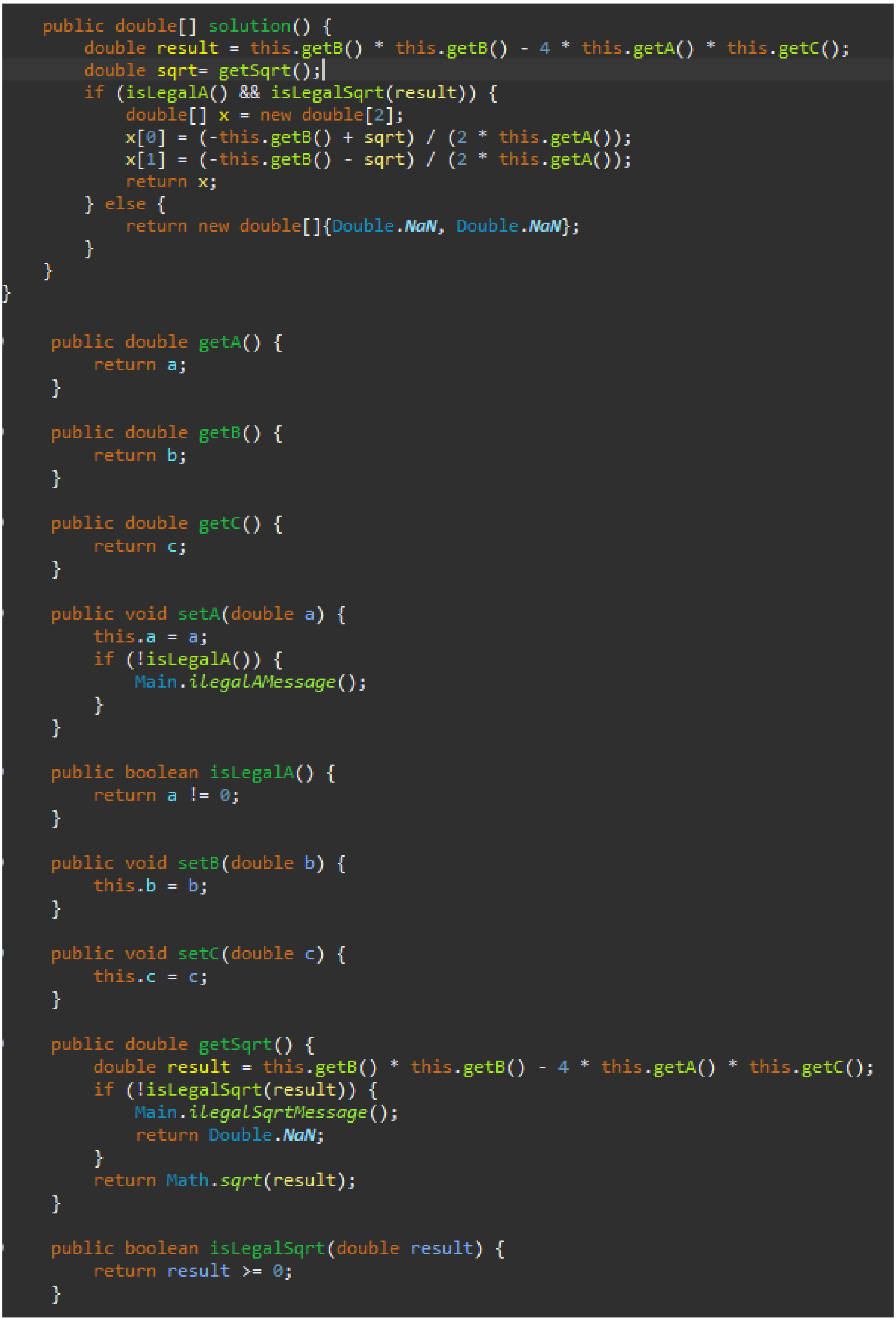
THEORETICAL EXERCISE 2

QUADRATIC EQUATION PROBLEM

GROUP A03

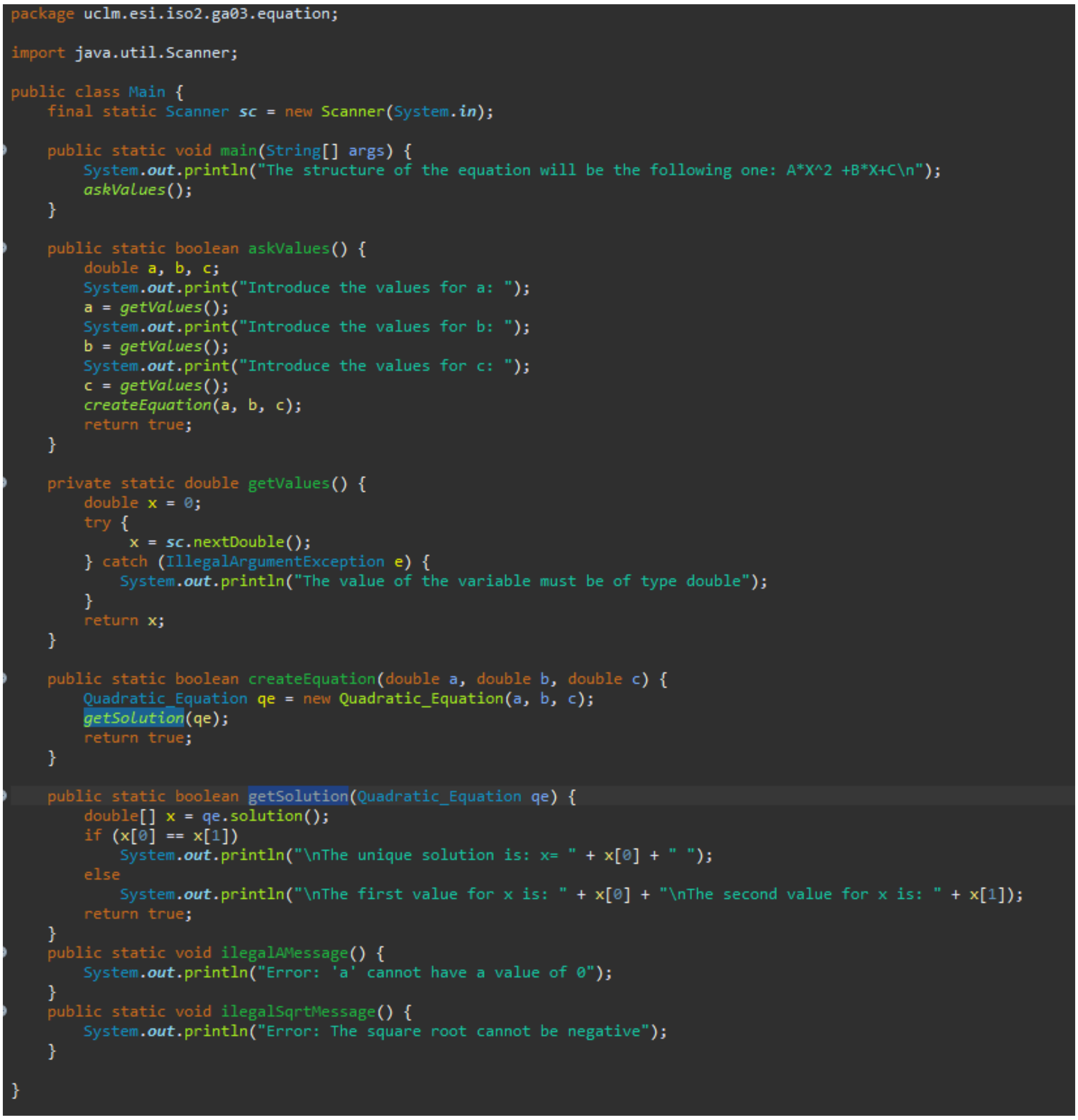
1. Pseudocode of the identified methods

Quadratic equation class:



```
    public double[] solution() {
        double result = this.getB() * this.getB() - 4 * this.getA() * this.getC();
        double sqrt= getSqrt();
        if (isLegalA() && isLegalSqrt(result)) {
            double[] x = new double[2];
            x[0] = (-this.getB() + sqrt) / (2 * this.getA());
            x[1] = (-this.getB() - sqrt) / (2 * this.getA());
            return x;
        } else {
            return new double[]{Double.NaN, Double.NaN};
        }
    }
}


    public double getA() {
        return a;
    }

    public double getB() {
        return b;
    }

    public double getC() {
        return c;
    }

    public void setA(double a) {
        this.a = a;
        if (!isLegalA()) {
            Main.ilegalAMessage();
        }
    }

    public boolean isLegalA() {
        return a != 0;
    }

    public void setB(double b) {
        this.b = b;
    }

    public void setC(double c) {
        this.c = c;
    }

    public double getSqrt() {
        double result = this.getB() * this.getB() - 4 * this.getA() * this.getC();
        if (!isLegalSqrt(result)) {
            Main.ilegalSqrtMessage();
            return Double.NaN;
        }
        return Math.sqrt(result);
    }

    public boolean isLegalSqrt(double result) {
        return result >= 0;
    }
```

Main class:



```java
package uclm.esi.iso2.ga03.equation;

import java.util.Scanner;

public class Main {
    final static Scanner sc = new Scanner(System.in);

    public static void main(String[] args) {
        System.out.println("The structure of the equation will be the following one: A*X^2 +B*X+C\n");
        askValues();
    }

    public static boolean askValues() {
        double a, b, c;
        System.out.print("Introduce the values for a: ");
        a = getValues();
        System.out.print("Introduce the values for b: ");
        b = getValues();
        System.out.print("Introduce the values for c: ");
        c = getValues();
        createEquation(a, b, c);
        return true;
    }

    private static double getValues() {
        double x = 0;
        try {
             x = sc.nextDouble();
        } catch (IllegalArgumentException e) {
            System.out.println("The value of the variable must be of type double");
        }
        return x;
    }

    public static boolean createEquation(double a, double b, double c) {
        Quadratic_Equation qe = new Quadratic_Equation(a, b, c);
        getSolution(qe);
        return true;
    }

    public static boolean getSolution(Quadratic_Equation qe) {
        double[] x = qe.solution();
        if (x[0] == x[1])
            System.out.println("\nThe unique solution is: x= " + x[0] + " ");
        else
            System.out.println("\nThe first value for x is: " + x[0] + "\nThe second value for x is: " + x[1]);
        return true;
    }
    public static void ilegalAMessage() {
        System.out.println("Error: 'a' cannot have a value of 0");
    }
    public static void ilegalSqrtMessage() {
        System.out.println("Error: The square root cannot be negative");
    }

}
```

2. Identifying variables

The variables which can be checked at the testing are:

a, b, c

3. Identifying test values

Parameter	Equivalence class	Values	Boundary values
Quadratic_Equation.getA()	[-∞,0) (0,∞]	-12 10	-1 1
Quadratic_Equation.getB()	[-∞,∞] if b^2 > 4*a*c	30	-
Quadratic_Equation.getC()	[-∞,∞] if 4*a*c < b^2	5	-

4. Maximum number of test cases

The maximum number of test cases we will have is 4 [(2+2)*1*1].

5. Set of test cases

{Quadratic_Equation.getA(), Quadratic_Equation.getB(), Quadratic_Equation.getC()}

CP1: {-12,30,5}

CP2: {10,30,5}

CP3: {-1,30,5}

CP4: {1,30,5}

6. Pairwise Testing

getA()	getB()	getC()
-1	30	5
10	30	5
1	30	5
-12	30	5

7. Decision Coverage

a<>0 AND (b^2) > (4*a*c)

A = a<>0

B = (b^2) > (4*a*c)

C = a<>0 AND (b^2) > (4*a*c)

CONDITIONS		**DECISION**	
A	B	C	**DOMINANT**
F	F	F	A,B
F	T	F	A
T	F	F	B
T	T	T	A, B

TEST CASES BASED ON DECISIONS (The ones in orange are selected)

Quadratic_Equation.getA()	Quadratic_Equation.getB()	Quadratic_Equation.getC()	RESULT
1	8	-4	True
2	2	10	False

8. MC/DC coverage

a<>0 AND (b^2) > (4*a*c)

A = a<>0

B = (b^2) > (4*a*c)

C = a<>0 AND (b^2) > (4*a*c)

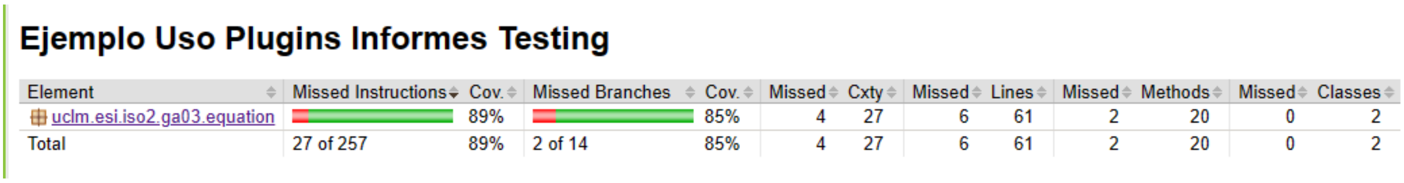
CONDITIONS		DECISION	
A	B	C	DOMINANT
F	F	F	A,B
F	T	F	A
T	F	F	B
T	T	T	A, B

Test Cases

Quadratic_Equation.getA()	Quadratic_Equation.getB()	Quadratic_Equation.getC()	RESULT
-3	-9	7	True
8	2	10	False
0	5	-2	False

9. Final comments

This photo represents the coverage of the Junit test. It has a coverage of 89%, which means that we have coverage of almost all methods and the results are the ones which were expected.



Ejemplo Uso Plugins Informes Testing

Element	Missed Instructions	Cov.	Missed Branches	Cov.	Missed	Cxty	Missed	Lines	Missed	Methods	Missed	Classes
uclm.esi.iso2.ga03.equation		89%		85%	4	27	6	61	2	20	0	2
Total	27 of 257	89%	2 of 14	85%	4	27	6	61	2	20	0	2