THEORETICAL EXERCISE 2

GRAVITATIONAL EQUATION PROBLEM

GROUP A03

1. Pseudocode of the identified methods

GravitationalEquation class:

```java
package es.uclm.esi.iso2.ga03.GravitationalEq;

public class GravitationalEquation {

    private static final double GRAVITATIONAL_CONSTANT = 6.67430e-11;

    private double mass1;
    private double mass2;
    private double distance;

    public GravitationalEquation(double mass1, double mass2, double distance) {
        this.setMass1(mass1);
        this.setMass2(mass2);
        this.setDistance(distance);
    }

    public double calculateGravitationalForce() {
        return (GRAVITATIONAL_CONSTANT * mass1 * mass2) / Math.pow(distance, 2);
    }

    public double getMass1() {
        return mass1;
    }

    public void setMass1(double mass1) {
        this.mass1 = mass1;
    }

    public double getMass2() {
        return mass2;
    }

    public void setMass2(double mass2) {
        this.mass2 = mass2;
    }

    public double getDistance() {
        return distance;
    }

    public void setDistance(double distance) {
        this.distance = distance;
    }

    public String toString() {
        return "GravitationalEquation [mass1=" + this.getMass1() + ", mass2=" + this.getMass2() + ", distance="
                + this.getDistance() + "]";
    }

}
```

Main class:

```java
package es.uclm.esi.iso2.ga03.GravitationalEq;

import java.util.InputMismatchException;
import java.util.Scanner;

public class Main {
    final static Scanner scanner = new Scanner(System.in);
    public static void main(String[] args) {
        try {
            printResult(askValues());

        } catch (IllegalArgumentException e) {
            System.out.println("Error: " + e.getMessage());
        }
    }

    public static void printResult(double gravitationalForce) {
        System.out.println("Gravitational Force: " + gravitationalForce + " Newtons");
    }

    public static double askValues() {
         double mass1 = returnValues("Enter the mass of the first object: ");
         double mass2 = returnValues("Enter the mass of the second object: ");
         double distance = returnValues("Enter the distance between the objects: ");
         GravitationalEquation gravitationalEquation = new GravitationalEquation(mass1, mass2, distance);
         return calculateResult(gravitationalEquation);
    }

    public static double returnValues(String task) {
        double value = 0.0;
            try {
                System.out.print(task);
                value = scanner.nextDouble();
                checkPositiveInput(value);
            } catch (InputMismatchException e) {
                System.err.println("Error: Enter a valid number.");
                scanner.next();
            } catch (IllegalArgumentException e) {
                System.err.println("Error: " + e.getMessage());
            }
            return value;
        }

    public static void checkPositiveInput(double input) {
        if (isPositive(input) == false) {
            throw new IllegalArgumentException("Enter a positive number greater than zero.");
        }
    }

    public static boolean isPositive(double input) {
        if (input <= 0) {
            return false;
        }
        else {
            return true;
        }
    }

    public static double calculateResult(GravitationalEquation gravitationalEquation) {
        return gravitationalEquation.calculateGravitationalForce();
    }

}
```

2. Identifying variables

In order to test the method for the gravitational equation, we must check the following variables:

- mass1
- mass2
- distance

3. Identifying test values

Parameter	Equivalence Classes	Values Error-guessing	Boundary values (Light variant)
Mass1	(-∞,0] (0, ∞)	-5 1.23 5 aaaa	0 1
Mass2	(-∞,0] (0, ∞)	-5 2.89 90 abba	0 1
Distance	(-∞,0] (0, ∞)	-5 8000.64 59991 aacc	0 1

4. Maximum number of test cases

The possible test cases are 5 per each variable, in total: 6*6*6 = 216

5. Set of test cases

Each-use:

Case1: {-5,-5,-5}
Case2: {1.23,2.89,8000.64}
Case3: {5,90,59991}
Case4: {aaaa,abba,aacc}
Case5: {0,0,0}
Case6: {1,1,1}

6. Pairwise Testing

We have used the following website to calculate it: Pairwise Pict Online (https://pairwise.yuuniworks.com/). After that we obtained the following combinations:

Mass1	Mass2	Distance
1	abba	8000.64
-5	90	-5
-5	2.89	0
1	2.89	aacc
1.23	2.89	8000.64
0	2.89	-5
0	-5	aacc
aaaa	abba	59991
1.23	0	0
0	1	1
0	abba	0
-5	-5	1
1.23	90	1
aaaa	90	aacc
-5	abba	aacc
0	0	8000.64
aaaa	2.89	1
1.23	1	aacc
aaaa	-5	-5
1	-5	0
-5	0	59991
1.23	-5	59991
-5	1	8000.64
1	0	-5
1	90	1
5	90	8000.64
1	1	59991
5	0	1
5	0	aacc
5	2.89	59991
5	-5	-5
1.23	abba	-5
0	90	59991
aaaa	-5	8000.64
5	abba	1
5	1	0
aaaa	1	0
5	90	0
aaaa	1	-5
aaaa	0	1

7. Decision Coverage

The condition we are going to test is the one condition in the **checkPositiveInput** method. This method is controlled by the method isPositive

A: isPositive

Condition	Decision	Dominant
A	A	
true	false	A
false	true	A

8. MC/DC coverage

As the code is very simplified, the set of test cases to achieve MC/DC coverage is the same as the one used to achieve coverage of decisions.

9. Final comments

The following capture shows the coverage of the JUnit tests. As we can see, it has a 79% of coverage. This means that it is almost covered all methods.

GravitationalEq

Element	Missed Instructions	Cov.	Missed Branches	Cov.	Missed	Cxty	Missed	Lines	Missed	Methods	Missed	Classes
es.uclm.esi.iso2.ga03.GravitationalEq		79%		100%	2	20	10	49	2	18	0	2
Total	37 of 178	79%	0 of 4	100%	2	20	10	49	2	18	0	2