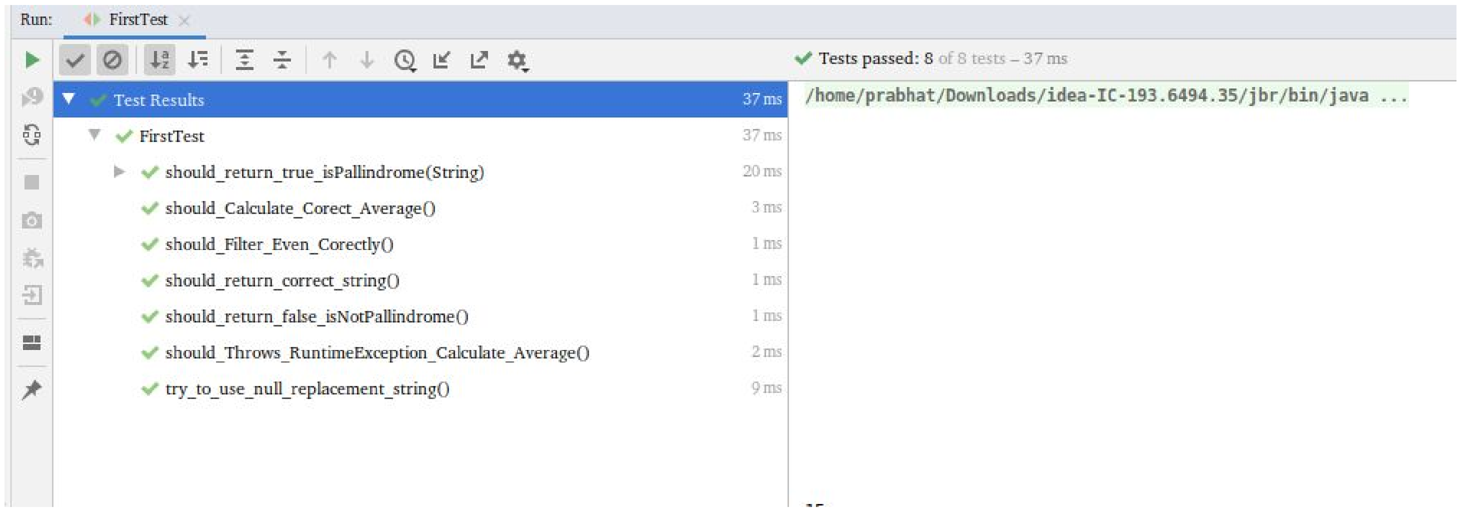
1. Write all possible (including failure, exception case) Unit Tests for all the methods in First.java.

```
package com.healthycoderapp;

import org.junit.jupiter.api.BeforeEach;

import org.junit.jupiter.api.Test;

import org.junit.jupiter.api.function.Executable;

import org.junit.jupiter.params.ParameterizedTest;

import org.junit.jupiter.params.provider.ValueSource;


import java.math.BigDecimal;

import java.util.ArrayList;

import java.util.Arrays;

import java.util.List;


import static org.junit.jupiter.api.Assertions.*;


class FirstTest {

private First first = new First();

  List<BigDecimal> decimalValue = new ArrayList();

  @ParameterizedTest

  @ValueSource(strings = {"bob","lol"})

  public void should_return_true_isPallindrome( String checkString){

     //given

  String str = checkString;

     //when

     boolean output = first.isPallindrome(str);

     //then

     assertTrue(output);

  }


  @Test

  public void should_return_false_isNotPallindrome(){
    //given
    String str = "Miracle";
    //when
    boolean output = first.isPallindrome(str);
    //then
    assertFalse(output);
}

@Test
public void should_Throws_RuntimeException_Calculate_Average(){
    //given
  List<BigDecimal> decimal = new ArrayList<>();
    //when
    Executable executable = () -> first.calculateAverage(decimal);
    //then
    assertThrows(RuntimeException.class, executable);
}



@Test
public void should_Calculate_Corect_Average(){
    //given
    decimalValue.add(new BigDecimal(1));
    decimalValue.add(new BigDecimal(2));
    decimalValue.add(new BigDecimal(3));
    decimalValue.add(new BigDecimal(4));
    decimalValue.add(new BigDecimal(5));
    //when
    BigDecimal actual = first.calculateAverage(decimalValue);
    BigDecimal expected = new BigDecimal(3);
    //then
    assertEquals(expected,actual);
}
```

```
 @Test

 public void should_Filter_Even_Corectly(){

     //given

    List<Integer> actual = new ArrayList<>(Arrays.asList(3,4,5,6,8,10));

     //when

     actual = first.filterEvenElements(actual);

     List<Integer> expected = new ArrayList(Arrays.asList(3,5));

     //then

     assertEquals(expected,actual);

 }


 @Test

 public void should_return_correct_string() {

     //given

String str = "Big Daddy";

String rpStr = "Small";

String substr= "Big";


     //when

     String expected = "Small Daddy";

     String actual = first.replaceSubString(str,substr,rpStr);

     //then

     assertEquals(expected,actual);

 }


 @Test

 public void try_to_use_null_replacement_string() {

     //given

     String str = "Big Daddy";

     String rpStr = null;

     String substr= "Big";


     //when

     String expected = "Big Daddy";
```

```
        String actual = first.replaceSubString(str,substr,rpStr);

        //then

        assertEquals(expected,actual);

    }




}
```



```
Run:   FirstTest
                                                     ✓ Tests passed: 8 of 8 tests – 37 ms
▼ ✓ Test Results                                  37 ms   /home/prabhat/Downloads/idea-IC-193.6494.35/jbr/bin/java ...
   ▼ ✓ FirstTest                                  37 ms
      ▶ ✓ should_return_true_isPallindrome(String)  20 ms
        ✓ should_Calculate_Corect_Average()          3 ms
        ✓ should_Filter_Even_Corectly()              1 ms
        ✓ should_return_correct_string()             1 ms
        ✓ should_return_false_isNotPallindrome()     1 ms
        ✓ should_Throws_RuntimeException_Calculate_Average()  2 ms
        ✓ try_to_use_null_replacement_string()       9 ms
```

2. Write Unit tests for HealthyCoder app given in the Udemy session. You need to write tests for the BMICalculator and DitePlanner.

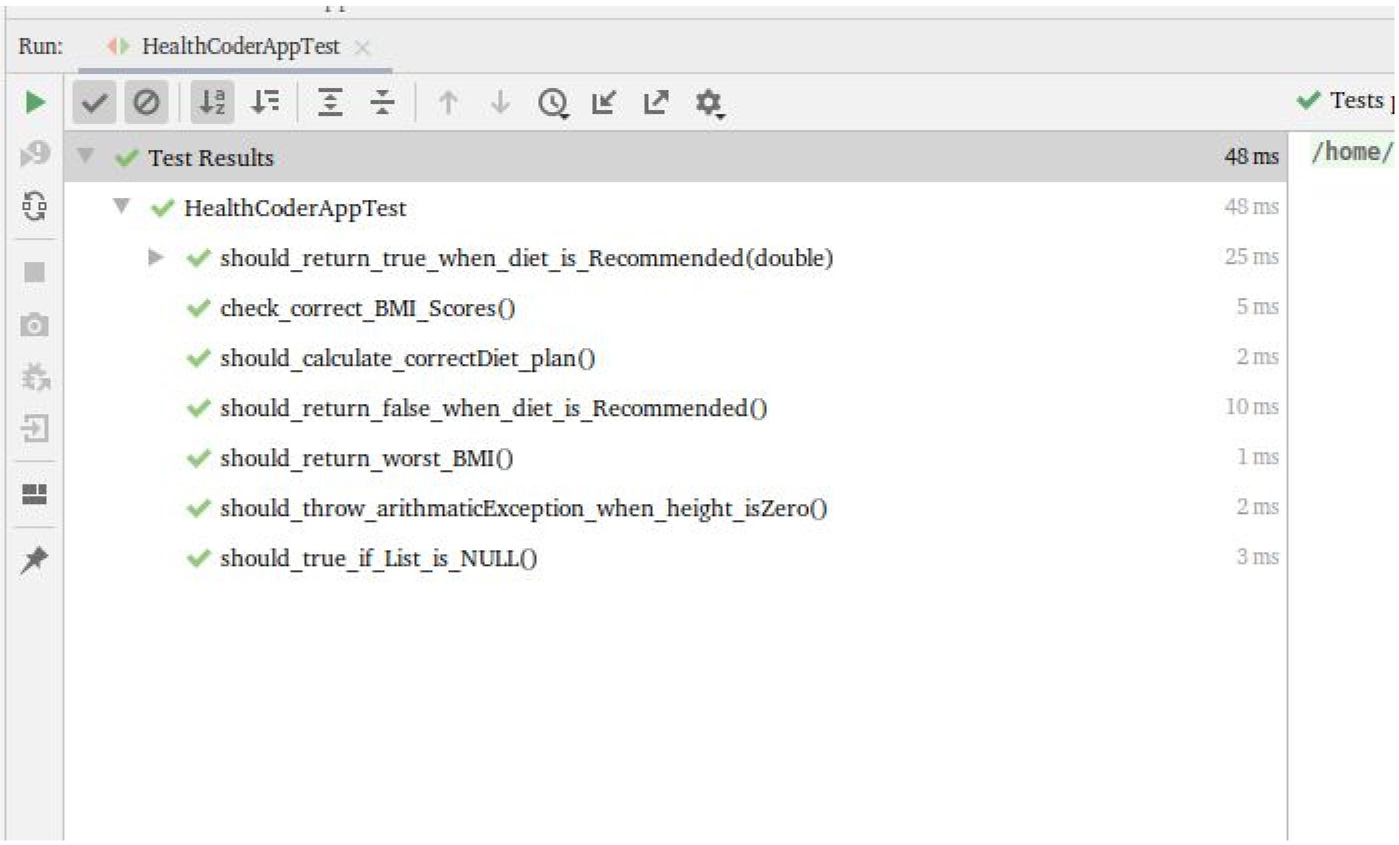
```
package com.healthycoderapp;

import org.junit.jupiter.api.BeforeEach;

import org.junit.jupiter.api.DynamicTest;

import org.junit.jupiter.api.Test;

import org.junit.jupiter.api.function.Executable;

import org.junit.jupiter.params.ParameterizedTest;

import org.junit.jupiter.params.provider.ValueSource;


import java.util.ArrayList;

import java.util.List;

import static org.junit.jupiter.api.Assertions.*;

public class HealthCoderAppTest{
```

```
private DietPlanner dietPlanner;

private Coder coder;

@ParameterizedTest

@ValueSource(doubles = {87.6,90.1,80.2})

public void should_return_true_when_diet_is_Recommended( double checkWeight){

    //given

    double weight = checkWeight;

    double height = 1.7;

    //when

    boolean output = BMICalculator.isDietRecommended(weight, height);

    //then

    assertTrue(output);

}


@Test

public void should_return_false_when_diet_is_Recommended() {

    //given

    double weight = 50;

    double height = 1.9;

    //when

    boolean output = BMICalculator.isDietRecommended(weight, height);

    //then

    assertFalse(output);

}


@Test

public void should_throw_arithmaticException_when_height_isZero() {

    //given

    double weight = 50;

    double height = 0;

    //when

    Executable executable = () -> BMICalculator.isDietRecommended(weight, height);

    //then

    assertThrows(ArithmeticException.class, executable);
```

```
}

// Multiple Assertions
@Test
public void should_return_worst_BMI() {
    //given
    List list = new ArrayList();
    list.add(new Coder(1.8, 60));
    list.add(new Coder(1.5, 70));
    list.add(new Coder(1.8, 92));
    //when
    Coder BMI = BMICalculator.findCoderWithWorstBMI(list);
    //then
    assertAll(
            () -> assertEquals(1.5, BMI.getHeight()),
            () -> assertEquals(70, BMI.getWeight()));
}


@Test
public void should_true_if_List_is_NULL() {
    //given
    List list = new ArrayList();
    //when
    Coder BMI = BMICalculator.findCoderWithWorstBMI(list);
    //then
    assertNull(BMI);
}


@Test
public void check_correct_BMI_Scores() {
    //given
    List<Coder> list = new ArrayList();
    list.add(new Coder(1.8, 60));
    list.add(new Coder(1.5, 70));
    list.add(new Coder(1.8, 92));

    //when
    double[] expected = {18.52,31.11,28.4};
    double[] actual = BMICalculator.getBMIScores(list);
    //then
    assertAll(
            () -> assertArrayEquals(expected, actual));
}
```

```
@BeforeEach
void setup(){
    this.dietPlanner = new DietPlanner(20,30,50);
}
@Test
void should_calculate_correctDiet_plan() {
    //given
    Coder coder = new Coder(1.82,75,26,Gender.MALE);
    DietPlan expected = new DietPlan(2202,110,73,275);
    //when
    DietPlan  actual = dietPlanner.calculateDiet(coder);
    assertAll(
            ()->assertEquals(expected.getCalories(),actual.getCalories()),
            ()->assertEquals(expected.getCarbohydrate(),actual.getCarbohydrate()),
            ()->assertEquals(expected.getFat(),actual.getFat()),
            ()->assertEquals(expected.getProtein(),actual.getProtein())
    );
}
```

```
}
```



```
Run:   HealthCoderAppTest

✓ Test Results                                                  48 ms
  ✓ HealthCoderAppTest                                          48 ms
    ✓ should_return_true_when_diet_is_Recommended(double)       25 ms
    ✓ check_correct_BMI_Scores()                                 5 ms
    ✓ should_calculate_correctDiet_plan()                        2 ms
    ✓ should_return_false_when_diet_is_Recommended()            10 ms
    ✓ should_return_worst_BMI()                                  1 ms
    ✓ should_throw_arithmaticException_when_height_isZero()      2 ms
    ✓ should_true_if_List_is_NULL()                              3 ms
```

✓ Tests

/home/