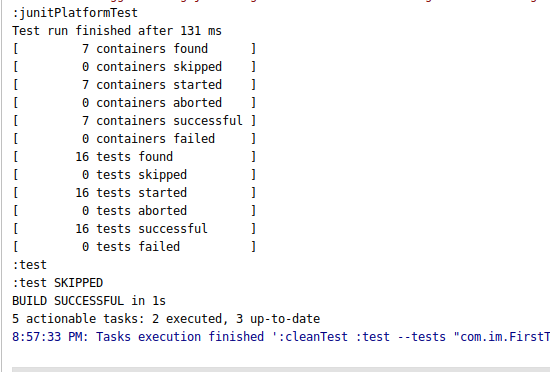
Unit Testing & Clean Code

Exercise

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



Code:

**package** com.im;

**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.CsvFileSource;

**import** java.math.BigDecimal;

**import** java.util.ArrayList;

**import** java.util.Arrays;

**import** java.util.List;

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

**public class** FirstTest {

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

@ParameterizedTest

@CsvFileSource(resources = **"/data.csv"**)

**void** should\_replaceSubStrings(String main,String sub,String replace){

*//given*

String main1 = main;

String sub1 = sub;

String replace1 = replace;

String required = **"helil"**;

*//when*

String result = **first**.replaceSubString(main1,sub1,replace1);

*//then*

*assertEquals*(result,required);

}

@ParameterizedTest

@CsvFileSource(resources = **"/data1.csv"**)

**void** should\_notAcceptNull(String main\_null,String sub\_null,String replace\_null){

*//given*

String main2 = main\_null;

String sub2 = sub\_null;

String replace2 = replace\_null;

*//when*

String result = **first**.replaceSubString(main2,sub2,replace2);

*//then*

*assertEquals*(result,main2);

}

@Test

**void** should\_filter\_even\_elements(){

*//given*

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

List<Integer> result = Arrays.*asList*(1,3,5,7);

list.add(1);

list.add(2);

list.add(3);

list.add(4);

list.add(5);

list.add(6);

list.add(7);

list.add(8);

*//when*

List<Integer> compare = **first**.filterEvenElements(list);

*//then*

*assertEquals*(compare,result);

}

@Test

**void** should\_calculateAverage(){

*//given*

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

list.add(**new** BigDecimal(**"1.2"**));

list.add(**new** BigDecimal(**"3.4"**));

list.add(**new** BigDecimal(**"2.3"**));

list.add(**new** BigDecimal(**"3.1"**));

BigDecimal result = **new** BigDecimal(**"2.5"**);

*//when*

BigDecimal compare = **first**.calculateAverage(list);

*//then*

*assertEquals*(compare,result);

}

@Test

**void** should\_notAcceptNullValues(){

*//given*

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

*//when*

Executable executable = ()->**first**.calculateAverage(list);

*//then*

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

}

@Test

**void** should\_checkPalindromeCondition\_true(){

*//given*

String str = **"madam"**;

*//when*

**boolean** expect = **first**.isPallindrome(str);

*//then*

*assertTrue*(expect);

}

@Test

**void** should\_checkPalindromeCondition\_false(){

*//given*

String str = **"whatsup"**;

*//when*

**boolean** expect = **first**.isPallindrome(str);

*//then*

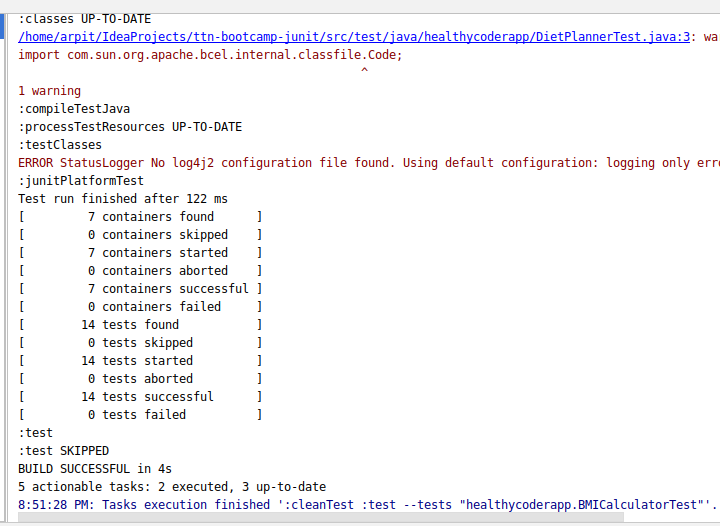
*assertFalse*(expect);

}

}

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

1. BMICalculator:-



Code:

**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\_Returntrue\_WhenDietRecommended(){

*//given*

**double** weight = 70.5;

**double** height = 1.65;

*//when*

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

*//then*

*assertTrue*(recommended);

}

@Test

**void** should\_ReturnFalse\_WhenDietNotRecommended(){

*//given*

**double** weight = 100.0;

**double** height = 2.5;

*//when*

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

*//then*

*assertFalse*(recommended);

}

@Test

**void** should\_Throw\_ArithmeticExpression\_when\_heightZero(){

*//given*

**double** weight = 76.0;

**double** height = 0.0;

*//when*

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

*//then*

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

}

@Test

**void** should\_Return\_CoderwithworstBMI\_whenlistNotEmpty(){

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

*//Given*

coder.add(**new** Coder(1.70,58));

coder.add(**new** Coder(1.65,70));

coder.add(**new** Coder(1.8,100));

*//when*

Coder coder\_with\_worstBMI = BMICalculator.*findCoderWithWorstBMI*(coder);

*//then*

*assertAll*(

()-> *assertEquals*(1.8,coder\_with\_worstBMI.getHeight()),

()->*assertEquals*(100,coder\_with\_worstBMI.getWeight())

);

}

@Test

**void** should\_return\_null\_whenlistEmpty(){

*//given*

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

*//when*

Coder coder\_with\_worstBMI = BMICalculator.*findCoderWithWorstBMI*(coder);

*//then*

*assertNull*(coder\_with\_worstBMI);

}

@Test

**void** should\_getBMIscores\_whencoderlistnotempty()

{

*//given*

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

coder.add(**new** Coder(1.8,98));

coder.add(**new** Coder(1.75,80));

**double**[] actual\_BMI = **new double**[]{30.25,26.12};

*//when*

**double**[] list\_coders = BMICalculator.*getBMIScores*(coder);

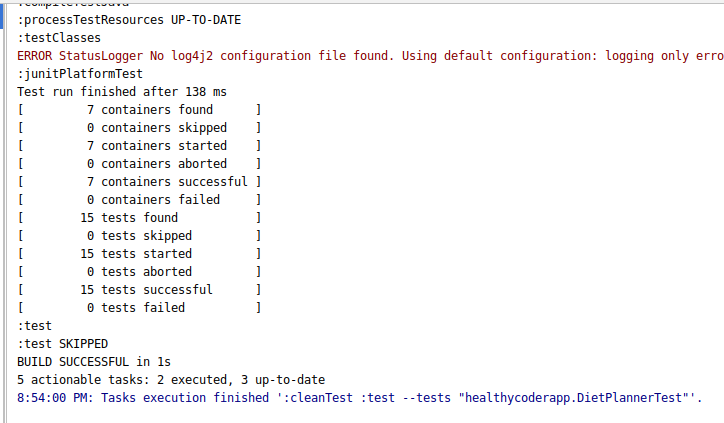
*//then*

*assertArrayEquals*(actual\_BMI,list\_coders);

}

}

1. DietPlanner:-



Code:

**package** healthycoderapp;

**import** com.sun.org.apache.bcel.internal.classfile.Code;

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

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

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

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

**public class** DietPlannerTest {

DietPlanner **dietPlanner**;

@BeforeEach

**void** setup(){

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

}

@Test

**void** should\_calculateDietFor\_positiveResult(){

*//given*

Coder coder = **new** Coder(1.82,75.0,21,Gender.***MALE***);

DietPlan expected = **new** DietPlan(2242,112,75,280);

*//when*

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

*//then*

*assertAll*(

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

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

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

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

);

}

@Test

**void** shouldReturnFalse\_WhenWrongDietPlan\_WhenBadCoder(){

*//given*

Coder coder = **new** Coder(1.82, 98.0,20, Gender.***MALE***);

DietPlan expected = **new** DietPlan(2631, 132, 88, 329);

*//when*

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

*//then*

*assertAll*(

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

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

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

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

);

}

}