**Lab Exercise - Equivalence Partitioning (EP) using JUnit**

**Objective:**

To implement **Equivalence Partitioning (EP)** using **JUnit** in Java by testing a function that **categorizes exam scores** into different grade levels.

**Scenario:**

We are testing a **grading system** that assigns grades based on an input score (0-100). The function should return:

* "Fail" for scores **0-39**
* "Pass" for scores **40-59**
* "Merit" for scores **60-79**
* "Distinction" for scores **80-100**
* "Invalid" for any score **less than 0 or greater than 100**

**Equivalence Partitions for Testing:**

| **Partition** | **Valid Range** | **Expected Output** |
| --- | --- | --- |
| Below Range | -10 | "Invalid" |
| Fail | 0-39 | "Fail" |
| Pass | 40-59 | "Pass" |
| Merit | 60-79 | "Merit" |
| Distinction | 80-100 | "Distinction" |
| Above Range | 110 | "Invalid" |

**Steps to Perform:**

1. Create a **Java class** with a getGrade(int score) function.
2. Use **JUnit** to test values from different partitions.
3. Validate results using assertions.

**Implementation**

**Step 1: Implement the Grading Function**

Create a Java class **GradingSystem.java**:

public class GradingSystem {

public static String getGrade(int score) {

if (score < 0 || score > 100) {

return "Invalid";

} else if (score <= 39) {

return "Fail";

} else if (score <= 59) {

return "Pass";

} else if (score <= 79) {

return "Merit";

} else {

return "Distinction";

}

}

}

**Step 2: Create a JUnit Test Class**

Create a test class **GradingSystemTest.java**:

import static org.junit.Assert.assertEquals;

import org.junit.Test;

public class GradingSystemTest {

@Test

public void testBelowRange() {

assertEquals("Invalid", GradingSystem.getGrade(-10));

}

@Test

public void testFailPartition() {

assertEquals("Fail", GradingSystem.getGrade(0));

assertEquals("Fail", GradingSystem.getGrade(20));

assertEquals("Fail", GradingSystem.getGrade(39));

}

@Test

public void testPassPartition() {

assertEquals("Pass", GradingSystem.getGrade(40));

assertEquals("Pass", GradingSystem.getGrade(50));

assertEquals("Pass", GradingSystem.getGrade(59));

}

@Test

public void testMeritPartition() {

assertEquals("Merit", GradingSystem.getGrade(60));

assertEquals("Merit", GradingSystem.getGrade(70));

assertEquals("Merit", GradingSystem.getGrade(79));

}

@Test

public void testDistinctionPartition() {

assertEquals("Distinction", GradingSystem.getGrade(80));

assertEquals("Distinction", GradingSystem.getGrade(90));

assertEquals("Distinction", GradingSystem.getGrade(100));

}

@Test

public void testAboveRange() {

assertEquals("Invalid", GradingSystem.getGrade(110));

}

}

**Step 3: Run the JUnit Tests**

* Use **JUnit 4 or JUnit 5** in **Eclipse, IntelliJ, or VS Code**.
* Run the test cases.
* Ensure all assertions pass.

**Expected Output (JUnit Test Results)**

✔ **testBelowRange()** → **Passed**  
✔ **testFailPartition()** → **Passed**  
✔ **testPassPartition()** → **Passed**  
✔ **testMeritPartition()** → **Passed**  
✔ **testDistinctionPartition()** → **Passed**  
✔ **testAboveRange()** → **Passed**

**Key Takeaways**

✅ **Equivalence Partitioning (EP)** reduces test cases by grouping similar inputs.  
✅ **JUnit Assertions** help validate expected vs actual outputs.  
✅ Ensures **efficient test coverage** without redundant cases.