



## INFORME DEL TALLER #3

# INTRODUCCIÓN A LA PROGRAMACIÓN ORIENTADA A OBJETOS

CRISTIAN CAMILO VALLECILLA CUELLAR	1628790-3743
CHRISTIAN CAMILO TABORDA CAMPIÑO	1632081-3743
JUAN SEBASTIAN PAZ VELÁSQUEZ	1626846-3743

## PUNTO 1: CONCEPTOS AVANZADOS

Para desarrollar este programa se diseñaron 4 clases: -Clave- (clase abstracta), -ClaveTexto-, -ClaveImagen- y -ClaveLlave-, la clase abstracta contiene los métodos para establecer y consultar el texto asociado a una contraseña, la clase -ClaveTexto- se puede instanciar ingresando un texto y tiene su respectivo método set, con la clase -ClaveImagen- se puede instanciar ingresando la matriz numérica o un texto, en caso de ingresar una matriz(imagen) realiza la conversión a texto, en caso de ingresar un texto lo convierte a una matriz(imagen) y tiene sus respectivos métodos set, por último la clase -ClaveLlave- se puede instanciar ingresando un texto o el algoritmo y un texto, en el caso del texto se separa el algoritmo del texto y en el caso del algoritmo y el texto se unen para formar un texto, esta clase también tiene sus respectivos métodos set. En el main se instancian las clases usando los diferentes constructores y métodos y se imprimen.

### Instancias de las clases:

```
int main(int argc, char* argv){  
    //Instancias de cada tipo de clave sin parámetros de entrada:  
    ClaveTexto * objClaveTextoA = new ClaveTexto();  
    ClaveImagen * objClaveImagenA = new ClaveImagen();  
    ClaveLlave * objClaveLlaveA = new ClaveLlave();  
  
    //Instancias de cada tipo de clave con texto de parámetro:  
    ClaveTexto * objClaveTextoB = new ClaveTexto("admin123");  
    ClaveImagen * objClaveImagenC = new ClaveImagen("1,243,445,6");  
    ClaveLlave * objClaveLlaveC = new ClaveLlave("rsa#a23f2321");  
  
    //Instancias de las claves imagen y llave con sus parámetros particulares:  
    int imagen[3][2] = {{1,2},{3,4},{5,6}};  
    ClaveImagen * objClaveImagenB = new ClaveImagen(imagen);  
    ClaveLlave * objClaveLlaveB = new ClaveLlave("rsa", "a23f2321");  
  
    //Consulta de cada clave:  
    cout << "\nANTES DEL SET -----< n\n";  
    cout << "Texto A: " << objClaveTextoA->getTextoClave() << endl;  
    cout << "Texto B: " << objClaveTextoB->getTextoClave() << endl;  
    cout << "Imagen A: " << objClaveImagenA->getTextoClave() << endl;  
    cout << "Imagen B: " << objClaveImagenB->getTextoClave() << endl;  
    cout << "Imagen C: " << objClaveImagenC->getTextoClave() << endl;  
    cout << "Llave A: " << objClaveLlaveA->getTextoClave() << endl;  
    cout << "Llave B: " << objClaveLlaveB->getTextoClave() << endl;  
    cout << "Llave C: " << objClaveLlaveC->getTextoClave() << endl;  
}
```

```

//Seteo de todas las claves:
objClaveTextoA->setTextoClave("camilo12");
objClaveTextoB->setTextoClave("christian12");
int matriz1[3][2] = {{6,5},{4,3},{2,1}};
int matriz2[3][2] = {{2,4},{6,8},{10,12}};
objClaveImagenA->setTextoClave(matriz1);
objClaveImagenB->setTextoClave("6,5#4,3#2,1");
objClaveImagenC->setTextoClave(matriz2);
objClaveLlaveA->setTextoClave("chr", "123456");
objClaveLlaveB->setTextoClave("chr#123456");
objClaveLlaveC->setTextoClave("cml", "789");

//Consulta de cada clave nuevamente:
cout << endl;
cout << "DESPUÉS DEL SET -----\\n\\n";
cout << "Texto A: " << objClaveTextoA->getTextoClave() << endl;
cout << "Texto B: " << objClaveTextoB->getTextoClave() << endl;
cout << "Imagen A: " << objClaveImagenA->getTextoClave() << endl;
cout << "Imagen B: " << objClaveImagenB->getTextoClave() << endl;
cout << "Imagen C: " << objClaveImagenC->getTextoClave() << endl;
cout << "Llave A: " << objClaveLlaveA->getTextoClave() << endl;
cout << "Llave B: " << objClaveLlaveB->getTextoClave() << endl;
cout << "Llave C: " << objClaveLlaveC->getTextoClave() << endl;
cout << endl;

return 0;
}

```

**SALIDA:**

cristian@cristian-VirtualBox: ~/Descargas/TALLER3/Punto1

ANTES DEL SET -----

Texto A:  
 Texto B: admin123  
 Imagen A:  
 Imagen B: 1,2#3,4#5,6  
 Imagen C: 1,2#3,4#5,6  
 Llave A:  
 Llave B: rsa#a23f2321  
 Llave C: rsa#a23f2321

DESPUÉS DEL SET -----

Texto A: camilo12  
 Texto B: christian12  
 Imagen A: 6,5#4,3#2,1  
 Imagen B: 6,5#4,3#2,1  
 Imagen C: 2,4#6,8#10,12  
 Llave A: chr#123456  
 Llave B: chr#123456  
 Llave C: cml#789

cristian@cristian-VirtualBox:~/Descargas/TALLER3/Punto1\$

## PUNTO 2: CONTROL DE EXCEPCIONES

### CALCULADORA:

Para el desarrollo de este punto se creó una clase –Calculadora-, la cual tiene los métodos sumar, restar, multiplicar y dividir, que realizan las operaciones como su nombre lo indica, en el main se crearon varios métodos: para imprimir la parte del menú y para realizar el control de excepciones al ingresar el primer y segundo número. Se obtienen los números, realizando el control, se instancia la clase y por medio de un switch se escoge la operación a realizar utilizando los métodos mencionados anteriormente, cabe aclarar que en el caso de dividir se valida también que el segundo número sea diferente de 0.

Inicio del programa:

```
cristian@cristian-VirtualBox:~/Descargas/TALLER3/Punto2/Calculadora$ g++ -o exe *.cpp
cristian@cristian-VirtualBox:~/Descargas/TALLER3/Punto2/Calculadora$ ./exe

*****
*          CALCULADORA          *
*****

Ingrese el primer número:

```

Primera validación:

Ingresando una letra en vez de un número.

```
cristian@cristian-VirtualBox:~/Descargas/TALLER3/Punto2/Calculadora$ g++ -o exe *.cpp
cristian@cristian-VirtualBox:~/Descargas/TALLER3/Punto2/Calculadora$ ./exe

*****
*          CALCULADORA          *
*****

Ingrese el primer número:
d
Debe ingresar un valor numérico.
Ingrese nuevamente el primer número:

```

Ingresando el primer número:

```
cristian@cristian-VirtualBox: ~/Descargas/TALLER3/Punto2/Calculadora
cristian@cristian-VirtualBox:~/Descargas/TALLER3$ cd Punto2
cristian@cristian-VirtualBox:~/Descargas/TALLER3/Punto2$ ls
Calculadora  Inventario
cristian@cristian-VirtualBox:~/Descargas/TALLER3/Punto2$ cd calculadora
bash: cd: calculadora: No existe el archivo o el directorio
cristian@cristian-VirtualBox:~/Descargas/TALLER3/Punto2$ cd Calculadora
cristian@cristian-VirtualBox:~/Descargas/TALLER3/Punto2/Calculadora$ g++ -o exe *.cpp
cristian@cristian-VirtualBox:~/Descargas/TALLER3/Punto2/Calculadora$ ./exe

*****
*          CALCULADORA          *
*****

Ingrese el primer número:
d

Debe ingresar un valor numérico.

Ingrese nuevamente el primer número:
2

Ingrese el segundo número:

```

Segunda validación:

Ingresando una letra en vez de un número.

```
cristian@cristian-VirtualBox: ~/Descargas/TALLER3/Punto2/Calculadora
cristian@cristian-VirtualBox:~/Descargas/TALLER3/Punto2$ cd Calculadora
cristian@cristian-VirtualBox:~/Descargas/TALLER3/Punto2/Calculadora$ g++ -o exe *.cpp
cristian@cristian-VirtualBox:~/Descargas/TALLER3/Punto2/Calculadora$ ./exe

*****
*          CALCULADORA          *
*****

Ingrese el primer número:
d

Debe ingresar un valor numérico.

Ingrese nuevamente el primer número:
2

Ingrese el segundo número:
e

Debe ingresar un valor numérico.

Ingrese nuevamente el segundo número:

```

Ingresando el segundo número:

```
cristian@cristian-VirtualBox: ~/Descargas/TALLER3/Punto2/Calculadora
Debe ingresar un valor numérico.
Ingrese nuevamente el primer número:
2
Ingrese el segundo número:
4
Debe ingresar un valor numérico.
Ingrese nuevamente el segundo número:
2

*****
* Ingrese la operación deseada: *
*-----*
* 1. Suma.                      *
* 2. Resta.                     *
* 3. Multiplicación.            *
* 4. División.                  *
*-----*
```

Tercera validación:

Suma:

```
cristian@cristian-VirtualBox: ~/Descargas/TALLER3/Punto2/Calculadora
* Ingrese la operación deseada: *
*-----*
* 1. Suma.                      *
* 2. Resta.                     *
* 3. Multiplicación.            *
* 4. División.                  *
*-----*
1
*****
*          RESULTADO          *
*-----*
Su resultado es: 4

*****
* ¿Desea continuar?          *
*-----*
* 1. Si.                     *
* 2. No.                      *
*-----*
```

**Resta:**

```
cristian@cristian-VirtualBox: ~/Descargas/TALLER3/Punto2/Calculadora
* Ingrese la operación deseada: *
* ..... *
* 1. Suma. *
* 2. Resta. *
* 3. Multiplicación. *
* 4. División. *
* ..... *
2
* ..... *
* RESULTADO *
* ..... *
Su resultado es: 0
* ..... *
* ¿Desea continuar? *
* ..... *
* 1. Si. *
* 2. No. *
* ..... *
```

**Multiplicación:**

```
cristian@cristian-VirtualBox: ~/Descargas/TALLER3/Punto2/Calculadora
* Ingrese la operación deseada: *
* ..... *
* 1. Suma. *
* 2. Resta. *
* 3. Multiplicación. *
* 4. División. *
* ..... *
3
* ..... *
* RESULTADO *
* ..... *
Su resultado es: 4
* ..... *
* ¿Desea continuar? *
* ..... *
* 1. Si. *
* 2. No. *
* ..... *
```



## División:

```
cristian@cristian-VirtualBox: ~/Descargas/TALLER3/Punto2/Calculadora
Ingrese la operación deseada: *
.....*
1. Suma. *
2. Resta. *
3. Multiplicación. *
4. División. *
.....*
4
.....*
RESULTADO *
.....*
Su resultado es: 1
.....*
¿Desea continuar? *
.....*
1. Si. *
2. No. *
.....*
```

## Caso de 0 en el segundo número:

```
cristian@cristian-VirtualBox: ~/Descargas/TALLER3/Punto2/Calculadora
Ingrese el segundo número:
0
Ingrese la operación deseada: *
.....*
1. Suma. *
2. Resta. *
3. Multiplicación. *
4. División. *
.....*
No se puede dividir entre cero.
.....*
¿Desea continuar? *
.....*
1. Si. *
2. No. *
.....*
```

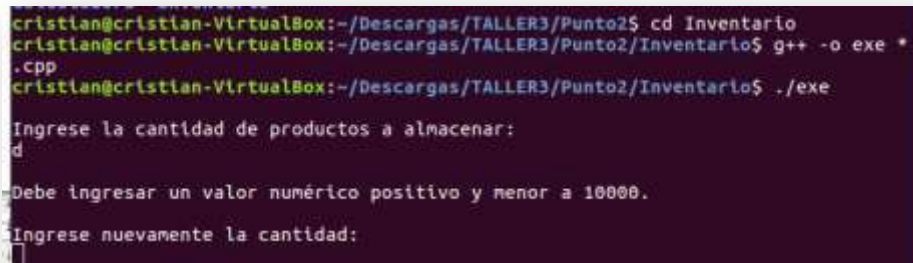


## INVENTARIO:

Para el desarrollo de este punto se crearon dos clases –Producto- e –Inventario-, la primera para instanciar recibe dos parámetros (nombre y código) y tiene sus métodos get respectivos, la segunda se instancia con la cantidad; está compuesta de un vector de tipo Producto para coleccionar los objetos, posee sus respectivos get y también tiene método para ingresar y retornar un producto dada su posición. En el main se crearon los métodos para realizar el control de excepciones. Se obtiene la cantidad realizando el control y se instancia la clase –Inventario-, con la ayuda de una estructura iterativa (for) cuya condición de parada es la cantidad ingresada, se obtiene el código y nombre del producto realizando el control, se instancia la clase –Producto- y se ingresa al inventario.

### Primera validación:

Ingresando una letra en vez de un número



```
cristian@cristian-VirtualBox:~/Descargas/TALLER3/Punto2$ cd Inventario
cristian@cristian-VirtualBox:~/Descargas/TALLER3/Punto2/Inventario$ g++ -o exe *.cpp
cristian@cristian-VirtualBox:~/Descargas/TALLER3/Punto2/Inventario$ ./exe

Ingrese la cantidad de productos a almacenar:
d

Debe ingresar un valor numérico positivo y menor a 10000.
Ingrese nuevamente la cantidad:
```

Ingresando un dato valido, el nombre del primer elemento del inventario y un dato invalido en el campo de código:

```
cristian@cristian-VirtualBox:~/Descargas/TALLER3/Punto2/Inventario$ ./exe
Ingrese la cantidad de productos a almacenar:
2
Ingrese el nombre del producto número 1:
papa
Ingrese el código del producto número 1:
d
Debe ingresar un valor numérico.
Ingrese nuevamente el código:
```

Ingresando un dato valido en el campo de código, el nombre del segundo elemento del inventario y un dato invalido en el campo de código:

```
Ingrese el código del producto número 1:
d
Debe ingresar un valor numérico.
Ingrese nuevamente el código:
1245
Ingrese el nombre del producto número 2:
arracacha
Ingrese el código del producto número 2:
d
Debe ingresar un valor numérico.
Ingrese nuevamente el código:
```

Ingresando un dato no valido en el campo de código y otro valido:

```
cristian@cristian-VirtualBox:~/Descargas/TALLER3/Punto2/Inventario$
Ingrese el nombre del producto número 1:
papa
Ingrese el código del producto número 1:
d
Debe ingresar un valor numérico.
Ingrese nuevamente el código:
1245
Ingrese el nombre del producto número 2:
arracacha
Ingrese el código del producto número 2:
d
Debe ingresar un valor numérico.
Ingrese nuevamente el código:
28282
cristian@cristian-VirtualBox:~/Descargas/TALLER3/Punto2/Inventario$
```

Ingresando un número mayor a 100000 y uno negativo en el campo cantidad:

```
*C
cristian@cristian-VirtualBox:~/Descargas/Taller3/Punto3/Calculadora$ cd ..
cristian@cristian-VirtualBox:~/Descargas/Taller3/Punto3$ cd inventario
cristian@cristian-VirtualBox:~/Descargas/Taller3/Punto3/Inventario$ ./exe

Ingrese la cantidad de productos a almacenar:
20000000

Debe ingresar un valor numérico positivo y menor a 10000.
Ingrese nuevamente la cantidad:
-1

Debe ingresar un valor numérico positivo y menor a 10000.
Ingrese nuevamente la cantidad:
```

### PUNTO 3: INTERFAZ GRAFICA: TRIQUI

Para este programa se desarrollaron tres clases: -Juego-, -Jugador- y -Tablero-, se implementó una relación de composición (Jugador y tablero componen a juego) y entre las dos últimas se implementó la relación de uso en uno de los métodos de -Jugador-, que modifica uno de los elementos de la matriz que hace parte de los atributos de -Tablero- por medio de una posición (X,Y) que aporta cada botón de la clase -Juego-. También se implementaron métodos para validar si una entrada estaba ocupada o no, y para decidir cuándo terminar la ejecución; todo lo anterior a través de la estructura condicional -if-.

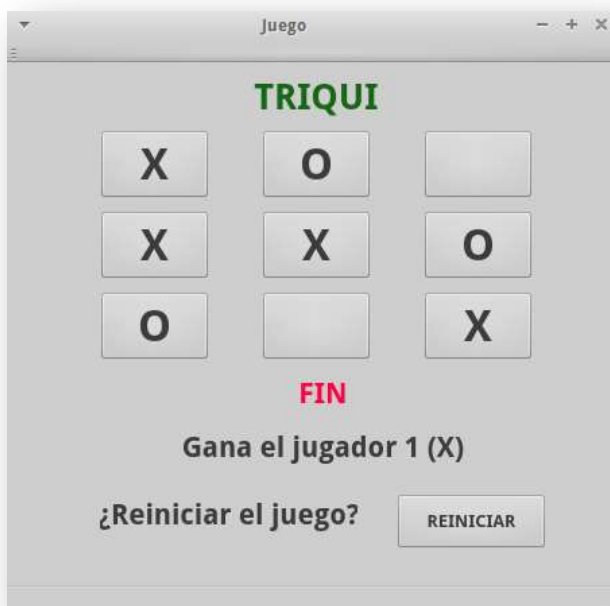
Inicio del juego:



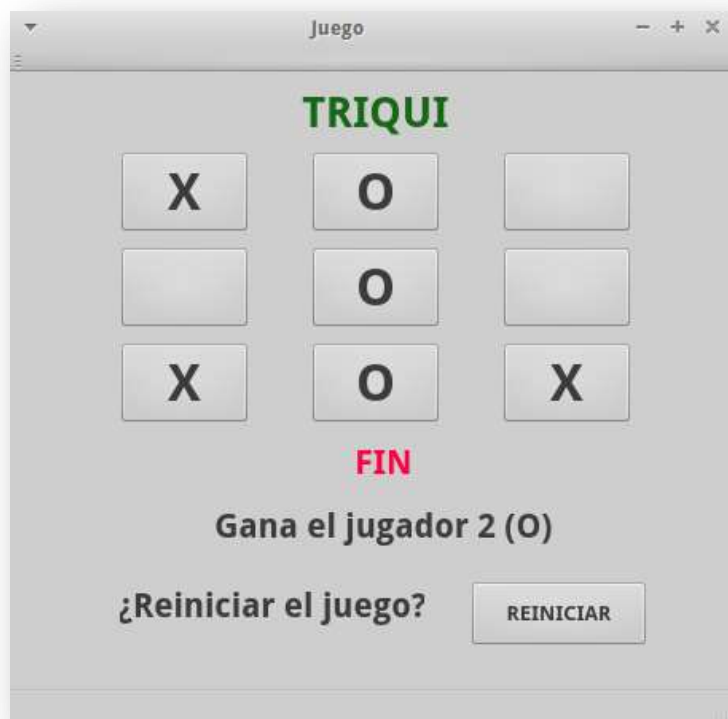
Click en una posición ocupada:



Gana el jugador 1:



Gana el jugador 2:



Empate:

