



LABORATORIO N° 2 CLASE CIUDAD: PARTE II

Dada la definición de la clase *Ciudad*, parcialmente implementada en Java (en **negrita** se listan los métodos sin implementar):

Ciudad
<< Atributos de instancia >> CP: entero poblacion: entero superficie: real
<< Constructores >> Ciudad (c: entero) Ciudad (c: entero, p: entero, s: entero)
<< Comandos >> establecerPoblacion (pob: entero) establecerSuperficie (sup: real) aumentarPoblacion (cre: entero) copy(otra: Ciudad)
<< Consultas >> obtenerCP(): entero obtenerPoblacion(): entero obtenerSuperficie(): real obtenerDensidad(): real toString(): String equals(otra: Ciudad): boolean clone(): Ciudad

Requiere que la variable **otra** se encuentre ligada.

1. Completar la implementación de la clase *Ciudad*, teniendo en cuenta la siguiente especificación:

- **copy(otra: Ciudad):** Modifica el estado interno del objeto que recibe el mensaje con los valores de los atributos de la ciudad recibida por parámetro. Requiere que la variable **otra** se encuentre ligada.
- **equals(otra: Ciudad): boolean:** Retorna verdadero *si y solo si* el estado interno del objeto que recibe el mensaje mantiene los mismos valores que el estado interno del objeto recibido por parámetro. Requiere que la variable **otra** se encuentre ligada.
- **clone(): Ciudad:** Retorna una nueva ciudad equivalente a la que recibió el mensaje, esto es, un objeto con el mismo estado interno que el objeto que recibe el mensaje.



Introducción a la Programación Orientada a Objetos

DCIC - UNS
2019



2. Implementar una clase Tester para probar la implementación. Recuerde realizar sus pruebas con casos significativos. A continuación se muestra una imagen referente a cómo podría ser la salida de un programa de testeo para este laboratorio.

```
Blue: Terminal Window - Laboratorio3 - Implementado
Options

/--Inicio testeo--//

Estado interno ciu1:Codigo Postal: 20 - Poblacion: 5400 - Superficie: 65485.0

Estado interno ciu2:Codigo Postal: 20 - Poblacion: 5400 - Superficie: 65485.0

Estado interno ciu3:Codigo Postal: 20546 - Poblacion: 5400 - Superficie: 65485.0

//-----Prueba equals-----//

{ciu1.equals(ciu2)? : true
{ciu1.equals(ciu3)? : false

//-----Prueba clone-----//

Clonando ciu1 en clon1 ...

Clonando ciu3 en clon3 ...

{ciu1.equals(clon1)? : true
{ciu1==clon1? : false

{ciu3.equals(clon3)? : true
{ciu3==clon3? : false

//-----Prueba copy-----//

Estado interno ciu1:Codigo Postal: 20 - Poblacion: 5400 - Superficie: 65485.0

Estado interno ciu3:Codigo Postal: 20546 - Poblacion: 5400 - Superficie: 65485.0

copiando en ciu1 el estado interno de ciu3 ...

Estado interno ciu1 después del copy:Codigo Postal: 20546 - Poblacion: 5400 - Superficie: 65485.0
```

Tips:

- Prestar atención a la palabra “**requiere**” en la especificación. La misma nos indica que la clase asume que el control está hecho por sus clientes.
- Si bien en el proceso de desarrollo de software existe una etapa dedicada exclusivamente al testeo, los programadores tienen que habituarse a probar la correctitud de sus programas. Al momento de implementar la clase tester hay que tener en cuenta qué aspectos se quieren probar de su código utilizando valores representativos.