

**Universidad Nacional de Misiones**

**Facultad de Ciencias  
Exactas, Químicas y  
Naturales**

**POOI – Programación  
Orientada a Objetos I**

Trabajo Práctico Numero 2.

Titular: Lic. Claudio O. Biale

Alumno: Ulises C. Ramirez.

Año: 2017

## Introducción

En este documento podrá encontrar un fácil acceso a la resolución, o parte de la resolución de las consignas para el trabajo práctico número 2 de la cátedra, perteneciente al año 2017 reunidas en un único documento.



En este repositorio se podrá encontrar todo el código relacionado con los trabajos prácticos de la cátedra, cada uno de ellos se encontrará actualizado y corregido.

1. Las modificaciones necesarias a la clase `Circulo` para que esta sea capaz de decir cuantas instancias se han creado son las siguientes:

```
11 public class Circulo {
12     private static int cantInstanciasCirtulo;
13     private double radio;
14
15     public Circulo()
16     {
17         this.radio = 0;
18         cantInstanciasCirtulo++;
19     }
20
21     public Circulo(double Radio)
22     {
23         this.radio = Radio;
24         cantInstanciasCirtulo++;
25     }
26
27     public static int getCantInstanciasCirculo()
28     {
29         return cantInstanciasCirtulo;
30     }
31 }
```

Código 1: Primer Consigna

En la línea 12 se puede apreciar la declaración de una variable de clase llamada `cantInstanciasCirculo` la cual aumenta en uno su valor cada vez que se ejecuta un constructor, como se puede corroborar en las líneas 18 y 24. Luego podemos obtener el valor de la variable a través del método de clase llamado `getCantInstanciasCirculo()` el cual se encuentra definido a partir de la línea 27.

## 2. Diagrama de clase para la consigna:

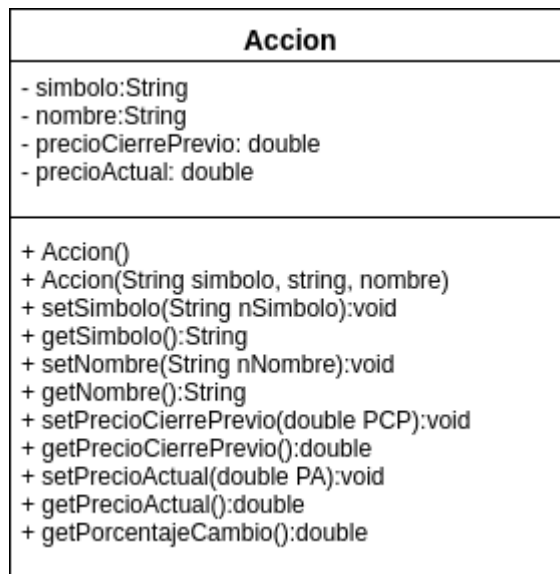


Diagrama 1: UML - Clase Accion

A continuación se presenta una captura de pantalla de la prueba solicitada en el enunciado para su fácil acceso.

```
13 public class TP2 {
14
15     /**
16      * @param args the command line arguments
17      */
18     public static void main(String[] args) {
19         double porcentajeCambio;
20
21         Accion a = new Accion("ORCL", "Oracle Corporation");
22
23         a.setPrecioActual(34.35);
24         a.setPrecioCierrePrevio(34.5);
25
26         porcentajeCambio = a.getPorcentajeCambio();
27
28         if (porcentajeCambio < 0) {
29             System.out.print("El valor de las acciones cayo, porcentaje de "
30                 + "cambio es de: "+porcentajeCambio+"%\n");
31         } else if (porcentajeCambio > 0) {
32             System.out.print("El valor de las acciones subio, porcentaje de "
33                 + "cambio es de: "+porcentajeCambio+"%\n");
34         } else {
35             System.out.print("El valor de las acciones se mantuvo%\n");
36         }
37     }
38 }
39
40
```

Debugger Console TP1 (run) TP1 (run) #2 TP2 (run)

run:  
El valor de las acciones cayo, porcentaje de cambio es de: -0.43%  
BUILD SUCCESSFUL (total time: 0 seconds)

Código 2: Implementación de los métodos de la clase Accion

En el repositorio mencionado en la introducción se podrá encontrar el código necesario para solicitarle al usuario los valores de Precio de Cierre Previo y Precio Actual, junto con el código de la implementación de la clase Accion

### 3. Diagrama de clase para la consigna:

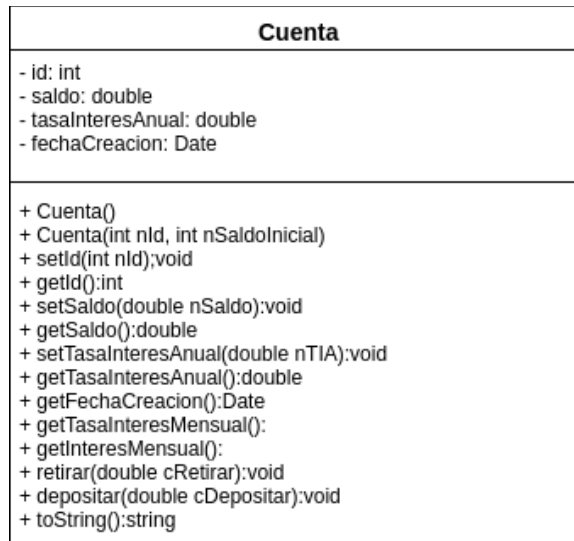
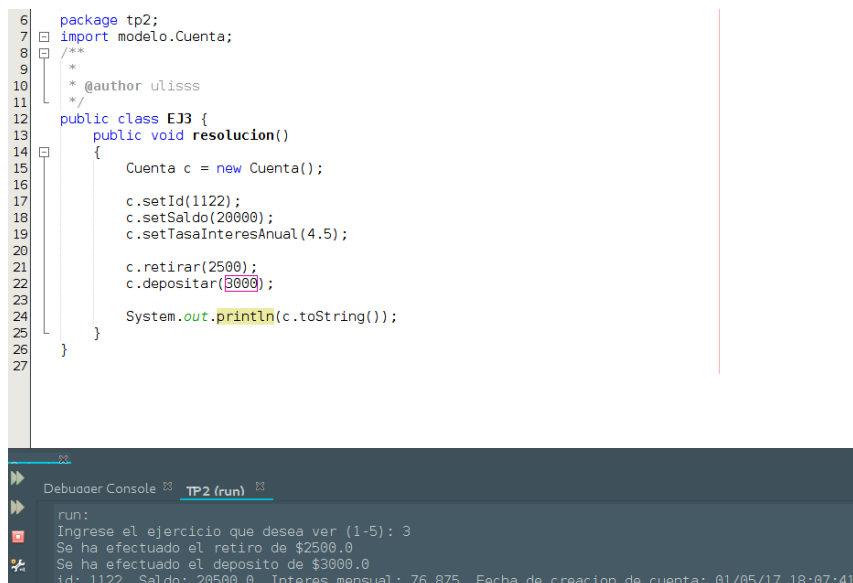


Diagrama 2: UML - Clase Cuenta

A continuación se presenta una captura de pantalla de la prueba solicitada en el enunciado para su fácil acceso.

```
6 package tp2;
7 import modelo.Cuenta;
8 /**
9  * @author ulisss
10  */
11
12 public class EJ3 {
13     public void resolucion()
14     {
15         Cuenta c = new Cuenta();
16
17         c.setId(1122);
18         c.setSaldo(20000);
19         c.setTasaInteresAnual(4.5);
20
21         c.retirar(2500);
22         c.depositar(3000);
23
24         System.out.println(c.toString());
25     }
26 }
27
```



Código 3: Implementación de los métodos de la clase Cuenta

En el repositorio mencionado en la introducción se podrá encontrar el código presentado en la captura de pantalla, además podrá encontrar el código perteneciente a la implementación de la clase Cuenta

#### 4. Diagrama de clase para la consigna

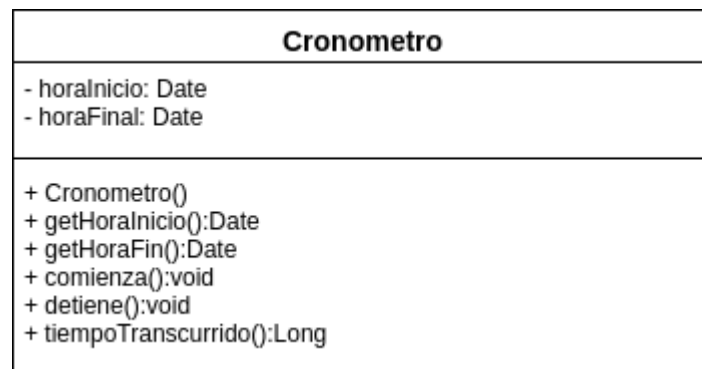


Diagrama 3: UML - Clase Cronometro

A continuación se presenta una captura de pantalla en donde se puede apreciar de manera rápida la implementación de lo solicitado en el enunciado junto con el resultado obtenido

Nuevamente, en el repositorio mencionado en la introducción se tendrá el código ligeramente modificado, en éste, el programa solicitará al usuario el ingreso de la cantidad de números que desea imprimir por pantalla, al terminar la impresión dirá cuantos mili-segundos le llevó realizar el trabajo, como así también encontrará el código perteneciente a la implementación de la clase Cronometro

```
13 public class EJ4 {
14     public void resolucion()
15     {
16         int i; long n;
17         n = 100000;
18         Cronometro c = new Cronometro();
19         for (i=0 ; i <= n ; i++) {
20             System.out.println("\t"+i);
21         }
22         c.detiene();
23         System.out.println("El tiempo transcurrido durante la impresion de los"
24             + " numeros hasta el "+ n + " fue: " + c.tiempoTranscurrido() +
25             " Milisegundos");
26     }
27 }
28
```



Código 4: Implementación de los métodos de la clase Cronometro

5. Diagrama de clase para la consigna:

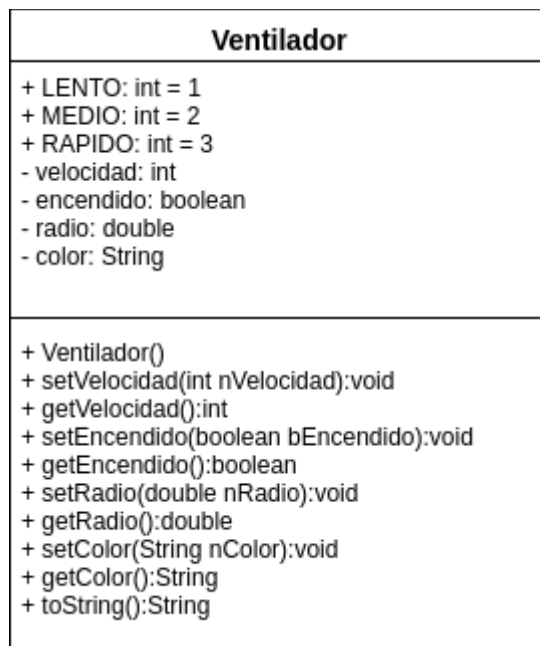
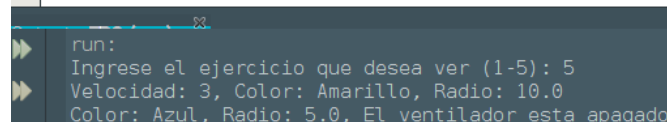


Diagrama 4: UML - Clase Ventilador

A continuación se presenta una captura de pantalla en donde se puede apreciar de manera rápida la implementación de lo solicitado en el enunciado junto con el resultado obtenido

```
12 public class EJ5 {
13     public void resolucion()
14     {
15         Ventilador v1, v2;
16         v1 = new Ventilador();
17         v2 = new Ventilador();
18         // Ventilador 1
19         v1.setVelocidad(v1.RAPIDO);
20         v1.setRadio(10);
21         v1.setColor("Amarillo");
22         v1.setEncendido(true);
23         // Ventilador 2
24         v2.setVelocidad(v2.MEDIO);
25         v2.setRadio(5);
26         v2.setColor("Azul");
27         v2.setEncendido(false);
28
29         System.out.println(v1.toString());
30         System.out.println(v2.toString());
31     }
32 }
33
```



```
run:
Ingrese el ejercicio que desea ver (1-5): 5
Velocidad: 3, Color: Amarillo, Radio: 10.0
Color: Azul, Radio: 5.0, El ventilador esta apagado
```

Código 5: Implementación de los métodos de la clase Ventilador

En el repositorio mencionado podrá encontrar el código de la implementación de los métodos junto con toda la implementación de la clase Ventilador