Laboratory unit 1

Manuel Castaño - A00358994, Andrés Mayor - A00359333

Josué Rodríguez - A00359703, Jhonatan Arboleda - A00358993

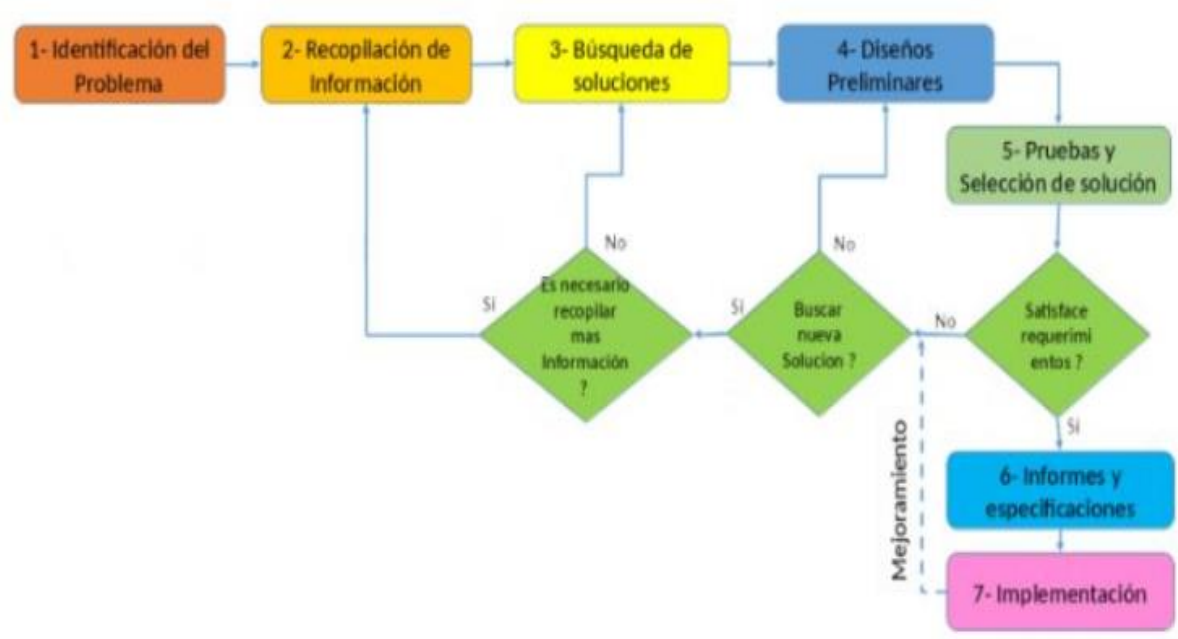
Context of the problem.

At present, the branch of computer science called Cryptography develops a very important role for software developed to meet data security. This branch is supported by the great ally of engineers, mathematics, used to define some encryption algorithms with the primes.

A company hired my team to implement more security to their systems and make their own encryption software.

Solution development.

To carry out the solution of the previous situation, we decided to follow the steps in the chart below, which is presented in the book "Introduction to Engineering" written by Paul Wright.



Step 1. Identification of the problem

Definition of the problem:

The company requires an algorithm that generates prime numbers in different ways for encryption processes.

The needs of the situation are recognized, defined the terms and conditions under which must be resolved.

- The program must have a user interface for the convenience of customers. This interface serves to receive an integer that is requested for methods that generate prime numbers.
- They must be made with three different methods which generate prime numbers up to a number n (three methods have the same function in different ways).
- Generating an array of integers running from 1 to n. This matrix should be as square as possible (i.e., if n = 100, the matrix must be 10 x 10, or if n = 20, the matrix must be 5x4 or 4x5).
- In the matrix, the generated numbers that are categorized as prime numbers must be painted green, the other red.
- You should be able to see in real time what it takes to finish the algorithm execution.

Step 2. Gather Information.

To be clear about the situation in question must specifically know some terms (mostly mathematicians), and so we can have a good basis for designing a possible solution.

Concepts:

- Prime number: A prime number is greater than 1 natural number has only two divisors, the same number and 1. The number 1 has properties of being a prime number and a composite number, therefore, it is not considered to none of these features.
  By Theorem Euclid, prime numbers are infinite, and some of them are 2, 3, 5, 7, 11, ...
- GUI: The GUI is a type of user interface that uses images, icons and menus to facilitate interaction of the client or user with a program, also providing a more friendly environment.

- Matrix: A matrix is a two-dimensional array of numbers for mathematics. For programming, an array can contain any type of data. In our case we use a composite matrix of integers.

$$\mathbf{A} = \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{pmatrix}$$

Step 3. Search for solutions.

On the Internet we can find different methods to generate square matrices or generate prime numbers, which is the primary need. In this step, we can come to evaluate our own solutions or even those already found on the Internet.

- Methods to generate prime numbers.
  1. Option 1: This method is to explore all natural numbers there from the number 2 to the number being evaluated and entered as parameter.
     To achieve this, within one cycle (either for or while) is created a counter initialized to the number 2 and which goes to the number is evaluated (but not including). Within this cycle checks the module number to evaluate the meter, if this is zero then one light variable (in this case indicates if it is prime) to value "false" and the loop end, is changed indicating if evaluated the number is prime or not.
     Resource: http://lineadecodigo.com/java/numeros-primos-en-java/
  2. Option 2. For this case runs from the number 2 to the limit number.
     A loop counter value 2. The condition of the cycle is created: counter * counter <= limit, which means that if the result of multiplying the counter by the same is greater than the number that is evaluated, then the cycle do not enter and is defined as the evaluated prime.
     Resource: http://ayudasprogramacionweb.blogspot.com/2012/11/los-numeros-primos-en-java.html (They made some important changes to the method that appears on the page).
  3. Option 3: In this method runs from number 1 to number is evaluated.
     In a cycle a variable initialized to 1 that goes to number being evaluated within the cycle being evaluated, if the module counter with the number evaluated is equal to 0. When it is 0, a variable x is created it is initialized to zero adds 1.
     Out of the two conditional cycle analyzing the value of the variable x, if the value is less than or equal to 2 then defined that the number is prime evaluated. If it is greater than two, then it is defined as not prime.
     Resource: https://www.lawebdelprogramador.com/codigo/Java/3567-Determinar-si-un-numero-es-primo.html

- Method for generating a matrix:
  - To generate a square matrix as possible depending on the limit number, we must first take the square root of the number in order to limit and assess whether this results in an integer or decimal number. If is an integer, then the matrix is created with the result of the square-root value of its rows and columns. If the result is a decimal number, then the matrix is created with the result of the square root adding the number 1 as the value of the rows and columns.

Step 4. Preliminary designs.

In this step we will analyze the options to see if they can correct or should be discarded.

1. Option 1: For this solution the only change we will do to make it a little more efficient iterative method will turn this into a recursive method.
2. Option 2: In this algorithm found an error indicating that the number 1 is also a prime number, this occurs by an indication that must be corrected on the condition that defines whether a prime number. Rest correctly identifies that number are prime and which are not. This algorithm will also make the change to recursive method iterative method.
3. Option 3: This method will be implemented with the same structure.
4. The method generates matrix is a completely correct algorithm to be implemented without any changes.

Step 5. Evaluation and Selection.

In this step we will evaluate some criteria for the solution options that have been proposed. The weight of each criterion is defined with a number which will go to the side of each option.

Criterion 1: Analysis of temporal complexity.

1: O (n log (n)), 1.5: O (n), 2: O (log (n))

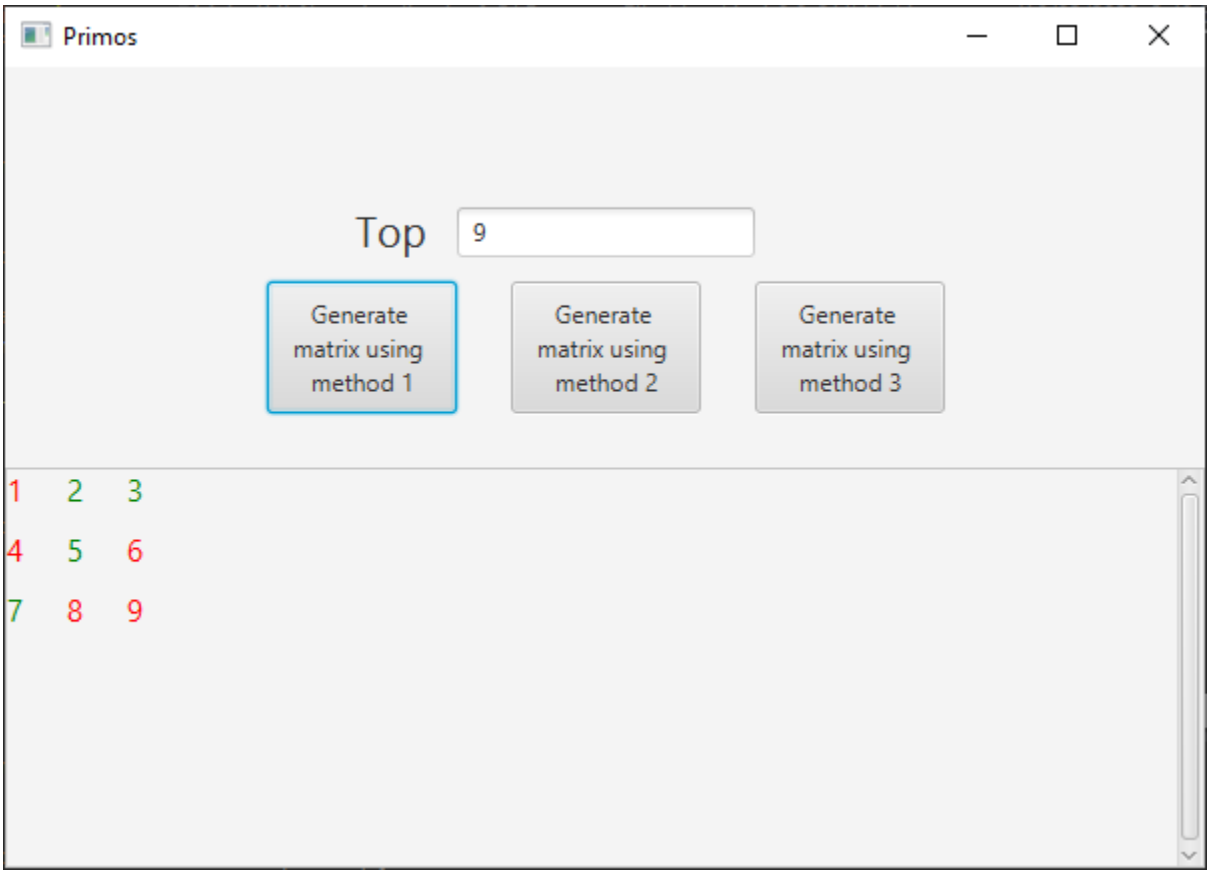
Criterion 2: For all entries of numbers the method provides the correct answer.

0: No, 1: Yes

Evaluation algorithms that generate prime numbers			
	criterion 1	criterion 2	Total
Option 1	1.5	1	2.5
Option 2	2	1	3
Option 3	1.5	1	2.5

Conclusion.

The situation raised by the company asks us that the program has 3 different methods which generate prime numbers; therefore, the three options are chosen efficient solutions provided they have made the corrections were necessary. In the case of the matrix, the selected method is quite efficient and meets all requirements posed the problem, including colored numbers depending on the property to have. The graphical interface is designed with all requested requirements and forth a screenshot of this is shown.



Step 6. Preparation of reports and specifications.

- Functional Requirements

Name	**R. # 1. Allow the generation of prime numbers.**
Summary	It Allows the generation of prime numbers.
inputs	
• Number (n).	
results	
Prime Numbers Have Been generated.	

Name	**R. # 2. Allow the maximum number of fingering or cap (n).**
Summary	It Allows the fingering of the number, for the search of the prime numbers less than or equal to the number (n).
inputs	
• Number (n).	
results	
Matrix of numbers (from 1 to n) as square as possible.	

Name	**R. # 3. Paint the prime numbers are green and Those That not red.**
Summary	It Allows painting the prime numbers of green and red are not Those That.
inputs	
• None.	
results	
The prime numbers and Those With Their Corresponding not Have Been colored painted.	

Name	**R. # 4. Allows it to show in real time the process That the algorithm performs.**
Summary	It allows it to show in real time the process of the algorithm to find the prime numbers.
inputs	
• None.	
results	
The actual process of the algorithm shown has-been in real time.	

- Analysis of time complexity

```
public boolean isPrimeNumber1(int number) {
            if(number == 1) {                                   1
                  return false;                                 1
            }
            boolean prime = true;                               1
            for (int i = 2; prime && i < number; i++) {         n-1
                  if(number % i == 0) {                         n-2
                        prime = false;                          n-2
                  }
            }
            return prime;                                       1
      }
```

Complejidad temporal -1+3n = O(n)

```
public boolean isPrimeNumber2(int number)
      {
          boolean prime = true;                             1
          if(number < 2)                                    1
          {
            prime = false;                                  1
          }
          Else                                              1
          {
              for(int x = 2; x*x <= number; x++)            ⌊log2(n)⌋
              {
                  if(number % x == 0){                      ⌊log2(n)⌋-1
                   prime = false;                           ⌊log2(n)⌋-1
                   break;                                   ⌊log2(n)⌋-1
                  }
              }
          }
          return prime;                                     1
      }
```

Complejidad temporal 2+4$\lfloor log_2(n) \rfloor$ = O(log(n))

```
public boolean isPrimeNumber3(int n){
            int a = 0;                          1
            for (int i = 1; i <= n; i++) {      n+1
                  if (n % i == 0) {             n
                        a++;                    n
                  }
            }
            if (a != 2)                         1
                  return false;                 1
            else                                1
                  return true;                  1
      }
```

Complejidad temporal 6+3n=O(n)

- Analysis of spatial complexity

isPrimeNumber1

Type	Variable	Number of values
Input	number	1
Assistant	i	1
Output	prime	1

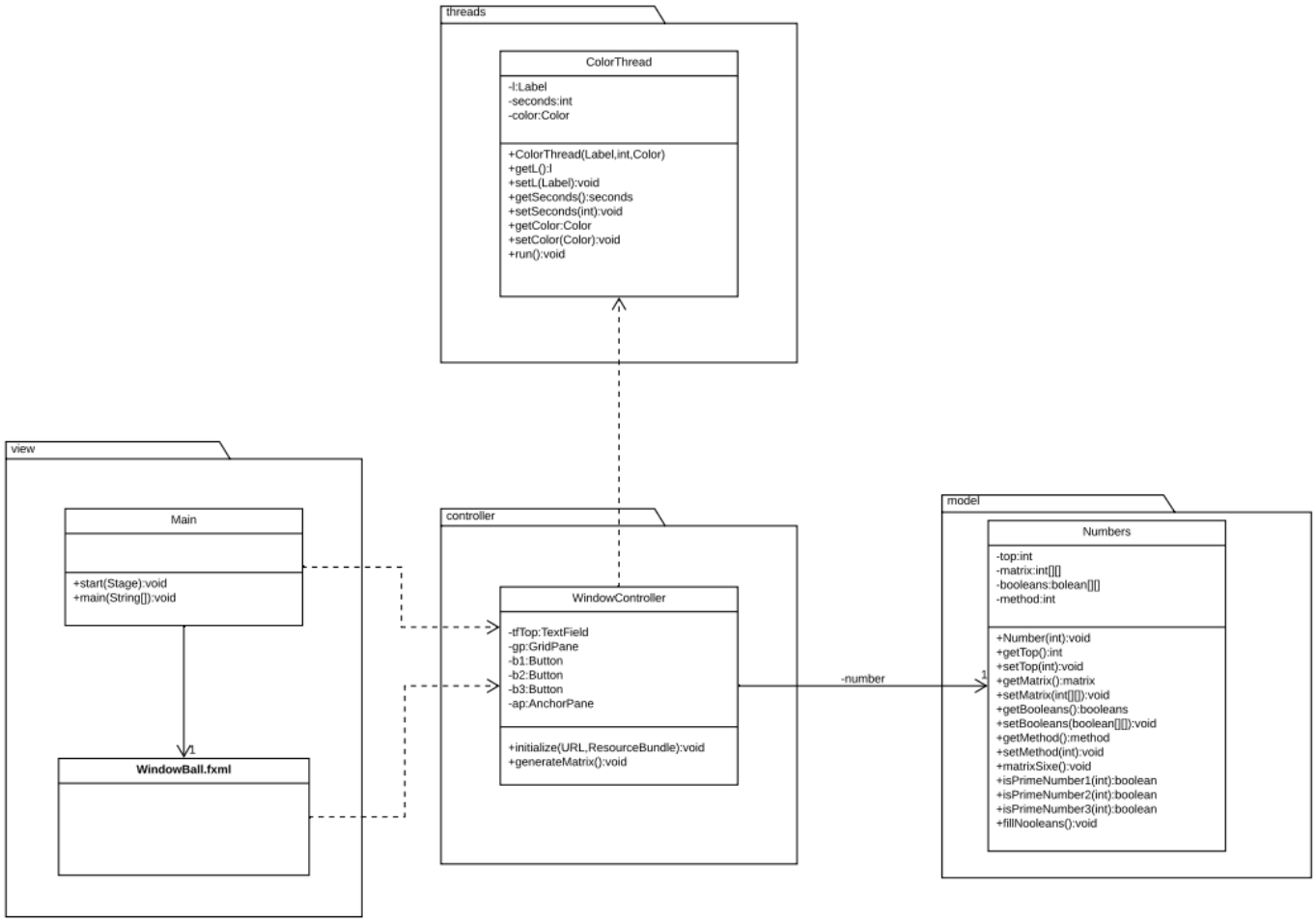
isPrimeNumber2

Type	Variable	Number of values
Input	number	1
Assistant	x	1
Output	prime	1

isPrimeNumber3

Type	Variable	Number of values
Input	n	1
Assistant	a	1
Assistant	i	1

- Class Diagram



Step 7. Implementation of the design.

In the following repository is the implementation of the solution proposed above:

https://github.com/manuelcastano/Primos