



ESCUELA SUPERIOR DE COMPUTO



PROGRAMACIÓN ORIENTADA A OBJETOS

PRÁCTICA 2

Programas a POO

Comenzar a utilizar el paradigma de la Programación Orientada a Objetos para entender su evolución sobre la programación estructurada.

Rubio Haro Rodrigo R.

CDMX. FEBRERO, 2020.

INSTITUTO POLITÉCNICO NACIONAL

Practica 2: Programas a POO

1. Introducción

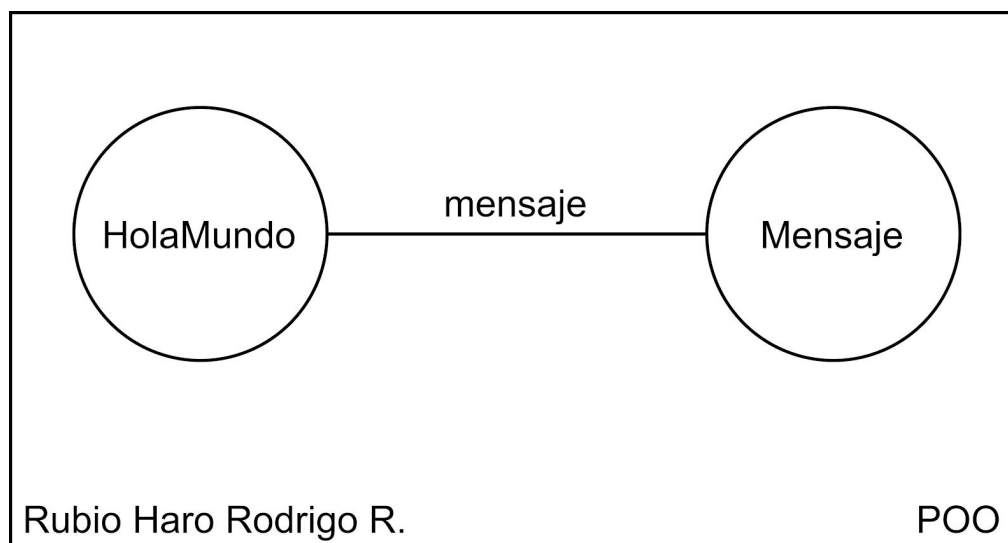
En 1979 C++ es inventado en los mismos laboratorios que dieron vida a C (Laboratorios Bell en Murray Hill, New Jersey) por el danés **Bjarne Stroustrup**. Quien originalmente llamo al nuevo lenguaje “C con clases”. Por su parte, en 1991 surgía en las oficinas de Sun Microsystems el proyecto Oak, un nuevo lenguaje de programación. Renombrado como JAVA en 1995, este proyecto se convertiría en uno de los lenguajes de programación más relevantes de la programación orientada a objetos. Con esto, se gestaba una nueva forma de programar, mucho más legible y escalable. Entender que la programación orientada a objetos nos permite dividir un gran problema en problemas más sencillos, dejando de usar tipos de datos primitivos para dar paso a los objetos. El fin de esta práctica es llevar los conceptos de la programación orientada a objetos a un código de programación estructurada.

2. Desarrollo

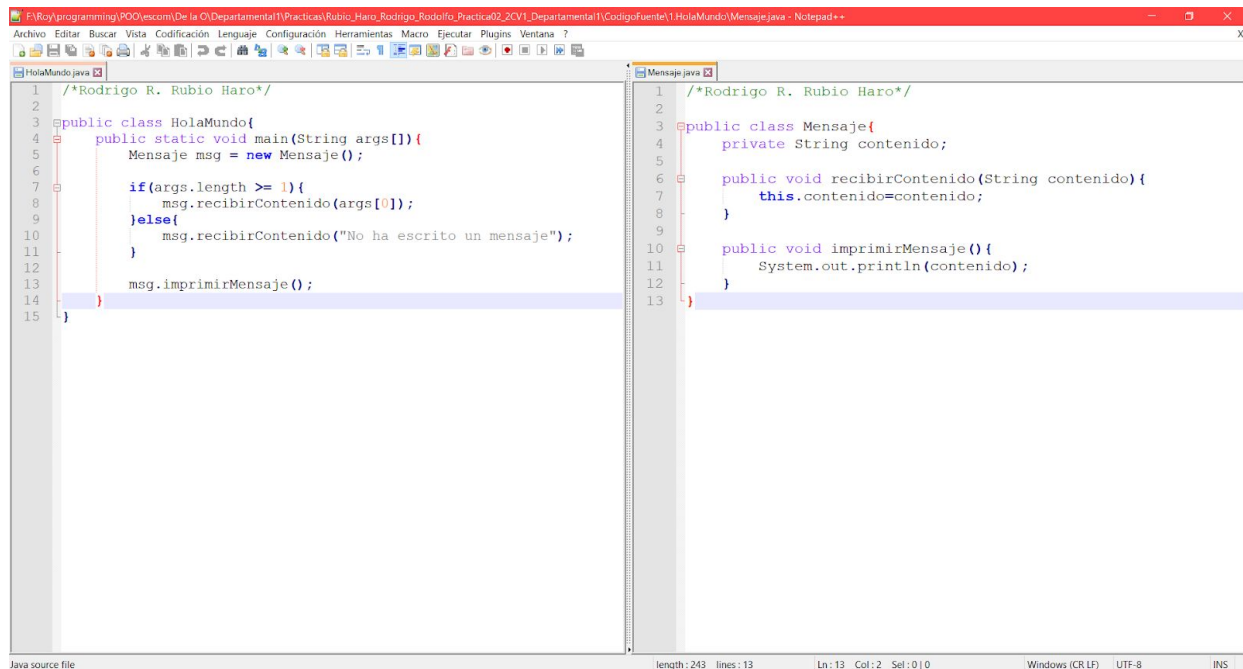
2.1 Hola Mundo

El Hola mundo es un programa muy sencillo que imprime el mensaje “Hola Mundo”. Con programación orientada a objetos podemos abstraer objetos como podría ser el Mensaje.

2.1.1 Conceptualización



2.1.2 Desarrollo



```

1  /*Rodrigo R. Rubio Haro*/
2
3  public class HolaMundo{
4      public static void main(String args[]){
5          Mensaje msg = new Mensaje();
6
7          if(args.length >= 1){
8              msg.recibirContenido(args[0]);
9          }else{
10             msg.recibirContenido("No ha escrito un mensaje");
11          }
12
13             msg.imprimirMensaje();
14         }
15     }

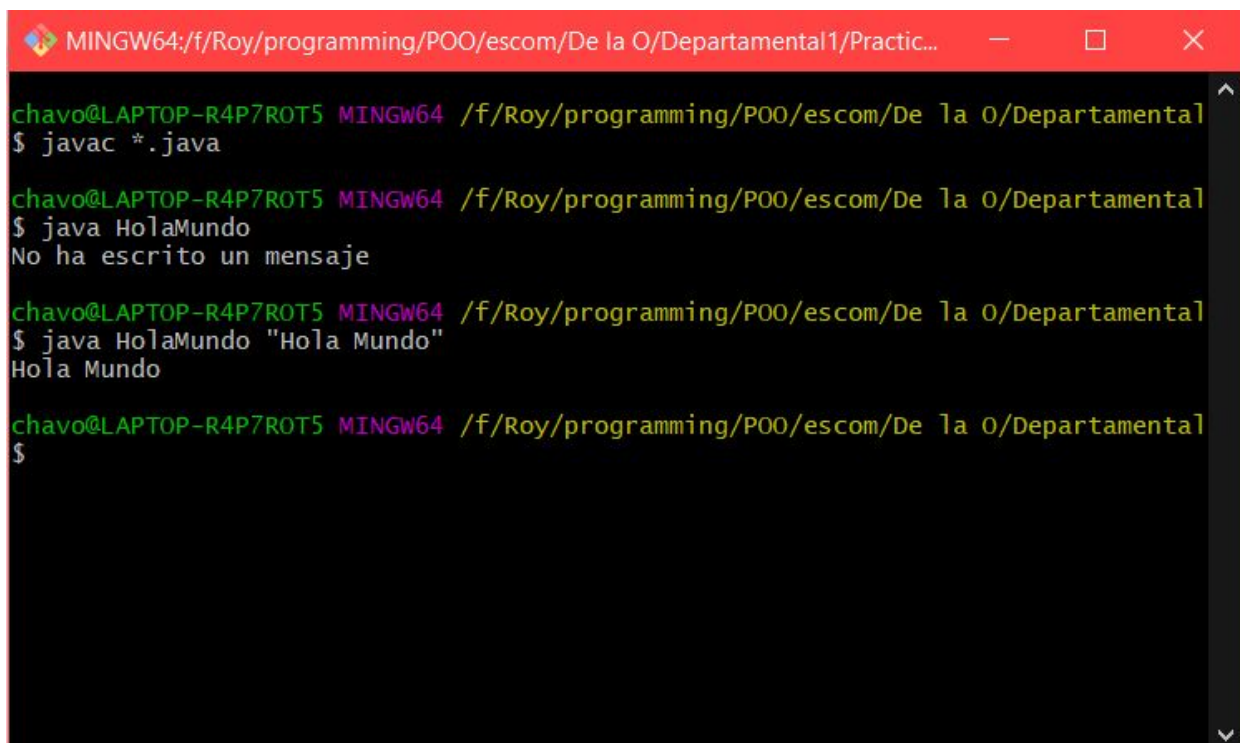
```

```

1  /*Rodrigo R. Rubio Haro*/
2
3  public class Mensaje{
4      private String contenido;
5
6      public void recibirContenido(String contenido){
7          this.contenido=contenido;
8      }
9
10     public void imprimirMensaje(){
11         System.out.println(contenido);
12     }
13 }

```

Se creó una Clase HolaMundo en donde se colocó el main. También, se creó una clase Mensaje con “contenido” como principal y único atributo; y dos métodos, uno para recibir el contenido y otro para imprimir el atributo contenido. Desde la clase HolaMundo se creó el objeto Mensaje y mediante los argumentos del main se le asignó el contenido al Mensaje. Por último la clase main manda a ejecutar el método imprimirMensaje de la clase Mensaje.



```

MINGW64:/f/Roy/programming/POO/escom/De la O/Departamental1/Practic...
chavo@LAPTOP-R4P7ROT5 MINGW64 /f/Roy/programming/POO/escom/De la O/Departamental1
$ javac *.java

chavo@LAPTOP-R4P7ROT5 MINGW64 /f/Roy/programming/POO/escom/De la O/Departamental1
$ java HolaMundo
No ha escrito un mensaje

chavo@LAPTOP-R4P7ROT5 MINGW64 /f/Roy/programming/POO/escom/De la O/Departamental1
$ java HolaMundo "Hola Mundo"
Hola Mundo

chavo@LAPTOP-R4P7ROT5 MINGW64 /f/Roy/programming/POO/escom/De la O/Departamental1
$

```

2.2 Formula General

2.2.1 Discriminante de una Ecuación de Segundo Grado con una Incógnita.

El discriminante de una ecuación de segundo grado con una incógnita, es un número que discrimina las ecuaciones que tienen soluciones reales de las que no. Así,

Si el discriminante es mayor o igual a 0, La ecuación tiene soluciones reales.

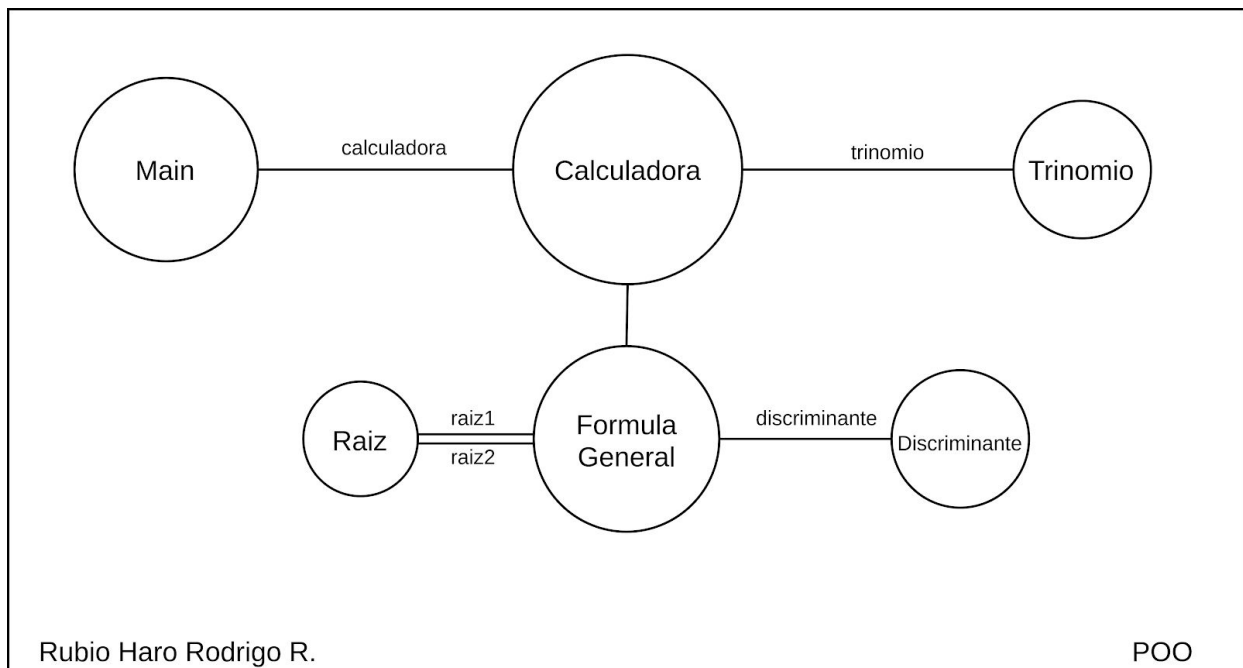
Si el discriminante es menor a 0, La ecuación NO tiene soluciones reales.

2.2.2 Formula

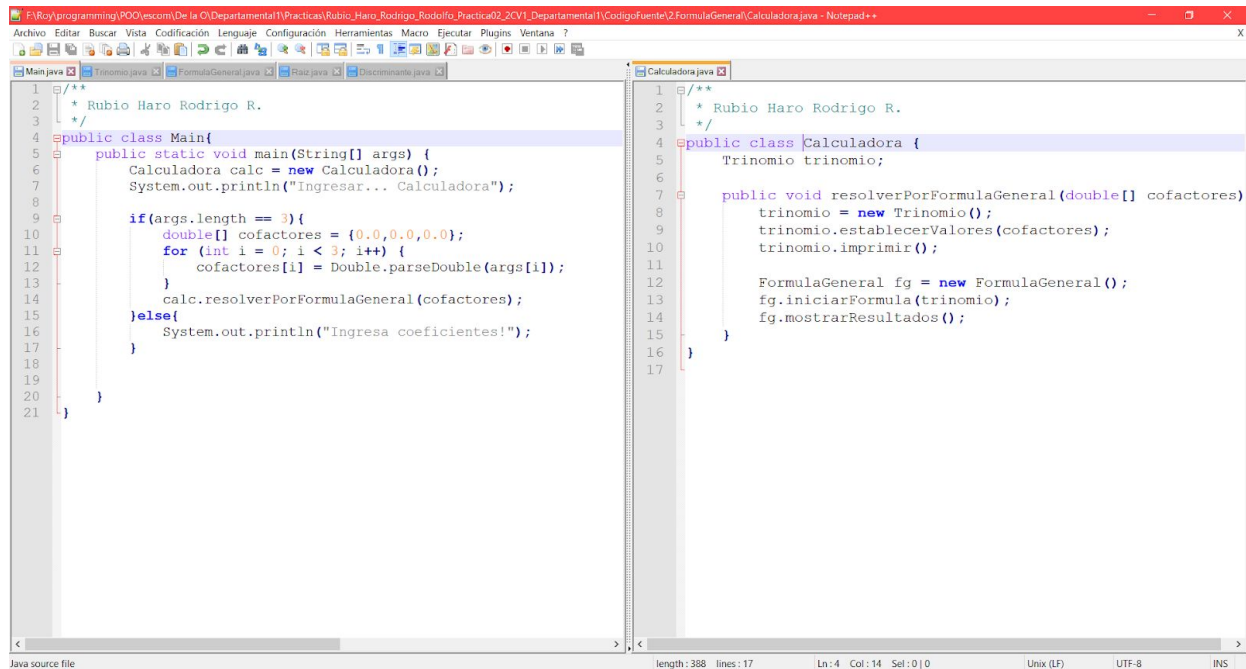
Existe una fórmula general para resolver ecuaciones de segundo grado con una incógnita

$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

2.2.3 Conceptualización



2.2.3 Desarrollo



```

1  /**
2  * Rubio Haro Rodrigo R.
3  */
4  public class Main{
5      public static void main(String[] args) {
6          Calculadora calc = new Calculadora();
7          System.out.println("Ingresar... Calculadora");
8
9          if(args.length == 3){
10             double[] cofactores = {0.0,0.0,0.0};
11             for (int i = 0; i < 3; i++) {
12                 cofactores[i] = Double.parseDouble(args[i]);
13             }
14             calc.resolverPorFormulaGeneral(cofactores);
15         } else {
16             System.out.println("Ingresa coeficientes!");
17         }
18     }
19 }
20
21

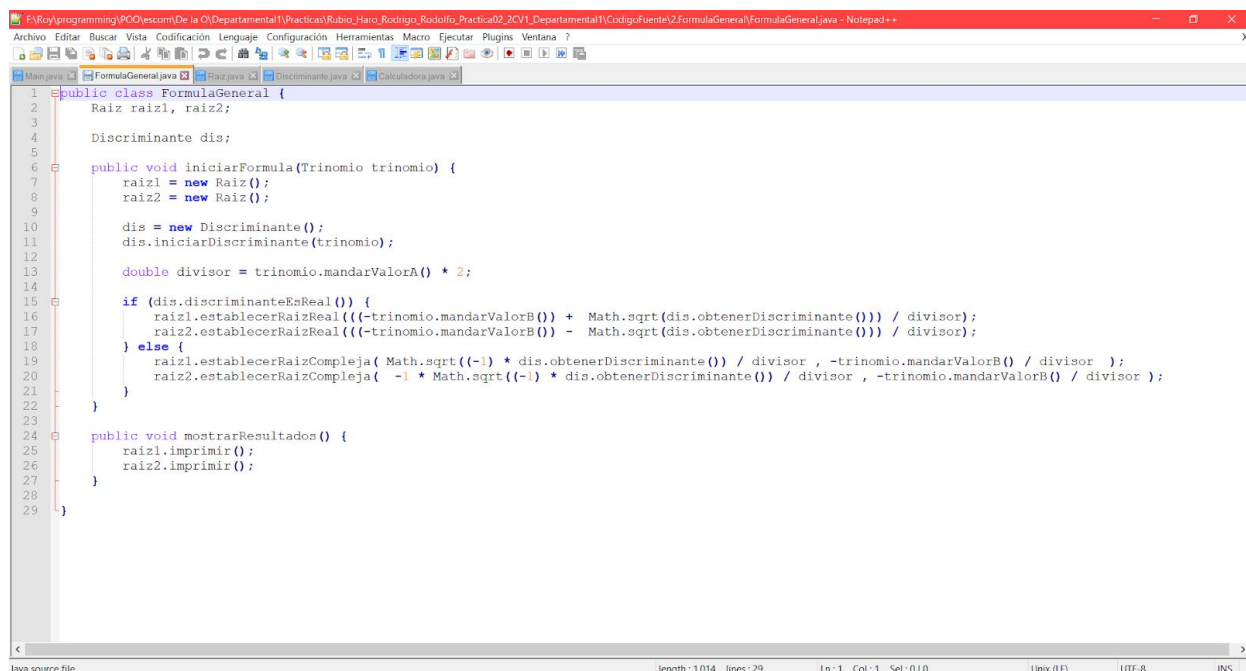
```

```

1  /**
2  * Rubio Haro Rodrigo R.
3  */
4  public class Calculadora {
5      Trinomio trinomio;
6
7      public void resolverPorFormulaGeneral(double[] cofactores)
8      {
9          trinomio = new Trinomio();
10         trinomio.establecerValores(cofactores);
11         trinomio.imprimir();
12
13         FormulaGeneral fg = new FormulaGeneral();
14         fg.iniciarFormula(trinomio);
15         fg.mostrarResultados();
16     }
17 }

```

Se desarrollaron las clases Main y Calculadora, principales del programa. Main recibe los argumentos de programa, crea un objeto Calculadora y manda a resolver por el método general los coeficientes de un trinomio. Calculadora, por su parte, crea un objeto de tipo Trinomio (que podría ser visto como una ecuación) y crea un objeto de tipo FormulaGeneral al que le manda el trinomio para resolverlo.



```

1  public class FormulaGeneral {
2      Raiz raiz1, raiz2;
3
4      Discriminante dis;
5
6      public void iniciarFormula(Trinomio trinomio) {
7          raiz1 = new Raiz();
8          raiz2 = new Raiz();
9
10         dis = new Discriminante();
11         dis.iniciarDiscriminante(trinomio);
12
13         double divisor = trinomio.mandarValorA() * 2;
14
15         if (dis.discriminanteEsReal()) {
16             raiz1.establecerRaizReal(((-trinomio.mandarValorB()) + Math.sqrt(dis.obtenerDiscriminante())) / divisor);
17             raiz2.establecerRaizReal(((-trinomio.mandarValorB()) - Math.sqrt(dis.obtenerDiscriminante())) / divisor);
18         } else {
19             raiz1.establecerRaizCompleja( Math.sqrt(-1) * dis.obtenerDiscriminante() / divisor , -trinomio.mandarValorB() / divisor );
20             raiz2.establecerRaizCompleja( -1 * Math.sqrt(-1) * dis.obtenerDiscriminante() / divisor , -trinomio.mandarValorB() / divisor );
21         }
22     }
23
24     public void mostrarResultados() {
25         raiz1.imprimir();
26         raiz2.imprimir();
27     }
28 }
29

```

La clase Formula general crea dos objetos de tipo Raiz en donde guardará los resultados, calcula la operación del divisor que es $2a$, después, crea un objeto discriminante para determinar si la ecuación tiene solución real. Por último tiene el método mostrar resultado que imprime las raíces, este es llamado desde Calculadora.

```
F:\Roy\programming\POO\escom\De la O\Departamental\Practicas\Rubio_Haro_Rodolfo_Rodolfo_Practica02_2CV1_Departamental\CodigoFuente\2FormulaGeneral\Discriminante.java - Notepad++
Archivo Editar Buscar Vista Codificación Lenguaje Configuración Herramientas Macro Ejecutar Plugins Ventana ?
Main.java Raiz.java Calculadora.java Discriminante.java
1
2 public class Discriminante {
3
4     Trinomio trinomio;
5     double valor;
6
7     public void iniciarDiscriminante(Trinomio trinomio){
8         this.trinomio = trinomio;
9     }
10
11     public boolean discriminanteEsReal() {
12         valor = (trinomio.mandarValorB() * trinomio.mandarValorB()) - (4 * trinomio.mandarValorA() * trinomio.mandarValorC());
13
14         if (valor >= 0) {
15             return true;
16         } else {
17             return false;
18         }
19     }
20
21     public double obtenerDiscriminante() {
22         return valor;
23     }
24
25 }
```

El discriminante realiza la operación definida como discriminante:

$$\text{valor} = b^2 - 4ac$$

Este nos permite saber si la ecuación tendrá como solución raíces reales, como se mencionó anteriormente. El método obtener discriminante regresa este valor sin tener que volverlo a calcular. Discriminante calcula esto obteniendo los coeficientes de un objeto Trinomio.

```
F:\Roy\programming\POO\escom\De la O\Departamental\Practicas\Rubio_Haro_Rodolfo_Rodolfo_Practica02_2CV1_Departamental\CodigoFuente\2FormulaGeneral\Raiz.java - Notepad++
Archivo Editar Buscar Vista Codificación Lenguaje Configuración Herramientas Macro Ejecutar Plugins Ventana ?
Main.java Raiz.java Calculadora.java
1
2 public class Raiz {
3     private double real;
4     private double complejo;
5
6     public void establecerRaizReal(double real) {
7         this.real = real;
8     }
9
10    public void establecerRaizCompleja(double complejo, double real) {
11        this.real = real;
12        this.complejo = complejo;
13    }
14
15    public void imprimir() {
16        System.out.print("x = " + real);
17        if (this.complejo != 0) {
18            System.out.println(" + " + complejo + "i");
19        }
20    }
21
22 }
```

Finalmente, el objeto Raiz sirve de repositorio o almacén de la solución del cálculo de la fórmula general y está compuesta por una parte compleja y una parte real.

3. Conclusión

Lo más complicado de esta practica, en lo personal, es el hecho de aplicar los fundamentos de programación orientada a objetos más allá de lo práctico y funcional del código. Sin embargo, creo que el resultado es sumamente interesante y sin duda alguna una clara mejora en mi forma de hacer programación, orientada a objetos.

Referencias

1. Oracle. (n.d.). Java SE Development Kit 8 - Downloads. Retrieved February 16, 2020, from <https://www.oracle.com/java/technologies/javase-jdk8-downloads.html>