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

methods in First.java.

code:-

**package** com.im;

**import** java.lang.reflect.Array;

**import** java.math.BigDecimal;

**import** java.util.ArrayList;

**import** java.util.Iterator;

**import** java.util.List;

**import** java.util.stream.Collectors;

**import** java.util.stream.IntStream;

**import** java.util.stream.Stream;

**public class** First {

**public static void** main(String[] args) {

First first = **new** First();

*//System.out.println(first.replaceSubString("This is my main string text", "main", "modified"));*

*//System.out.println(first.filterEvenElements(IntStream.of(1,2,3,4,5).boxed().collect(Collectors.toList())));*

BigDecimal result = first.calculateAverage(**new** ArrayList<>());

}

*/\*\**

*\* This method is used to replace the part of string with new string.*

*\** ***@param*** *mainString string which needs to be modified*

*\** ***@param*** *subString string which needs to be replaced*

*\** ***@param*** *replacementString string to be replaced with*

*\** ***@return*** *updated string if mainString contains substring, else original string*

*\*/*

**public** 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;

}

}

*/\*\**

*\* This method is used to filter even elements from list.*

*\** ***@param*** *list list of integer*

*\** ***@return*** *list*

*\*/*

**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();

*// Check palindrome string*

**if** (origString.equals(reverseString)) {

isPallindrome = **true**;

}

**return** isPallindrome;

}

}

Whole code of unit Testing :

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=​ "sddsddddddr"​ ;

String subString=​ "dmllkdkkd"​ ;

String replacementString=​ ""​ ;

//when

String srt =First.​ replaceSubString ​ (mainString, subString,replacementString);

//then

assertEquals ​ (srt,​ "sddsddddddr"​ );

}

@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);

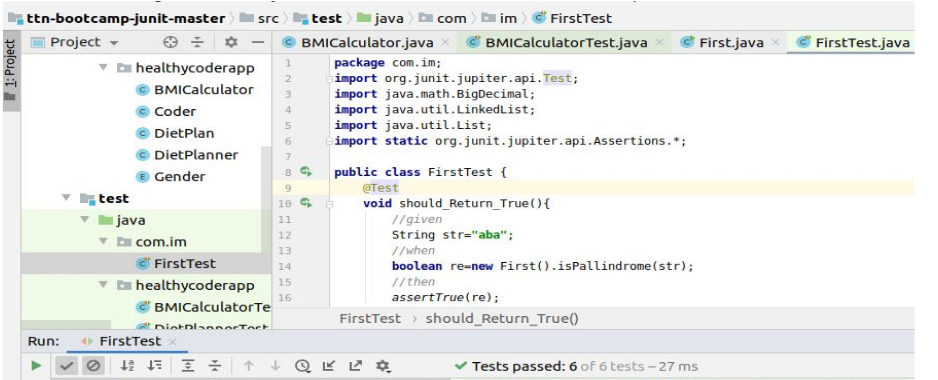
//then

assertEquals ​ (decimal,str);

}

}

Output:-



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

write tests for the BMICalculator and DitePlanner.

Ans:-

Code:-

**package** healthycoderapp;

**import** java.util.Comparator;

**import** java.util.List;

**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(**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;

}

}

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

d

ouble ​ height=​ 1.7​ ;

double ​ weight=​ 79.2​ ;

//when

b

oolean ​ recommend=BMICalculator.​ isDietRecommended ( ​ weight,height);

//then

a

ssertTrue ​ (recommend);

}

@Test

void ​ should\_Return\_False(){

//given

d

ouble ​ height=​ 1.9​ ;

double ​ weight=​ 50.2​ ;

//when

b

oolean ​ recommend=BMICalculator.​ isDietRecommended ( ​ weight,height);

//then

a

ssertFalse ( ​ recommend);

}

@Test

void ​ should\_Return\_Exception\_when\_height\_zero(){

//given

d

ouble ​ height=​ 0.0​ ;

double ​ weight=​ 50.7​ ;

//when

E

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

//then

a

ssertThrows ( ​ 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

C

oder coderWorstBMI=BMICalculator.​ findCoderWithWorstBMI ​ (coders);

//then

a

ssertAll ​ (

()-> ​ assertEquals ( ​ ​ 1.2​ , ​ coderWorstBMI​ .getHeight()),

()-> ​ assertEquals ​ ( ​ 68.0​ , ​ coderWorstBMI​ .getWeight())

);

}

@Test

void ​ should\_Return\_null(){//given

L

ist<Coder> coders=​ new ​ ArrayList<Coder>();​ //no coder element added

//when

C

oder coderNull=BMICalculator.​ findCoderWithWorstBMI ( ​ coders);

//then

assertNull ( ​ coderNull);

}

@Test

void ​ should\_returnCorrectBMIScoreArray(){

//given

L

ist<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

d

ouble​ [] BMIScore= BMICalculator.​ getBMIScores ​ (coders);

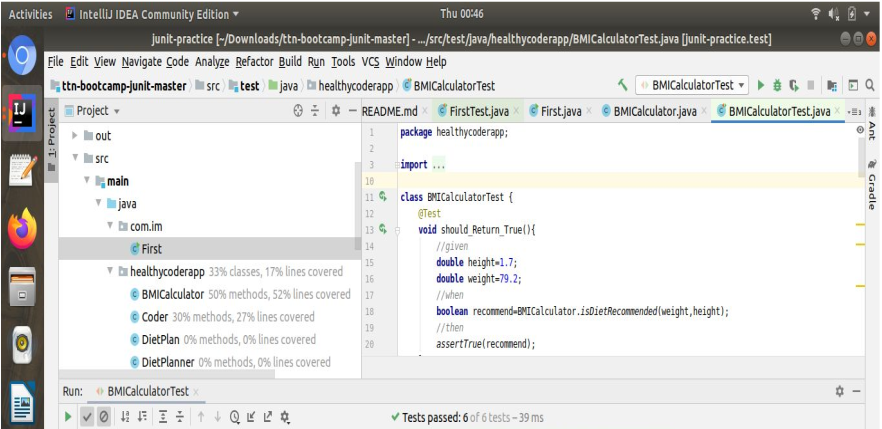
//then

a

ssertArrayEquals ​ (expected,BMIScore);

}

}



Now for file DitePlanner:-

**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) {

**return** (**int**) Math.round(bmr \* proteinPercentage / 400.0);

}

**private int** calculateFat(**int** bmr) {

**return** (**int**) Math.round(bmr \* fatPercentage / 900.0);

}

**private int** calculateCarbohydrate(**int** bmr) {

**return** (**int**) Math.round(bmr \* carbohydratePercentage / 400.0);

}

**private int** calculateBMR(Coder coder) {

**if** (coder.getGender() == Gender.MALE) {

**return** (**int**) Math.round(

(66.5 + 13.8 \* coder.getWeight()

+ 5.0 \* coder.getHeight() \* 100

- 6.8 \* coder.getAge()) \* 1.2

);

}

**return** (**int**) Math.round(

(655.1 + 9.6 \* coder.getWeight()

+ 1.9 \* coder.getHeight() \* 100

- 4.7 \* coder.getAge()) \* 1.2

);

}

}

Test case1:

@Test

void ​ should\_Return\_Exception\_when\_percentage\_isWrong(){

//given

i ​ nt ​ proteinPercentage=​ 20​ ;

int ​ fatPercentage=​ 50​ ;}

int ​ carbohydratePercentage = ​ 10​ ;

//when

E

xecutable executable=()-> ​ new ​ DietPlanner(​ proteinPercentage​ , ​ fatPercentage​ , ​ carbohydratePercentage​ );

//then

a

ssertThrows ( ​ RuntimeException.​ class​ , executable);