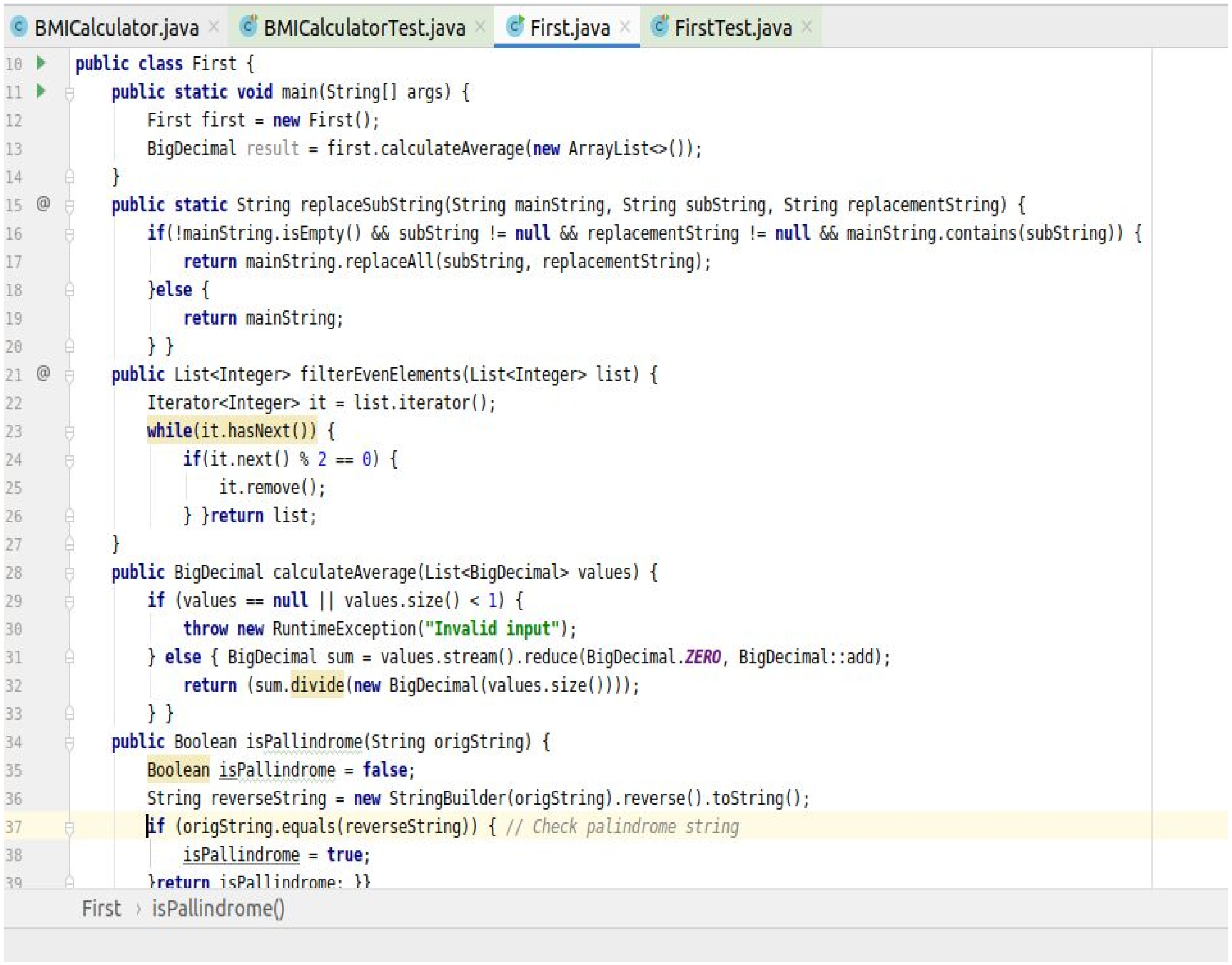
Java: Unit testing

Q1. Write all possible (including failure, exception case) Unit Tests for all the methods in First.java.

Answer1.

For test cases let me first show you the actual content of First.java file in the screenshot.



```java
public class First {
    public static void main(String[] args) {
        First first = new First();
        BigDecimal result = first.calculateAverage(new ArrayList<>());
    }
    public static String replaceSubString(String mainString, String subString, String replacementString) {
        if(!mainString.isEmpty() && subString != null && replacementString != null && mainString.contains(subString)) {
            return mainString.replaceAll(subString, replacementString);
        }else {
            return mainString;
        } }
    public List<Integer> filterEvenElements(List<Integer> list) {
        Iterator<Integer> it = list.iterator();
        while(it.hasNext()) {
            if(it.next() % 2 == 0) {
                it.remove();
            } }return list;
    }
    public BigDecimal calculateAverage(List<BigDecimal> values) {
        if (values == null || values.size() < 1) {
            throw new RuntimeException("Invalid input");
        } else { BigDecimal sum = values.stream().reduce(BigDecimal.ZERO, BigDecimal::add);
            return (sum.divide(new BigDecimal(values.size())));
        } }
    public Boolean isPallindrome(String origString) {
        Boolean isPallindrome = false;
        String reverseString = new StringBuilder(origString).reverse().toString();
        if (origString.equals(reverseString)) { // Check palindrome string
            isPallindrome = true;
        }return isPallindrome; }}
```

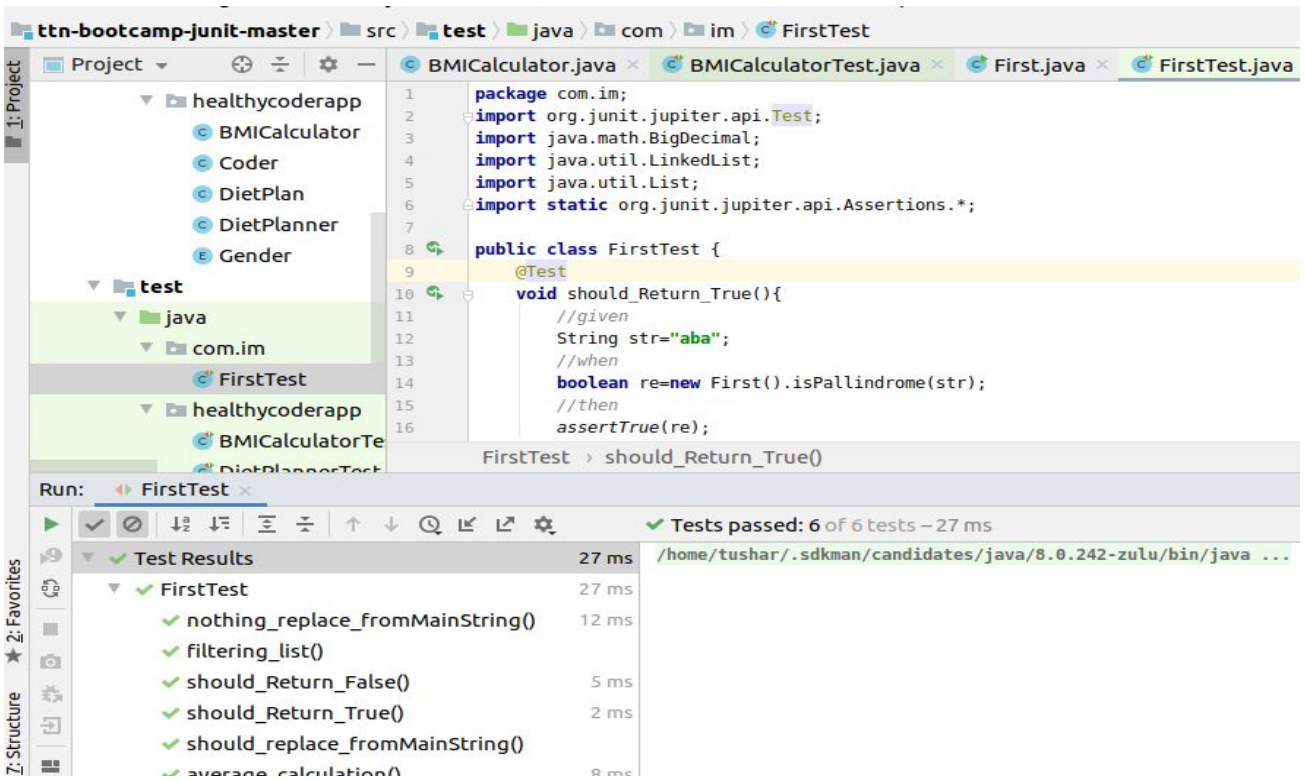
Whole code of unit Testing :

```java
package com.im;

import org.junit.jupiter.api.Test;

import java.math.BigDecimal;

import java.util.LinkedList;
import java.util.List;
import static org.junit.jupiter.api.Assertions.*;
public class FirstTest {
  @Test
  void should_Return_True(){
      //given
      String str="aba";
      //when
      boolean re=new First().isPallindrome(str);
      //then
      assertTrue(re);
  }
  @Test
  void should_Return_False(){
      //given
      String str="abaaaa";
      //when
      boolean re=new First().isPallindrome(str);
      //then
      assertFalse(re);
  }
  @Test
  void nothing_replace_fromMainString() {
      //given
      String mainString="Tushazzz";
      String subString="zzz";
      String replacementString="r";
      //when
     String srt =First.replaceSubString(mainString, subString,replacementString);
     //then
      assertEquals(srt,"Tushar");
  }
```

```
@Test
void should_replace_fromMainString() {
    //given
    String mainString="sddsdddddr";
    String subString="dmllkdkkd";
    String replacementString="";
    //when
    String srt =First.replaceSubString(mainString, subString,replacementString);
    //then
    assertEquals(srt,"sddsdddddr");
}
@Test
void  filtering_list(){
    //given
    List<Integer> integers=new LinkedList<>();
    integers.add(1);
    integers.add(10);
    integers.add(41);
    integers.add(16);
    integers.add(18);
    List<Integer>integers1=new LinkedList<>();
    integers1.add(1);
    integers1.add(41);
    //when
    List<Integer> red=new First().filterEvenElements(integers);
    //then
    assertEquals(red,integers1);
}
@Test
void average_calculation(){
    //given
    List<BigDecimal> list=new LinkedList<>();
```

```
        list.add(new BigDecimal(2));

        list.add(new BigDecimal(4));

        BigDecimal str=new BigDecimal(3);

        //when

        BigDecimal decimal=new First().calculateAverage(list);

        //then

        assertEquals(decimal,str);

    }

}
```

Screenshot of the output:--------



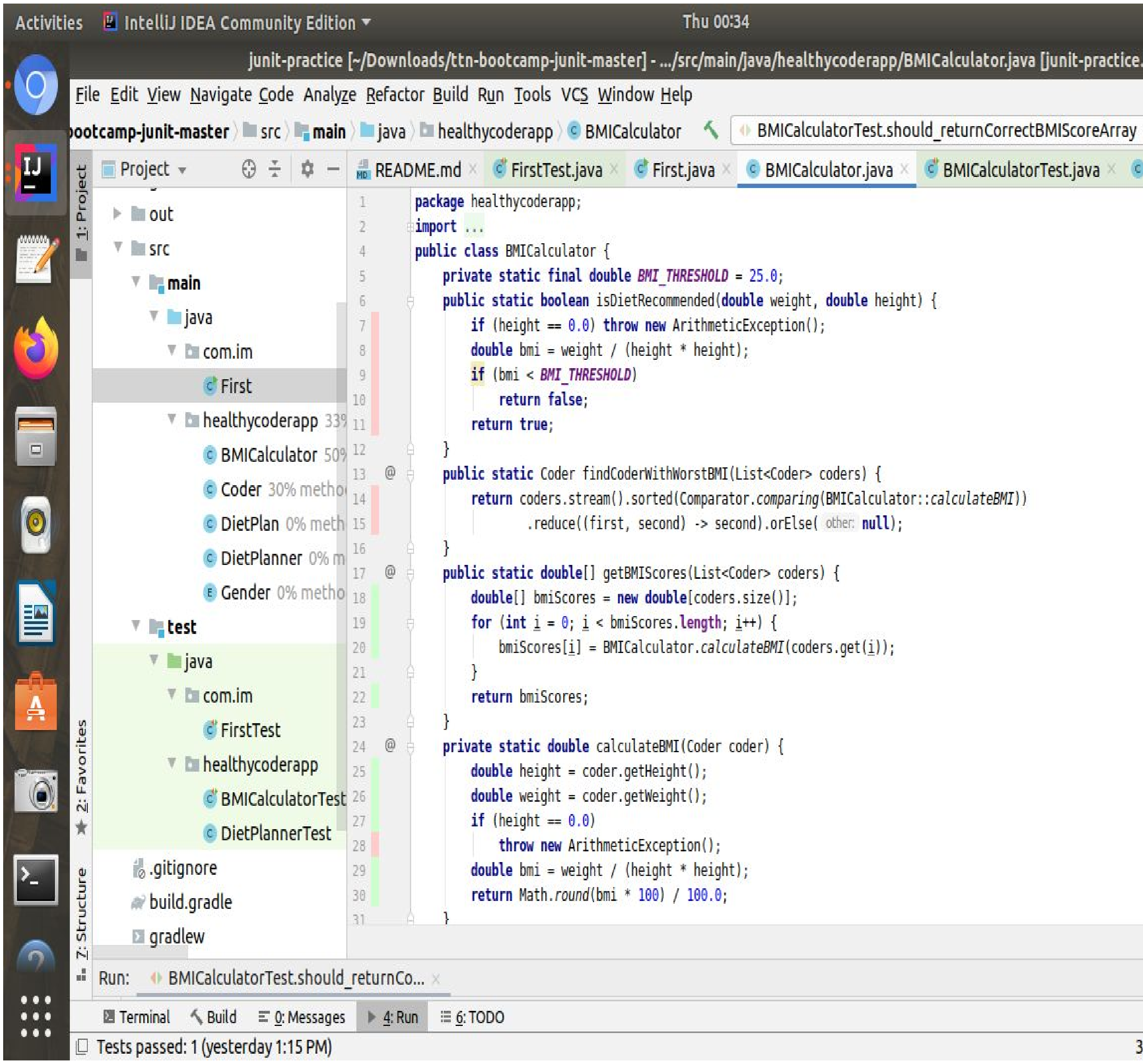
```
package com.im;
import org.junit.jupiter.api.Test;
import java.math.BigDecimal;
import java.util.LinkedList;
import java.util.List;
import static org.junit.jupiter.api.Assertions.*;

public class FirstTest {
    @Test
    void should_Return_True(){
        //given
        String str="aba";
        //when
        boolean re=new First().isPallindrome(str);
        //then
        assertTrue(re);
```

```
Tests passed: 6 of 6 tests – 27 ms
/home/tushar/.sdkman/candidates/java/8.0.242-zulu/bin/java ...

Test Results                              27 ms
  FirstTest                               27 ms
    nothing_replace_fromMainString()      12 ms
    filtering_list()
    should_Return_False()                  5 ms
    should_Return_True()                   2 ms
    should_replace_fromMainString()
    average_calculation()                  8 ms
```

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

Answer:

For test cases let me first show you the actual content of BMICalculator file in the screenshot.



Activities
IntelliJ IDEA Community Edition
Thu 00:34
junit-practice [~/Downloads/ttn-bootcamp-junit-master] - .../src/main/java/healthycoderapp/BMICalculator.java [junit-practice.
File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help
ootcamp-junit-master src main java healthycoderapp BMICalculator
BMICalculatorTest.should_returnCorrectBMIScoreArray
Project
README.md FirstTest.java First.java BMICalculator.java BMICalculatorTest.java
out
src
main
java
com.im
First
healthycoderapp 33%
BMICalculator 50%
Coder 30% metho
DietPlan 0% meth
DietPlanner 0% m
Gender 0% metho
test
java
com.im
FirstTest
healthycoderapp
BMICalculatorTest
DietPlannerTest
.gitignore
build.gradle
gradlew
1: Project
2: Favorites
7: Structure
Run: BMICalculatorTest.should_returnCo...
Terminal Build 0: Messages 4: Run 6: TODO

```java
package healthycoderapp;
import ...
public class BMICalculator {
    private static final double BMI_THRESHOLD = 25.0;
    public static boolean isDietRecommended(double weight, double height) {
        if (height == 0.0) throw new ArithmeticException();
        double bmi = weight / (height * height);
        if (bmi < BMI_THRESHOLD)
            return false;
        return true;
    }
    public static Coder findCoderWithWorstBMI(List<Coder> coders) {
        return coders.stream().sorted(Comparator.comparing(BMICalculator::calculateBMI))
                .reduce((first, second) -> second).orElse( other: null);
    }
    public static double[] getBMIScores(List<Coder> coders) {
        double[] bmiScores = new double[coders.size()];
        for (int i = 0; i < bmiScores.length; i++) {
            bmiScores[i] = BMICalculator.calculateBMI(coders.get(i));
        }
        return bmiScores;
    }
    private static double calculateBMI(Coder coder) {
        double height = coder.getHeight();
        double weight = coder.getWeight();
        if (height == 0.0)
            throw new ArithmeticException();
        double bmi = weight / (height * height);
        return Math.round(bmi * 100) / 100.0;
    }
```

Tests passed: 1 (yesterday 1:15 PM)

Test cases:

1st test ---- return true

```
class BMICalculatorTest {
  @Test
  void should_Return_True(){
     //given
     double height=1.7;
     double weight=79.2;
     //when
     boolean recommend=BMICalculator.isDietRecommended(weight,height);
     //then
     assertTrue(recommend);
  }
```

2nd test-----return false

```
@Test
void should_Return_False(){
  //given
  double height=1.9;
  double weight=50.2;
  //when
  boolean recommend=BMICalculator.isDietRecommended(weight,height);
  //then
  assertFalse(recommend);
}
```

3rd test----return exception

```
@Test
void should_Return_Exception_when_height_zero(){
  //given
  double height=0.0;
  double weight=50.7;
  //when
  Executable executable=()-> BMICalculator.isDietRecommended(weight,height);
  //then
  assertThrows(ArithmeticException.class, executable);
}
```

4th case------return WorstBMI

```
@Test
void should_Return_WorstBMI_when_coderList_notEmpty(){
  //given
 List<Coder> coders=new ArrayList<Coder>();
 coders.add(new Coder(1.2,30.2));
  coders.add(new Coder(1.7,90.5));
  coders.add(new Coder(1.2,68.0));
  //when
  Coder coderWorstBMI=BMICalculator.findCoderWithWorstBMI(coders);
  //then
  assertAll(
        ()-> assertEquals(1.2,coderWorstBMI.getHeight()),
     ()-> assertEquals(68.0,coderWorstBMI.getWeight())
  );
}
```

5th case-----------return Null

```
@Test
void should_Return_null(){
  //given
  List<Coder> coders=new ArrayList<Coder>();//no coder element added
```

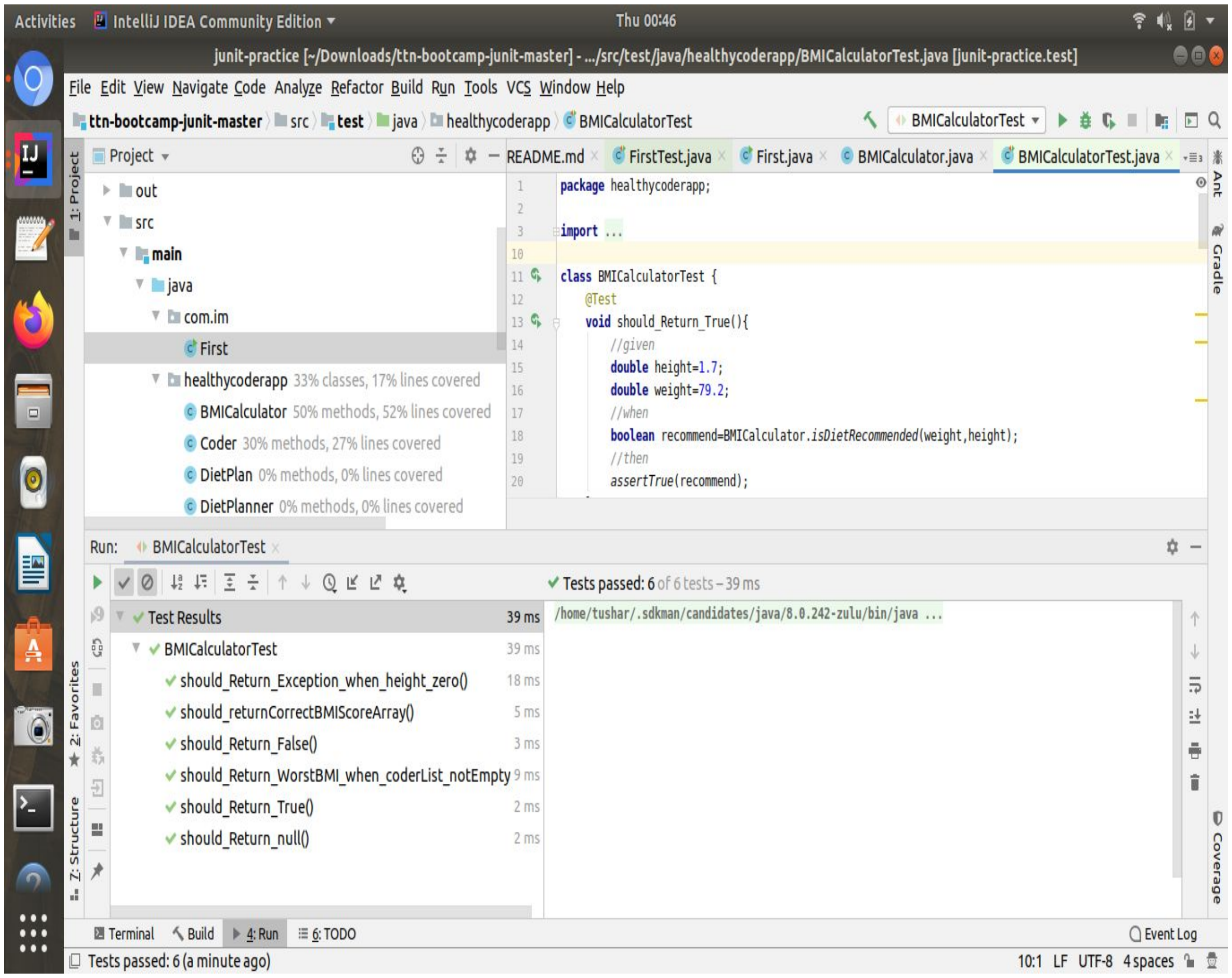
```
   //when
   Coder coderNull=BMICalculator.findCoderWithWorstBMI(coders);
   //then
  assertNull(coderNull);
}
```

6th case------------return correct BMI Score

```
@Test
void should_returnCorrectBMIScoreArray(){
   //given
   List<Coder> coders=new ArrayList<>();
   coders.add(new Coder(1.80,60.0));
   coders.add(new Coder(1.82,98.0));
   coders.add(new Coder(1.82,64.7));
   double[] expected={18.52,29.59,19.53};
   //when
   double[] BMIScore= BMICalculator.getBMIScores(coders);
   //then
   assertArrayEquals(expected,BMIScore);
}
```

OVERALL CODE FOR BMICalculatorTest :

```
package healthycoderapp;

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

import java.util.ArrayList;
import java.util.List;
import static org.junit.jupiter.api.Assertions.*;

class BMICalculatorTest {
  @Test
  void should_Return_True(){
     //given
     double height=1.7;
     double weight=79.2;
     //when
     boolean recommend=BMICalculator.isDietRecommended(weight,height);
     //then
     assertTrue(recommend);
  }
  @Test
  void should_Return_False(){
     //given
     double height=1.9;
     double weight=50.2;
     //when
     boolean recommend=BMICalculator.isDietRecommended(weight,height);
     //then
     assertFalse(recommend);
  }
  @Test
  void should_Return_Exception_when_height_zero(){
     //given
     double height=0.0;
     double weight=50.7;
     //when
     Executable executable=()-> BMICalculator.isDietRecommended(weight,height);
     //then
     assertThrows(ArithmeticException.class, executable);
  }
  @Test
  void should_Return_WorstBMI_when_coderList_notEmpty(){
     //given
    List<Coder> coders=new ArrayList<Coder>();
    coders.add(new Coder(1.2,30.2));
     coders.add(new Coder(1.7,90.5));
     coders.add(new Coder(1.2,68.0));
     //when
     Coder coderWorstBMI=BMICalculator.findCoderWithWorstBMI(coders);
     //then
     assertAll(
            ()-> assertEquals(1.2,coderWorstBMI.getHeight()),
        ()-> assertEquals(68.0,coderWorstBMI.getWeight())
     );
  }
  @Test
  void should_Return_null(){
        //given
        List<Coder> coders=new ArrayList<Coder>();//no coder element added
        //when
        Coder coderNull=BMICalculator.findCoderWithWorstBMI(coders);
        //then
       assertNull(coderNull);
    }
    @Test
    void should_returnCorrectBMIScoreArray(){
        //given
        List<Coder> coders=new ArrayList<>();
        coders.add(new Coder(1.80,60.0));
        coders.add(new Coder(1.82,98.0));
        coders.add(new Coder(1.82,64.7));
        double[] expected={18.52,29.59,19.53};
        //when
        double[] BMIScore= BMICalculator.getBMIScores(coders);
        //then
        assertArrayEquals(expected,BMIScore);
    }
}
```



junit-practice [~/Downloads/ttn-bootcamp-junit-master] - .../src/test/java/healthycoderapp/BMICalculatorTest.java [junit-practice.test]

```
package healthycoderapp;

import ...

class BMICalculatorTest {
    @Test
    void should_Return_True(){
        //given
        double height=1.7;
        double weight=79.2;
        //when
        boolean recommend=BMICalculator.isDietRecommended(weight,height);
        //then
        assertTrue(recommend);
```

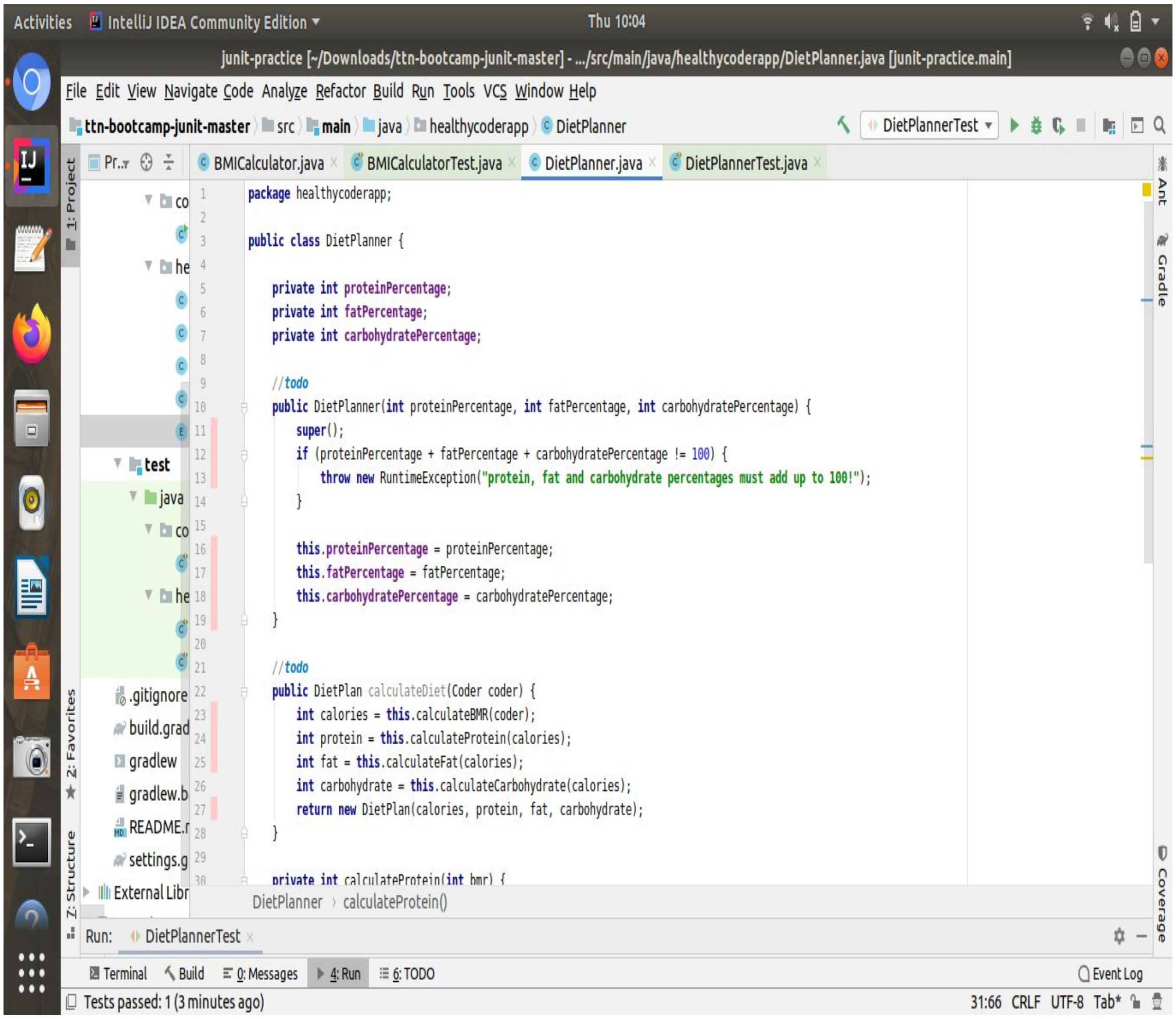
```
healthycoderapp 33% classes, 17% lines covered
  BMICalculator 50% methods, 52% lines covered
  Coder 30% methods, 27% lines covered
  DietPlan 0% methods, 0% lines covered
  DietPlanner 0% methods, 0% lines covered
```

```
✓ Tests passed: 6 of 6 tests – 39 ms

✓ Test Results                                          39 ms
  ✓ BMICalculatorTest                                   39 ms
    ✓ should_Return_Exception_when_height_zero()        18 ms
    ✓ should_returnCorrectBMIScoreArray()                5 ms
    ✓ should_Return_False()                              3 ms
    ✓ should_Return_WorstBMI_when_coderList_notEmpty     9 ms
    ✓ should_Return_True()                               2 ms
    ✓ should_Return_null()                               2 ms

/home/tushar/.sdkman/candidates/java/8.0.242-zulu/bin/java ...
```

Tests passed: 6 (a minute ago)

Now for file DitePlanner

The code of the dietPlanner file is: This has already been given to us.



```
junit-practice [~/Downloads/ttn-bootcamp-junit-master] - .../src/main/java/healthycoderapp/DietPlanner.java [junit-practice.main]
```

```java
package healthycoderapp;

public class DietPlanner {

    private int proteinPercentage;
    private int fatPercentage;
    private int carbohydratePercentage;

    //todo
    public DietPlanner(int proteinPercentage, int fatPercentage, int carbohydratePercentage) {
        super();
        if (proteinPercentage + fatPercentage + carbohydratePercentage != 100) {
            throw new RuntimeException("protein, fat and carbohydrate percentages must add up to 100!");
        }

        this.proteinPercentage = proteinPercentage;
        this.fatPercentage = fatPercentage;
        this.carbohydratePercentage = carbohydratePercentage;
    }

    //todo
    public DietPlan calculateDiet(Coder coder) {
        int calories = this.calculateBMR(coder);
        int protein = this.calculateProtein(calories);
        int fat = this.calculateFat(calories);
        int carbohydrate = this.calculateCarbohydrate(calories);
        return new DietPlan(calories, protein, fat, carbohydrate);
    }

    private int calculateProtein(int bmr) {
```

```
Tests passed: 1 (3 minutes ago)
```

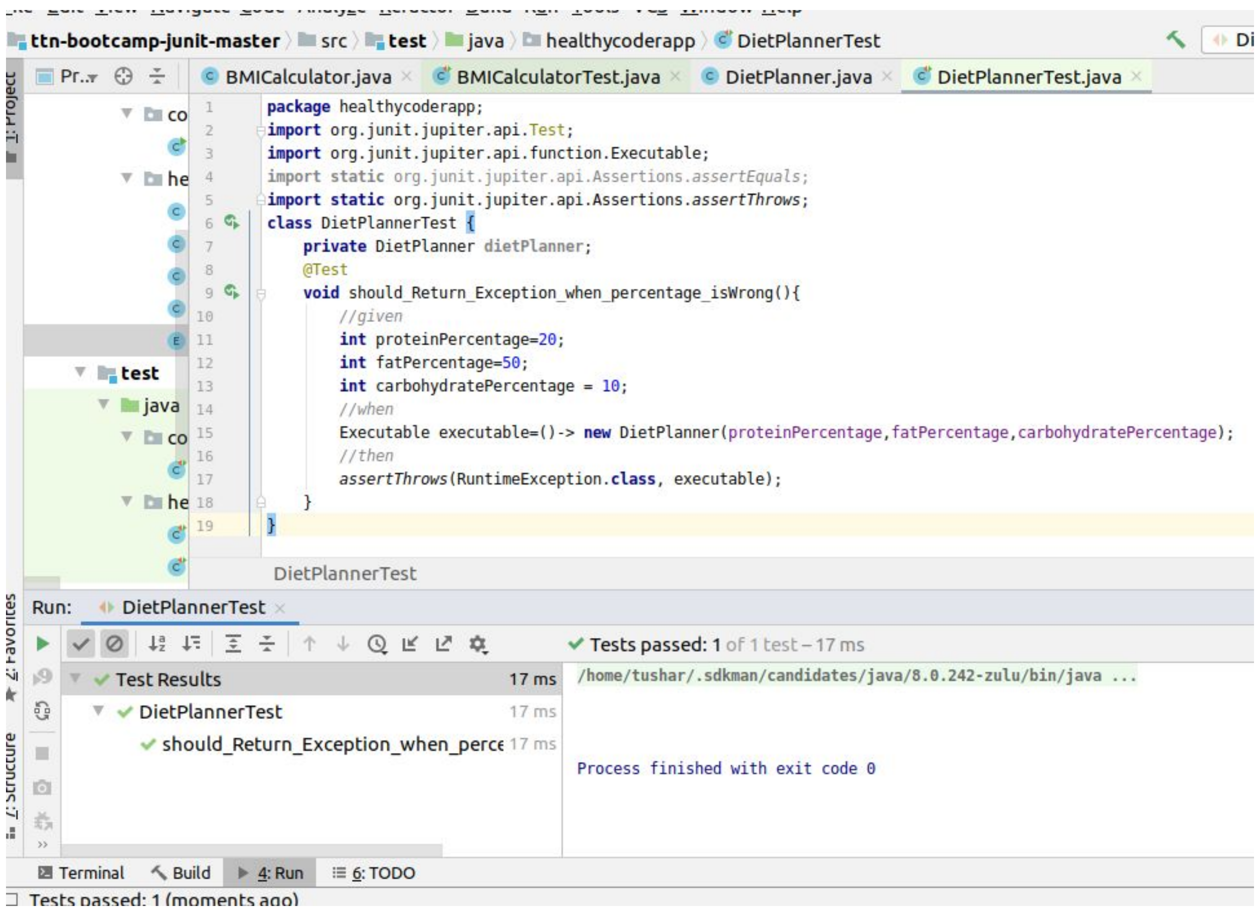
Activities
IntelliJ IDEA Community Edition
Thu 10:04
File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help
ttn-bootcamp-junit-master › src › main › java › healthycoderapp › DietPlanner
DietPlannerTest
BMICalculator.java
BMICalculatorTest.java
DietPlanner.java
DietPlannerTest.java
DietPlanner › calculateProtein()
Run: DietPlannerTest
Terminal
Build
0: Messages
4: Run
6: TODO
Event Log
31:66 CRLF UTF-8 Tab*

Now we need to design the unit test cases of this :

Test case1:

```java
@Test
void should_Return_Exception_when_percentage_isWrong(){
  //given
  int proteinPercentage=20;
  int fatPercentage=50;
```

```
    int carbohydratePercentage = 10;
    //when
    Executable executable=()-> new DietPlanner(proteinPercentage,fatPercentage,carbohydratePercentage);
    //then
    assertThrows(RuntimeException.class, executable);
}
```



```
package healthycoderapp;
import org.junit.jupiter.api.Test;
import org.junit.jupiter.api.function.Executable;
import static org.junit.jupiter.api.Assertions.assertEquals;
import static org.junit.jupiter.api.Assertions.assertThrows;
class DietPlannerTest {
    private DietPlanner dietPlanner;
    @Test
    void should_Return_Exception_when_percentage_isWrong(){
        //given
        int proteinPercentage=20;
        int fatPercentage=50;
        int carbohydratePercentage = 10;
        //when
        Executable executable=()-> new DietPlanner(proteinPercentage,fatPercentage,carbohydratePercentage);
        //then
        assertThrows(RuntimeException.class, executable);
    }
}
```

```
Tests passed: 1 of 1 test – 17 ms

Test Results                               17 ms
  DietPlannerTest                          17 ms
    should_Return_Exception_when_perce     17 ms

/home/tushar/.sdkman/candidates/java/8.0.242-zulu/bin/java ...

Process finished with exit code 0

Tests passed: 1 (moments ago)
```

We can not design any more unit test cases for this file because all other methods are declared private and the good practice of JUnit says not to write unit test cases for Private method.