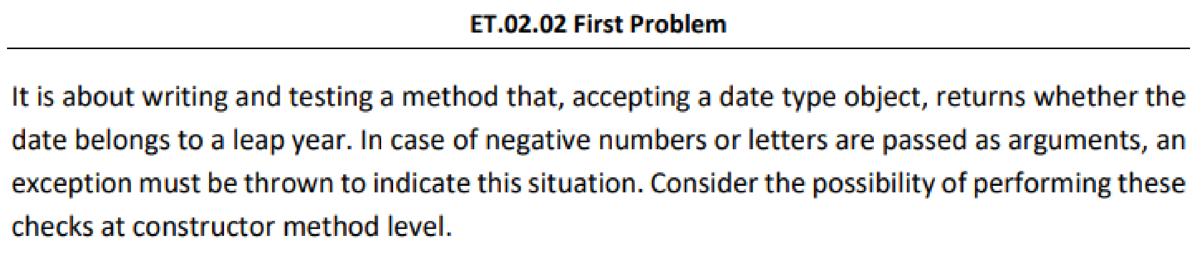
Theoretical Exercise - Problem #1

Distribution of work
Pseudocode: Team 1: Alejandro, Alejandro
Testing: Team 2: Alberto, Andrés

0 - Problem Description



ET.02.02 First Problem

It is about writing and testing a method that, accepting a date type object, returns whether the date belongs to a leap year. In case of negative numbers or letters are passed as arguments, an exception must be thrown to indicate this situation. Consider the possibility of performing these checks at constructor method level.

1- Write, at least the pseudocode of the identified method.

```
public class Date {

    int day;
    int month;
        int year;

        public Date(int day, int month, int year) throws NumberException  {
                this.day = day;
                if (this.day<1 || this.day>31) {
                        throw new NumberException();
                }
                this.month = month;
                if (this.day<1 || this.day>12) {
                        throw new NumberException();
                }
                this.year = year;
                if (this.year<0) {
                        throw new NumberException();
                }
    }

        public void setDay(int day) {
                this.day = day;

        }

        public void setMonth(int month) {
                this.month = month;
        }

        public void setYear(int year) {
                this.year = year;
        }

        public int getDay() {
                return day;
        }

        public int getMonth() {
        return month;
        }

        public int getYear() {
            return year;
        }

    boolean isLeap() {
        boolean leap = false;
                int year = getYear();
                if ((year % 4 == 0) && (year %  100 != 0) || (year % 400 == 0)){
                        leap = true;
                }
                return leap;
      }
}
```

2 - Identify the variables that must be considered to test the method.

For this problem we will be focusing on the method "isLeap()", which is the one that tells the information that we want. For this we will be focusing on the following variable:

- year: we should take into account the years as it will be introduced in the method in order to know if the year is Leap or not.

It is also important to make sure all of the values are valid, if not we could have a non expected result.

3 - Identify the test values for each one of the variables previously identified, specifying the technique used to obtain each of those values).

To perform the test cases, we will be using the **equivalence class**, **values** and **boundary values** for the years.

As we are talking about years, we will only take values as valid from 1582, as this is the date where Gregorian Calendar was implemented.

So our table would be like this:

Parameter	Equivalence class	Values from equivalence class	Boundary values (heavy variant)	Error guessing values
isLeap()	[-∞, 0) [1, 1581] [1582, ∞]	-1 1000 2000	0 1, 1581 1582	-20,0 1, 1500
day	[-∞, 0) [1, 15), [15, 31), [31, ∞)	-2147483648, 0, 1, 15, 31, 2147483648	-2147483649, -2147483648, -2147483649, -1, 0, 1, 2, 14, 15, 16, 30, 31, 32, 2147483647, 2147483648, 2147483649	0, 32, 33

month	[-∞, 0] [1, 12] [12, ∞]	-2147483648, 0, 1, 12, 2147483648	-2147483649, -2147483648, -2147483649, -1, 0, 1, 2, 11, , 12, 13,2147483647, 2147483648, 2147483649	0, 13
year	[-∞, ∞]	-2147483648, 0, 1, 1581, 1582, 2147483648	-2147483649, -2147483648, -2147483649, -1, 0, 1, 2, 1580,1581,1582,1583,2147483647, 2147483648, 2147483649	

4 - Calculate the maximum possible number of test cases that could be generated from the test values.

The possible combinations of test cases could be:

- For the years: (-20, -1, 0, 1, 1000, 1500, 1581, 1582, 1755, 1967, 2000, 2020, 2050, 2100).

  So the total number of test values is: 15.

* NOTE * : take into account that in the code, the condition to accept the year value is that it should be bigger than 0, so it is also tested. However, the results would be the expected ones as of 1582 where the Gregorian Calendar was implemented (and there is a proof that is true).

5 - Define some test suites using each use

We could be taking the following cases:
Model is this way: Date (day, month, year).
We use only the year to know if the year is a leap or not.

- Value divisible by 4, 400 but not by 100.
- Value non-divisible by 4, 400 but by 100.

Whenever one of the two conditions is met, then we can say that the year is a leap year.

6 - Define test suits to achieve pairwise coverage by using the proposed algorithm in lectures. You can check the results by means of the software PICT1

To this part we have done a class that will generate the results, that classe is called 'DateTestPairWaise"

7 - For code snippets that include decisions, propose a set of test cases to achieve coverage of decisions.

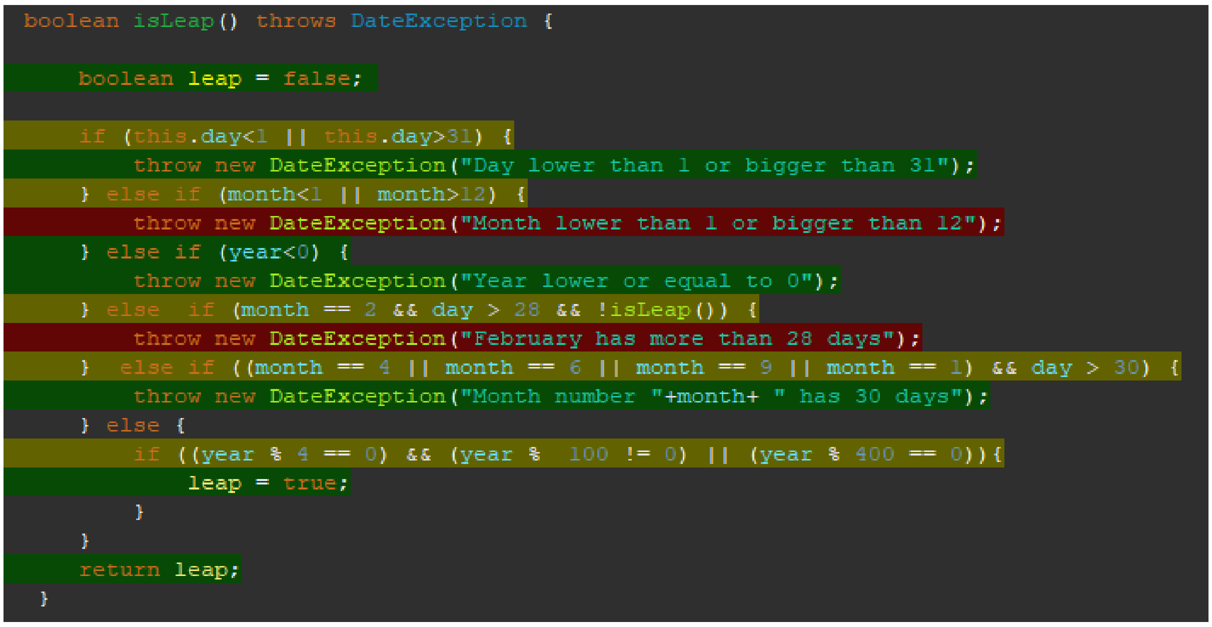
To this part we have done a class that will generate the results, that classe is called ‘DateTestDecision”

8 - For code snippets that include decisions, propose test case sets to achieve MC/DC coverage.

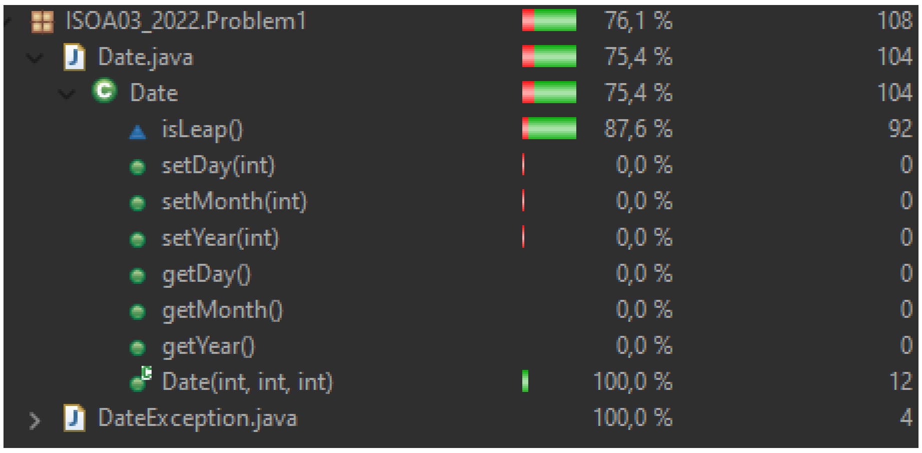
IsLeap()				
Conditions			Decision	
year % 4 = 0	year % 100 != 0	year % 400 = 0	(A AND !B) OR C	Dominant Condition
"TRUE"	"TRUE"	"TRUE"	YES	A,B,C
"TRUE"	"TRUE"	"FALSE"	YES	A,B
"TRUE"	"FALSE"	"TRUE"	YES	C
"TRUE"	"FALSE"	"FALSE"	NO	B,C
"FALSE"	"TRUE"	"TRUE"	YES	C
"FALSE"	"TRUE"	"FALSE"	NO	A,C
"FALSE"	"FALSE"	"TRUE"	YES	C
"FALSE"	"FALSE"	"FALSE"	NO	A,B,C

9 - Comment on the results of the number of test cases obtained in section 4, 5, and 6, as well as the execution of the oracles: what could be said about the coverage achieved?

9.1. Pair Waise Coverage

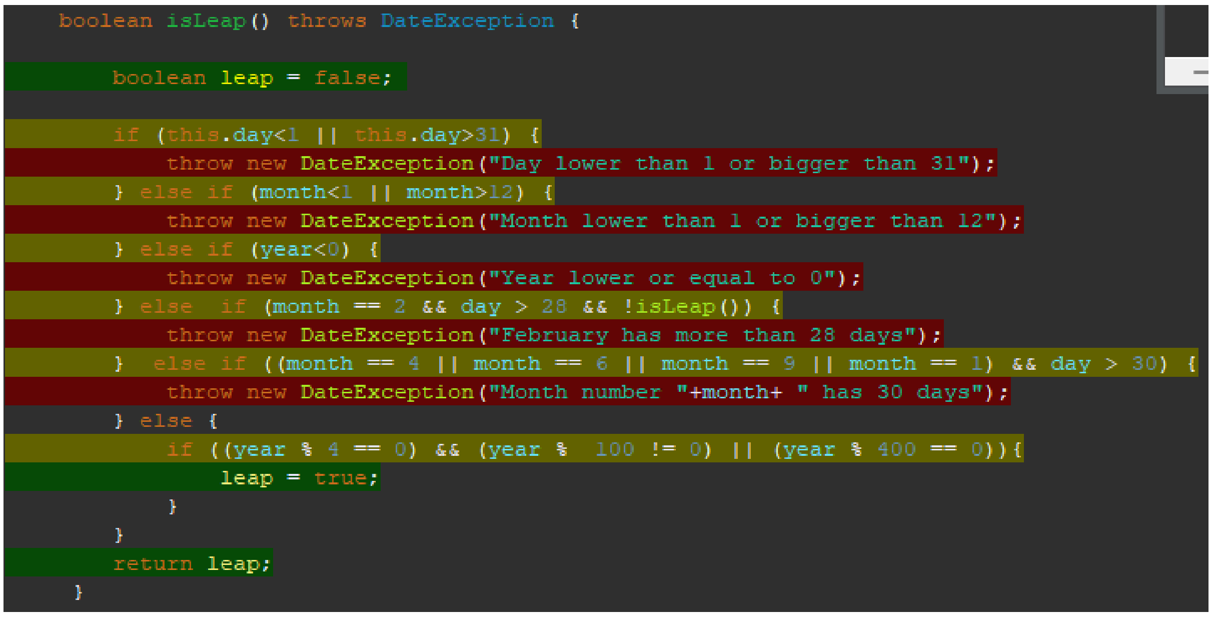


```
boolean isLeap() throws DateException {

    boolean leap = false;

    if (this.day<1 || this.day>31) {
        throw new DateException("Day lower than 1 or bigger than 31");
    } else if (month<1 || month>12) {
        throw new DateException("Month lower than 1 or bigger than 12");
    } else if (year<0) {
        throw new DateException("Year lower or equal to 0");
    } else  if (month == 2 && day > 28 && !isLeap()) {
        throw new DateException("February has more than 28 days");
    }  else if ((month == 4 || month == 6 || month == 9 || month == 1) && day > 30) {
        throw new DateException("Month number "+month+ " has 30 days");
    } else {
        if ((year % 4 == 0) && (year %  100 != 0) || (year % 400 == 0)){
            leap = true;
        }
    }
    return leap;
 }
```

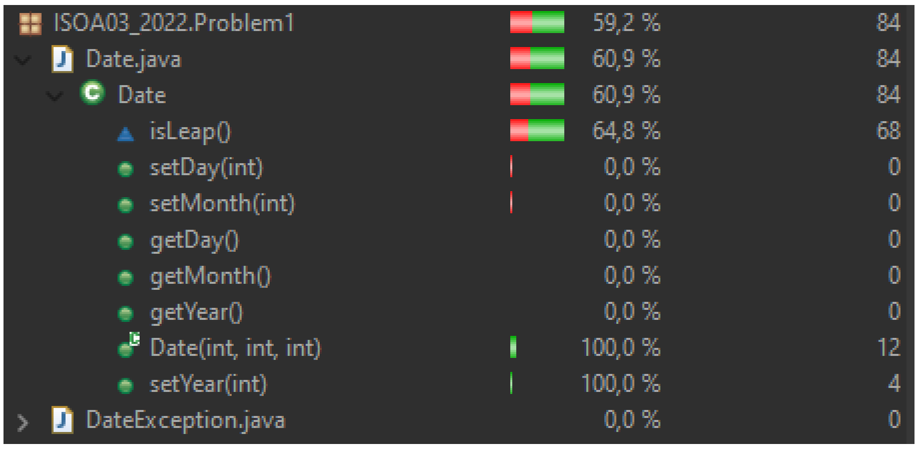


```
ISOA03_2022.Problem1      76,1 %   108
  Date.java               75,4 %   104
    Date                  75,4 %   104
      isLeap()            87,6 %    92
      setDay(int)          0,0 %     0
      setMonth(int)        0,0 %     0
      setYear(int)         0,0 %     0
      getDay()             0,0 %     0
      getMonth()           0,0 %     0
      getYear()            0,0 %     0
      Date(int, int, int) 100,0 %   12
  DateException.java     100,0 %     4
```

9.2. Decision Coverage

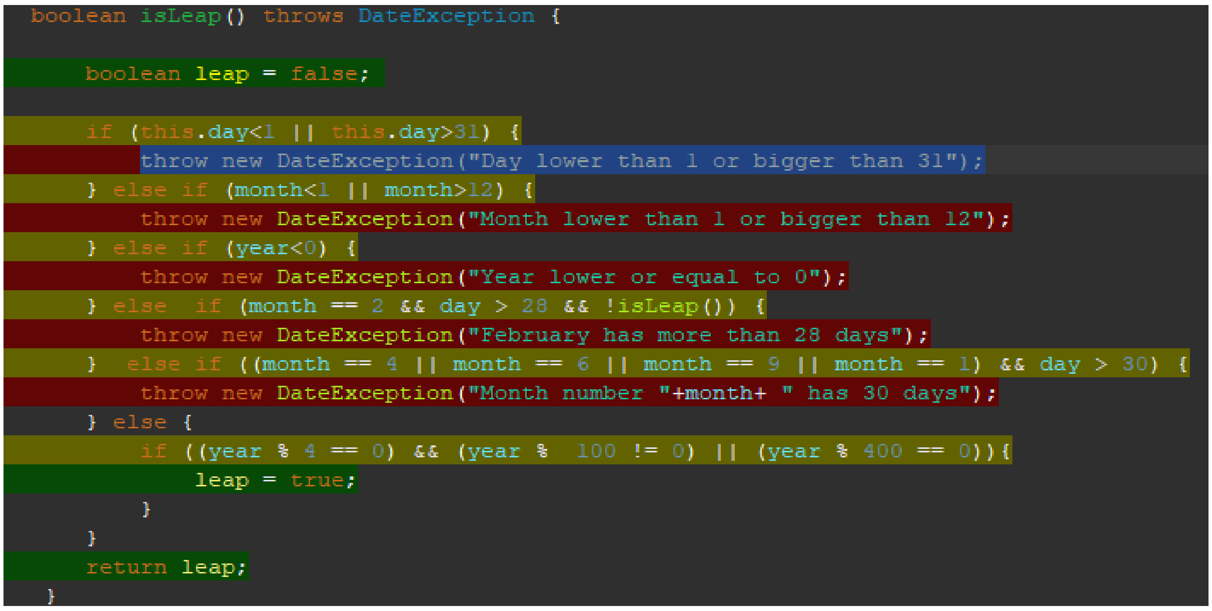


```java
    boolean isLeap() throws DateException {

        boolean leap = false;

        if (this.day<1 || this.day>31) {
            throw new DateException("Day lower than 1 or bigger than 31");
        } else if (month<1 || month>12) {
            throw new DateException("Month lower than 1 or bigger than 12");
        } else if (year<0) {
            throw new DateException("Year lower or equal to 0");
        } else  if (month == 2 && day > 28 && !isLeap()) {
            throw new DateException("February has more than 28 days");
        }  else if ((month == 4 || month == 6 || month == 9 || month == 1) && day > 30) {
            throw new DateException("Month number "+month+ " has 30 days");
        } else {
            if ((year % 4 == 0) && (year %  100 != 0) || (year % 400 == 0)){
                leap = true;
            }
        }
        return leap;
    }
```

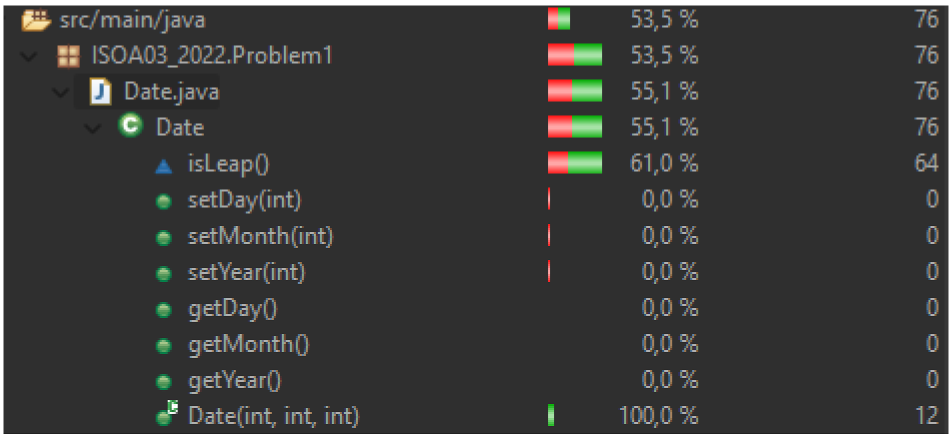


```
ISOA03_2022.Problem1      59,2 %    84
  Date.java               60,9 %    84
    Date                  60,9 %    84
      isLeap()            64,8 %    68
      setDay(int)          0,0 %     0
      setMonth(int)        0,0 %     0
      getDay()             0,0 %     0
      getMonth()           0,0 %     0
      getYear()            0,0 %     0
      Date(int, int, int) 100,0 %   12
      setYear(int)        100,0 %    4
  DateException.java       0,0 %     0
```

9.3. Each Use

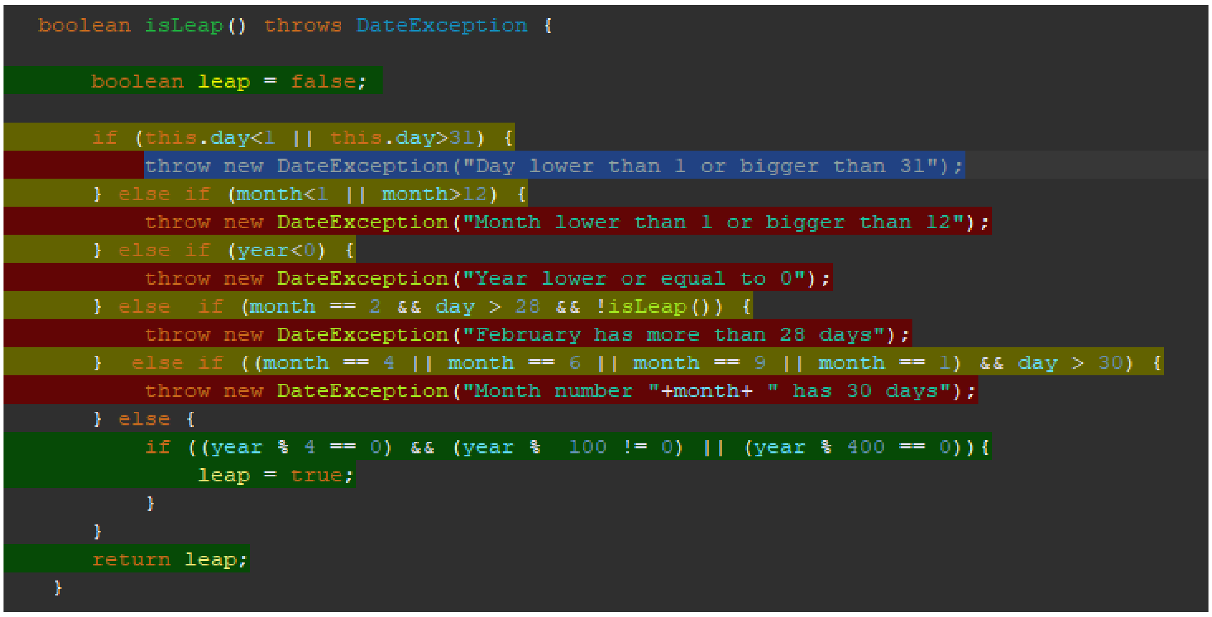


```
boolean isLeap() throws DateException {

    boolean leap = false;

    if (this.day<1 || this.day>31) {
        throw new DateException("Day lower than 1 or bigger than 31");
    } else if (month<1 || month>12) {
        throw new DateException("Month lower than 1 or bigger than 12");
    } else if (year<0) {
        throw new DateException("Year lower or equal to 0");
    } else  if (month == 2 && day > 28 && !isLeap()) {
        throw new DateException("February has more than 28 days");
    }  else if ((month == 4 || month == 6 || month == 9 || month == 1) && day > 30) {
        throw new DateException("Month number "+month+ " has 30 days");
    } else {
        if ((year % 4 == 0) && (year %  100 != 0) || (year % 400 == 0)){
            leap = true;
        }
    }
    return leap;
 }
```

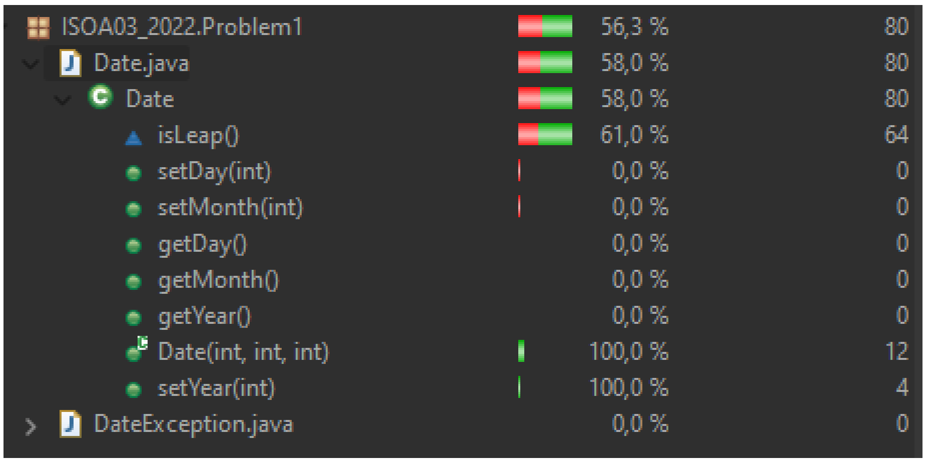


src/main/java	53,5 %	76
ISOA03_2022.Problem1	53,5 %	76
Date.java	55,1 %	76
Date	55,1 %	76
isLeap()	61,0 %	64
setDay(int)	0,0 %	0
setMonth(int)	0,0 %	0
setYear(int)	0,0 %	0
getDay()	0,0 %	0
getMonth()	0,0 %	0
getYear()	0,0 %	0
Date(int, int, int)	100,0 %	12

9.4. MC/DC



```
boolean isLeap() throws DateException {

    boolean leap = false;

    if (this.day<1 || this.day>31) {
        throw new DateException("Day lower than 1 or bigger than 31");
    } else if (month<1 || month>12) {
        throw new DateException("Month lower than 1 or bigger than 12");
    } else if (year<0) {
        throw new DateException("Year lower or equal to 0");
    } else  if (month == 2 && day > 28 && !isLeap()) {
        throw new DateException("February has more than 28 days");
    }  else if ((month == 4 || month == 6 || month == 9 || month == 1) && day > 30) {
        throw new DateException("Month number "+month+ " has 30 days");
    } else {
        if ((year % 4 == 0) && (year %  100 != 0) || (year % 400 == 0)){
            leap = true;
        }
    }
    return leap;
 }
```



```
ISOA03_2022.Problem1        56,3 %    80
  Date.java                 58,0 %    80
    Date                    58,0 %    80
      isLeap()              61,0 %    64
      setDay(int)            0,0 %     0
      setMonth(int)          0,0 %     0
      getDay()               0,0 %     0
      getMonth()             0,0 %     0
      getYear()              0,0 %     0
      Date(int, int, int)  100,0 %    12
      setYear(int)         100,0 %     4
  DateException.java         0,0 %     0
```