*//Ques1: Write all possible (including failure, exception case) Unit Tests for all the methods in First.java.*

**package** com.im;

**import** org.junit.jupiter.api.BeforeEach;

**import** org.junit.jupiter.api.Nested;

**import** org.junit.jupiter.api.Test;

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

**import** java.math.BigDecimal;

**import** java.util.ArrayList;

**import** java.util.Iterator;

**import** java.util.List;

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

**public class** FirstTest {

@Test

**void** canary(){

*assertTrue*(**true**);

}

@Nested

**class** replaceSubStringTest {

First **obj**;

@BeforeEach

**void** initialize() {

**obj** = **new** First();

}

@Test

**void** should\_returnTrue\_When\_replaceSubString\_isCorrect() {

*//given*

String mainString = **"Hello World"**;

String subString = **"World"**;

String replacementString = **"India"**;

*//when*

String temp = **obj**.replaceSubString(mainString, subString, replacementString);

*//then*

*assertEquals*(**"Hello India"**, temp);

}

@Test

**void** should\_returnTrue\_When\_mainString\_isEmpty() {

*//given*

String mainString = **""**;

String subString = **"World"**;

String replacementString = **"India"**;

*//when*

String temp = **obj**.replaceSubString(mainString, subString, replacementString);

*//then*

*assertEquals*(**""**, temp);

}

@Test

**void** should\_returnTrue\_When\_subString\_isNull(){ *//subString != null*

*//given*

String mainString = **"Hello World"**;

String subString=**null**;

String replacementString=**"India"**;

*//when*

String temp = **obj**.replaceSubString(mainString,subString,replacementString);

*//then*

*assertEquals*(**"Hello World"**, temp);

}

@Test

**void** should\_returnTrue\_When\_replacementString\_isNull(){

*//given*

String mainString=**"Hello World"**;

String subString=**"World"**;

String replacementString=**null**;

*//when*

String temp = **obj**.replaceSubString(mainString,subString,replacementString);

*//then*

*assertEquals*(**"Hello World"**, temp);

}

@Test

**void** should\_returnTrue\_When\_mainString\_contains\_subString(){

*//given*

String mainString=**"Hello World"**;

String subString=**"Usa"**;

String replacementString=**"India"**;

*//when*

String temp = **obj**.replaceSubString(mainString,subString,replacementString);

*//then*

*assertEquals*(**"Hello World"**, temp);

}

}

@Nested

**class** filterEvenElementsTest {

First **obj**;

@BeforeEach

**void** initialize() {

**obj** = **new** First();

}

@Test

**void** should\_returnTrue\_List\_isCorrect(){

*//given*

List<Integer> ls = **new** ArrayList<>();

ls.add(5);

ls.add(2);

ls.add(9);

List<Integer> expected = **new** ArrayList<>();

expected.add(5);

expected.add(9);

*//when*

List<Integer> output = **obj**.filterEvenElements(ls);

*//then*

*assertArrayEquals*(expected.toArray(),output.toArray());

}

}

@Nested

**class** calculateAverageTest {

First **obj**;

@BeforeEach

**void** initialize() {

**obj** = **new** First();

}

@Test

**void** should\_returnTrue\_When\_calculateAverage(){

*//given*

List<BigDecimal> values = **new** ArrayList<>();

values.add(**new** BigDecimal(**"10.3"**));

values.add(**new** BigDecimal(**"99.3"**));

values.add(**new** BigDecimal(**"22.1"**));

*//when*

BigDecimal avg = **obj**.calculateAverage(values);

System.***out***.println(avg);

*//then*

*assertEquals*(**new** BigDecimal(**"43.9"**),avg);

}

@Test

**void** should\_returnTrue\_When\_Value\_isNULL(){

*//given*

List<BigDecimal> values = **new** ArrayList<>();

*//when*

Executable exe = () -> **obj**.calculateAverage(values);

*//then*

*assertThrows*(RuntimeException.**class**,exe);

}

@Test

**void** should\_returnTrue\_When\_ValueSize\_Is\_LessThan\_1(){

*//given*

List<BigDecimal> values = **new** ArrayList<>();

*//when*

Executable exe = () -> **obj**.calculateAverage(values);

*//then*

*assertThrows*(RuntimeException.**class**,exe);

}

}

@Nested

**class** isPallindromeTest {

First **obj**;

@BeforeEach

**void** initialize() {

**obj** = **new** First();

}

@Test

**void** should\_returnTrue\_Palindrome\_isCorrect() {

*//given*

String origString = **"abcba"**;

String reverseString = **"abcba"**;

*//when*

Boolean expected = **obj**.isPallindrome(origString);

*//then*

*assertTrue*(expected);

}

@Test

**void** should\_returnFalse\_Palindrome\_isNotCorrect() {

*//given*

String origString = **"Aakash"**;

String reverseString = **"hsakaA"**;

*//when*

Boolean expected = **obj**.isPallindrome(origString);

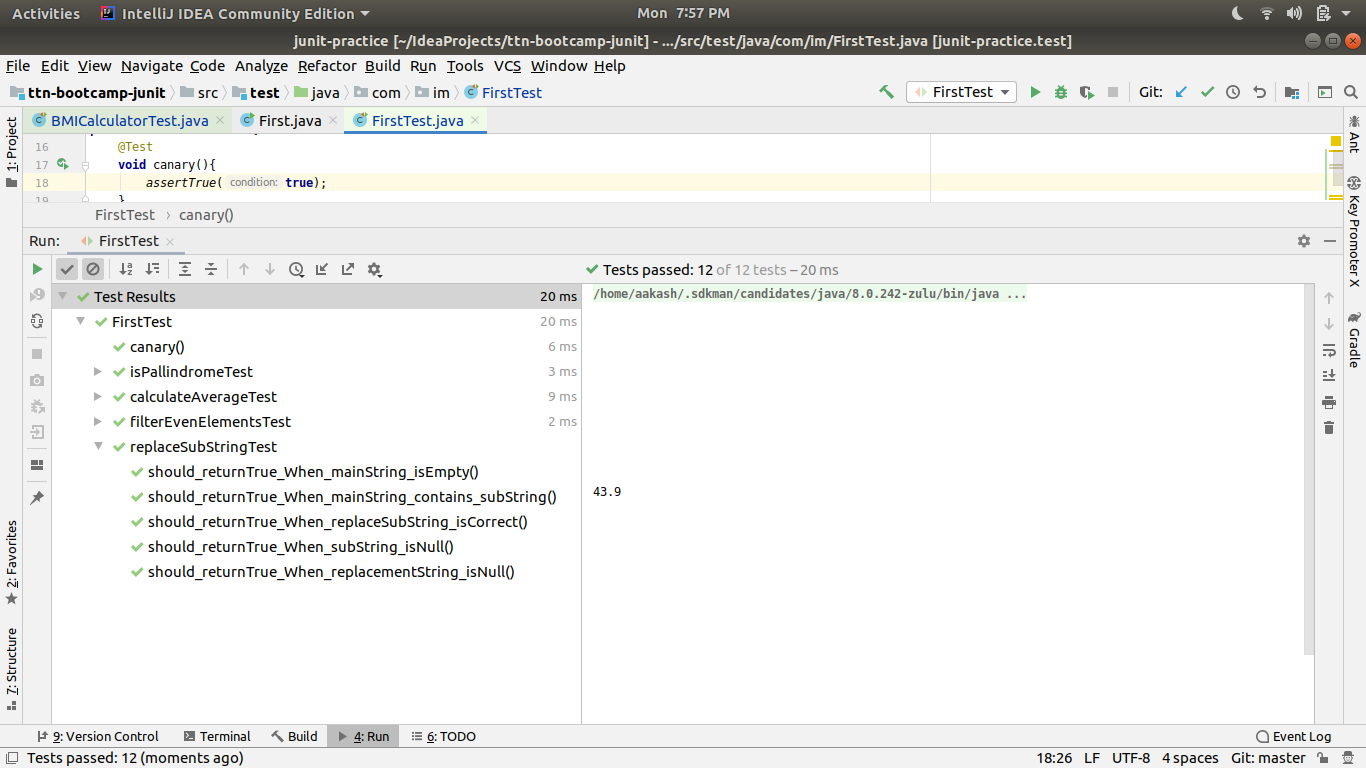
*//then*

*assertFalse*(expected);

}

}

}



*//Ques 2.1: Write Unit tests for HealthyCoder app given in the Udemy session. You need to write tests for the BMICalculator and DitePlanner.*

**package** healthycoderapp;

**import** com.im.First;

**import** org.junit.jupiter.api.BeforeEach;

**import** org.junit.jupiter.api.Nested;

**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 {

@Nested

**class** isDietRecommendedTest {

First **obj**;

@BeforeEach

**void** initialize() {

**obj** = **new** First();

}

@Test

**void** should\_ReturnTrue\_When\_DietRecommended() {

*//given*

**double** weight = 80;

**double** height = 1.5;

*//when*

**boolean** recommended = BMICalculator.*isDietRecommended*(weight, height);

*//then*

*assertTrue*(recommended);

}

@Test

**void** should\_ReturnFalse\_When\_DietNotRecommended() {

*//given*

**double** weight = 40;

**double** height = 1.95;

*//when*

**boolean** recommended = BMICalculator.*isDietRecommended*(weight, height);

*//then*

*assertFalse*(recommended);

}

@Test

**void** should\_Throw\_ArithmeticException\_When\_HeightIsZero() {

*//given*

**double** weight = 40;

**double** height = 0.0;

*//when*

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

*//then*

*assertThrows*(ArithmeticException.**class**,exe);

}

}

@Nested

**class** findCoderWithWorstBMITest {

First **obj**;

@BeforeEach

**void** initialize() {

**obj** = **new** First();

}

@Test

**void** should\_ReturnCoderWithWorstBMI\_When\_CoderListNotEmpty()

{

*//given*

List<Coder> coders = **new** ArrayList<>();

coders.add(**new** Coder(1.80,60.0));

coders.add(**new** Coder(1.82,65.2));

coders.add(**new** Coder(1.82,98.0));

*//when*

Coder coderWorstBMI = BMICalculator.*findCoderWithWorstBMI*(coders);

*//then*

*assertAll*(

() -> *assertEquals*(1.82,coderWorstBMI.getHeight()),

() -> *assertEquals*(98.0,coderWorstBMI.getWeight())

);

}

@Test

**void** should\_ReturnNullWorstBMICoder\_When\_CoderListEmpty()

{

*//given*

List<Coder> coders = **new** ArrayList<>();

*//when*

Coder coderWorstBMI = BMICalculator.*findCoderWithWorstBMI*(coders);

*//then*

*assertNull*(coderWorstBMI);

}

}

@Nested

**class** getBMIScoresTest {

First **obj**;

@BeforeEach

**void** initialize() {

**obj** = **new** First();

}

@Test

**void** should\_returnCorrectBMIScoreArray\_When\_CoderListNotEmpty()

{

*//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**[] bmiscores = BMICalculator.*getBMIScores*(coders);

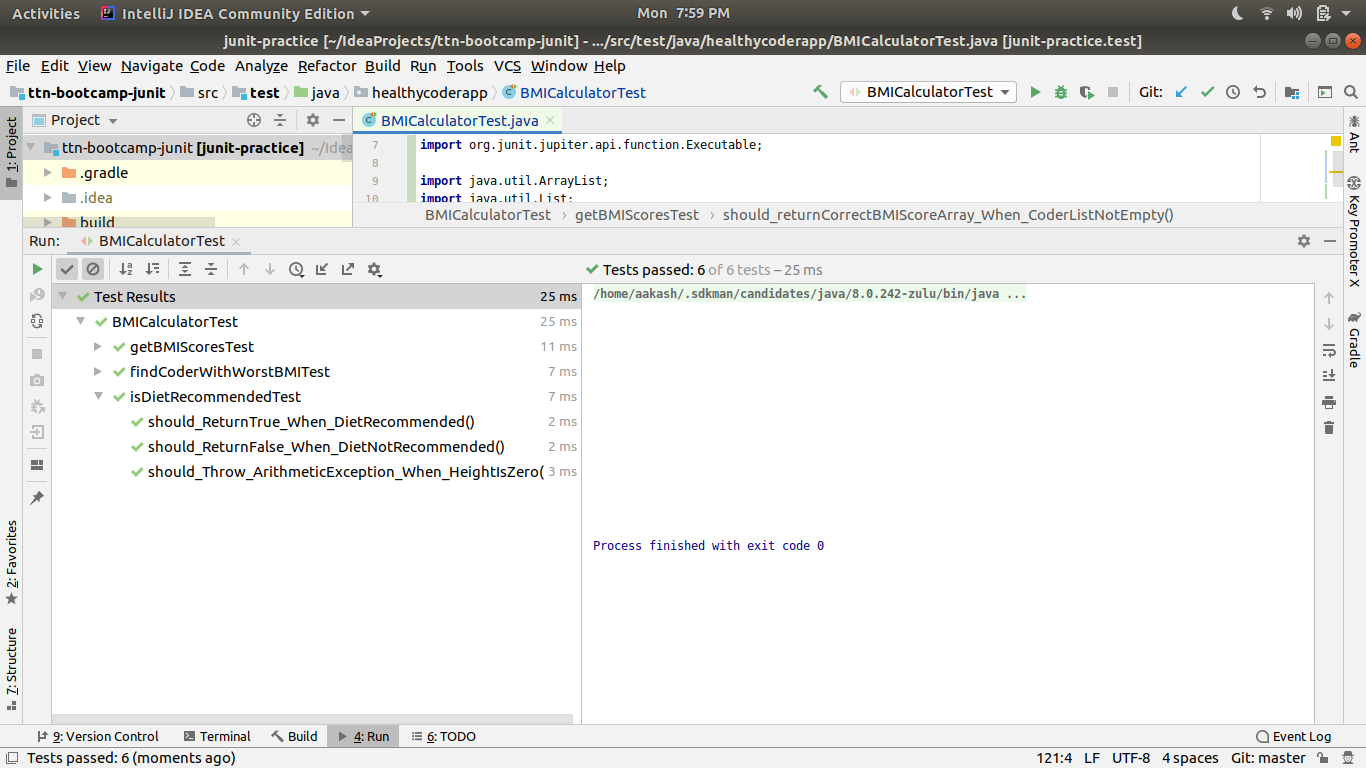
*//then*

*assertArrayEquals*(expected,bmiscores);

}

}

}



*//Ques2.2: Write Unit tests for HealthyCoder app given in the Udemy session. You need to write tests for the BMICalculator and DitePlanner.*

**package** healthycoderapp;

**import** org.junit.jupiter.api.BeforeEach;

**import** org.junit.jupiter.api.Test;

**import static** org.junit.Assert.*assertEquals*;

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

**class** DietPlannerTest {

**private** DietPlanner **dietPlanner**;

@BeforeEach

**void** setup(){

**this**.**dietPlanner** = **new** DietPlanner(20,30,50);

}

@Test

**void** should\_returnCorrect\_DietPlan\_When\_CorrectCoder(){

*//given*

Coder coder = **new** Coder(1.82,75.0,26,Gender.***MALE***);

DietPlan expected = **new** DietPlan(2202,110,73,275);

*//when*

DietPlan actual = **dietPlanner**.calculateDiet(coder);

*//then*

*assertAll*(

() -> *assertEquals*(expected.getCalories(),actual.getCalories()),

() -> *assertEquals*(expected.getProtein(),actual.getProtein()),

() -> *assertEquals*(expected.getCarbohydrate(),actual.getCarbohydrate()),

() -> *assertEquals*(expected.getFat(),actual.getFat())

);

}

}

