JUnit Assignment-3

Exercise1:

• Create an Employee class with attributes such as id, name, and salary.
• Write JUnit test cases to verify the behavior of the Employee class.
• Use different types of assertions provided by JUnit to validate the attributes and behavior of the Employee class.
• Ensure that the Employee class methods are working correctly.
Hint:
1. Create Employee Classes
a. public class Employee { private int id; private String name; private double salary;
b. Generate getter and setter
c. public void raiseSalary(double amount) { this.salary += amount; }
2. Create test class
a. public class EmployeeTest { @Test void testEmployeeAttributes() {
i. Employee employee = new Employee(1, "John Doe", 50000.0);
ii. Write assertEquals method for each attribute value
b. void testRaiseSalary() {
i. Employee employee = new Employee(1, "Jane Smith", 60000.0);
ii. Call raiseSalary(int amout)
iii. Write assertEquals method to check the salary values
c. void testEmployeeEquality() {
i. Create Two employee object
ii. Check for that both objects are not same.

Employee Class

ANS:
```
public class Employee {
    private int id;
    private String name;
    private double salary;

    public Employee(int id, String name, double salary) {
        this.id = id;
        this.name = name;
        this.salary = salary;
    }

    public int getId() {
        return id;
    }

    public String getName() {
        return name;
    }

    public double getSalary() {
        return salary;
    }

    public void setId(int id) {
        this.id = id;
    }

    public void setName(String name) {
```

```
        this.name = name;
    }

    public void setSalary(double salary) {
        this.salary = salary;
    }

    public void raiseSalary(double amount) {
        this.salary += amount;
    }

    @Override
    public boolean equals(Object o) {
        if (this == o) return true;
        if (o == null || getClass() != o.getClass()) return false;
        Employee employee = (Employee) o;
        return id == employee.id &&
        Double.compare(employee.salary, salary) == 0 &&
        name.equals(employee.name);
    }

    @Override
    public int hashCode() {
        return Objects.hash(id, name, salary);
    }
}
```

EmployeeTest

```
---------
import org.junit.jupiter.api.Test;
import static org.junit.jupiter.api.Assertions.*;

public class EmployeeTest {

    @Test
    void testEmployeeAttributes() {
        Employee employee = new Employee(1, "John Doe", 50000.0);

        assertEquals(1, employee.getId(), "Employee ID should be 1");
        assertEquals("John Doe", employee.getName(), "Employee name should be John Doe");
        assertEquals(50000.0, employee.getSalary(), "Employee salary should be 50000.0");
    }

    @Test
    void testRaiseSalary() {
        Employee employee = new Employee(1, "Jane Smith", 60000.0);
        employee.raiseSalary(5000.0);

        assertEquals(65000.0, employee.getSalary(), "Employee salary should be 65000.0 after raise");
    }

    @Test
    void testEmployeeEquality() {
        Employee employee1 = new Employee(1, "John Doe", 50000.0);
        Employee employee2 = new Employee(1, "John Doe", 50000.0);
        Employee employee3 = new Employee(2, "Jane Smith", 60000.0);

        assertEquals(employee1, employee2, "Employees with same id, name, and salary should be equal");
```

```
        assertNotEquals(employee1, employee3, "Employees with different id, name, or salary should not be
equal");
        assertNotSame(employee1, employee2, "Even if equal, employee1 and employee2 should not be the
same instance");
    }
}
```

Exercise 2:

Extend the Employee class with a new method to calculate the yearly bonus based on the salary.

• Write parameterized JUnit test cases to test the bonus calculation method with different salary values.

• Use parameterized tests to validate the correctness of the bonus calculation logic for various scenarios.

Hint:

1. Add the calculateYearlyBonus() method

```
public double calculateYearlyBonus() { return salary * 0.1;  }
```

2. Write parametized test

```
public class EmployeeParameterizedTest {
@ParameterizedTest
@ValueSource(doubles = {50000.0, 60000.0, 75000.0})
Void testCalculateYearlyBonus(double salary) {
Employee employee = new Employee(1, "John Doe", salary);
double expectedBonus = salary * 0.1;
assertEquals(expectedBonus, employee.calculateYearlyBonus());
} }
```

ANS:

```
public class Emp_Test extends Employee {
    private int id;
    private String name;
    private double salary;

    public Employee(int id, String name, double salary) {
        this.id = id;
        this.name = name;
        this.salary = salary;
    }

    public double calculateYearlyBonus() {
        return salary * 0.1;
}


}
```

EmployeeParameterizedTest

```
import org.junit.jupiter.params.ParameterizedTest;
import org.junit.jupiter.params.provider.ValueSource;
import static org.junit.jupiter.api.Assertions.assertEquals;

public class EmployeeParameterizedTest {

    @ParameterizedTest
    @ValueSource(doubles = {50000.0, 60000.0, 75000.0})
    void testCalculateYearlyBonus(double salary) {
```

```
        Employee employee = new Employee(1, "John Doe", salary);
        double expectedBonus = salary * 0.1;
        assertEquals(expectedBonus, employee.calculateYearlyBonus(),
                "The yearly bonus calculation is incorrect");
    }
}
```