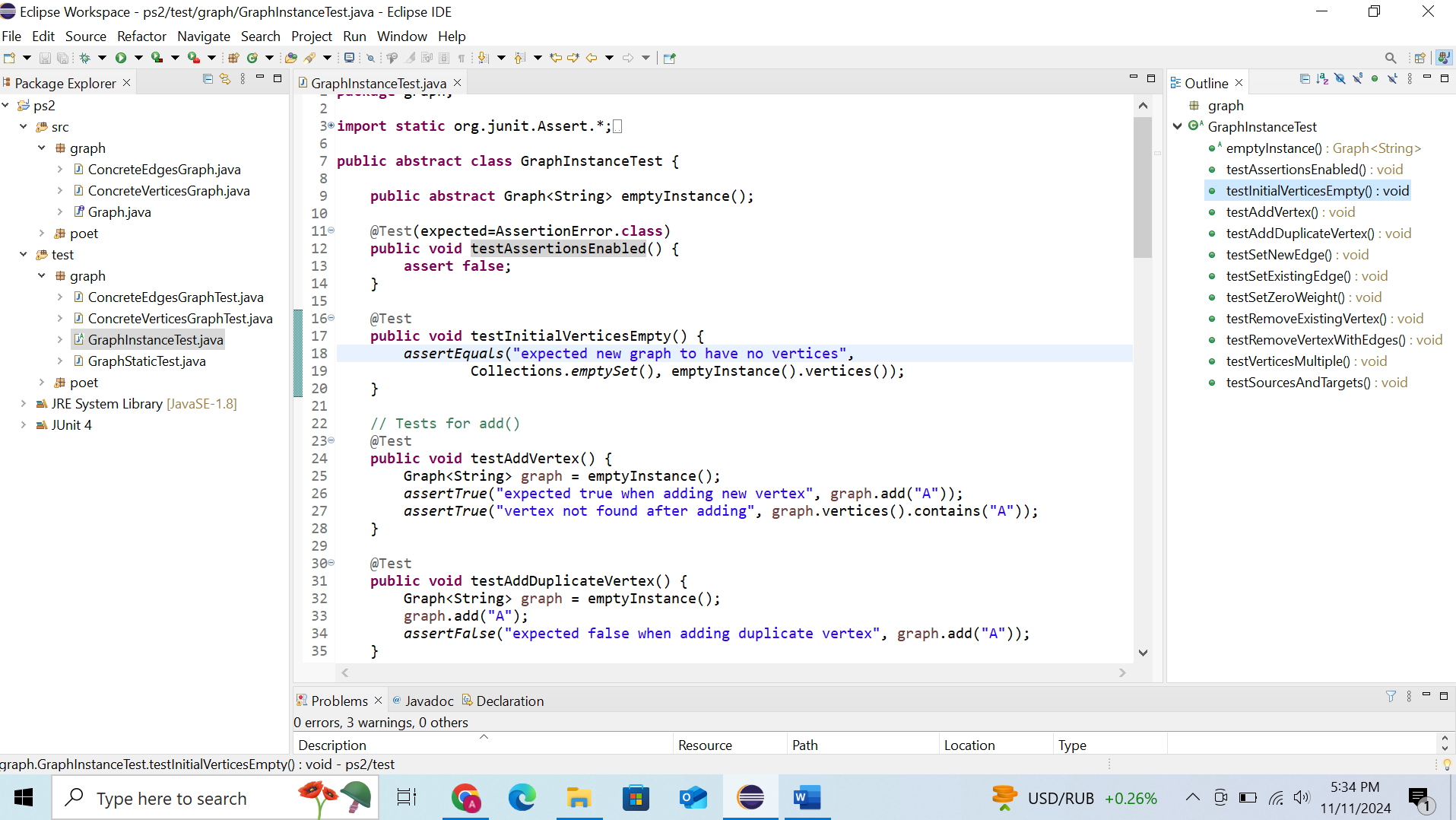
### Document the abstraction function and representation invariant .

* Along with the rep invariant, **document how the type prevents rep exposure** .
* **Implement checkRep** to check the rep invariant.
* **Implement toString** with a useful human-readable representation of the abstract value.

**Screenshots**

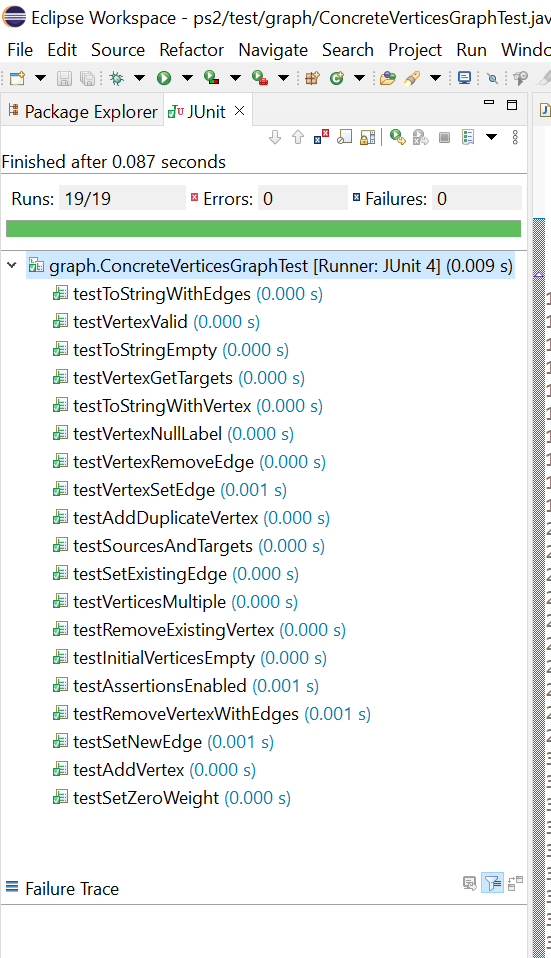
**Task 1:**

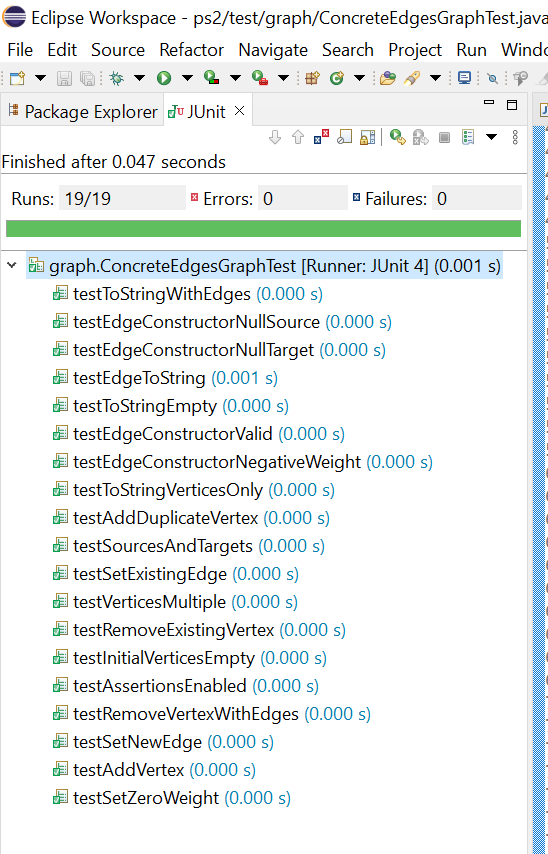
Test cases written and their working assured with the help of other classes, run as JUnit.



**Task 2:**

Concrete Edge Graph and Concrete Vertices Graph classes implemented, and their working shown through their test cases success.





|  |
| --- |
| **GITHUB LINK** |
| <https://github.com/MSaadUmer> |

### Deliverables

Compile a single word document containing Git link by filling in the solution part and submit this Word file on LMS. Also Upload your source code on LMS. In case of any problems with submissions on LMS, submit your Lab assignments by emailing it to [aftab.farooq@seecs.edu.pk.](mailto:aftab.farooq@seecs.edu.pk.)