UNIVERSIDAD DE BURGOS
Departamento de Ingeniería Civil
Área de Lenguajes y Sistemas Informáticos

METODOLOGÍA DE LA PROGRAMACIÓN **10-NOVIEMBRE-2020 – 1ª CONVOCATORIA**

Apellidos: _____ ***Nombre:*** _____

Estimación del alumno/a de su calificación (sobre 2 puntos): ☐

Total del ejercicio 2 ptos. Nota mínima de corte 0.9 Ptos

Nota: no se corrigen respuestas con tachones o realizadas a lápiz. **NO se solicita documentar el código fuente con comentarios, ni con comentarios *javadoc*.**

1. **Dada las siguientes declaraciones de clases en Java:**

```java
public class A {

	B b;
	private C c;

	public A(int i){
		b = new B();
	}

	public static C generar(){
		return new C();
	}

	void establecer(C c) {
		this.c = c;
	}
}
```

```java
public class B {
	static int valor;
	private C c;

	public B() {
		valor++;
		c = new C();
	}

	public int obtener() {
		return c.obtener() + valor;
	}
}
```

```java
public class C {
	private int k;

	public int obtener() {
		return k;
	}
}
```

```java
package x;
import x.y.A;
import x.y.z.B;
import z.y.x.C;
public class P {
	public static void main(String[] args) {
		A a1 = new A(0);
		B b1 = new B();
		C c1 = new C(); // línea 3
	}
}
```

a) Indicar <u>solo las líneas imprescindibles</u> a añadir en las clases `A`, `B` y `C`, para que se produzca el ensamblaje correcto del sistema, SIN modificar la clase `P`. (0.30 ptos)

b) Suponiendo que a continuación de la línea 3 del método `main`, añadimos las siguientes líneas:

```
int x = a1.obtener();          // linea 4
B b2 = new B(c1.obtener());    // linea 5
B.valor = 10;                  // linea 6
```

Indicar si son o no correctas en compilación, <u>explicando brevemente el motivo</u>. (0.15 ptos)

c) Si las anteriores clases se compilan dejando los binarios resultantes en el directorio `./bin` y además necesitamos para una correcta ejecución, algunos paquetes y clases contenidos en las bibliotecas `./lib/math-1.0.jar` y en el directorio `./bin2` indicar: (Nota: suponemos que el sistema operativo es GNU/Linux o Mac, pero se puede dar la solución para Windows). (0.25 ptos)

c.1) ¿Qué valor tenemos que dar al CLASSPATH para una correcta ejecución?

c.2) ¿Cómo ejecutamos en línea de comandos la clase principal `P` desde el directorio actual?

c.3) ¿Qué estructura de directorios y ficheros deberíamos tener en el directorio `./bin`?

a)

En la clase A:

```
package x.y;
import  x.y.z.B;
import  z.y.x.C;
```

En la clase B:

```
package x.y.z;
import z.y.x.C;
```

En la clase C:

```
package z.y.x;
```

b)

```
int x = a1.obtener();          // linea 4 // ERROR el método obtener() no está definido en la clase A
B b2 = new B(c1.obtener());    // linea 5 // ERROR constructor con entero no definido en B
B.valor = 10;                  // linea 6 // B.valor no es visible en la clase por ser amigable
```

c)

c.1) `EXPORT CLASSPATH=./bin;./lib/math-1.0.jar;./bin2`
c.2) `java x.P`

c.3)

```
./bin
   |--- x
   |     |---y
   |     |    |---z
   |     |    |     |---B.class
   |     |    |
   |     |    |---A.class
   |     |
   |     |---P.class
   |
   |---z
         |---y
             |---x
                 |---C.class
```

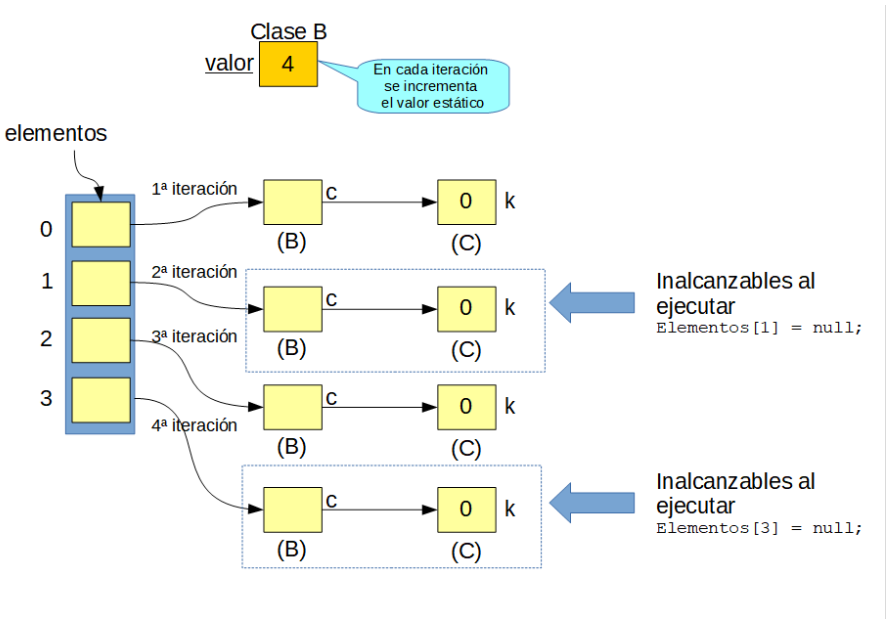
2. **A partir del siguiente código que utiliza las clases `B` y `C` mostradas en el Ejercicio 1:**

```
public class Principal {

	public static void main(String[] args) {
		B[] elementos = new B[4];
		int i = 0;
		do {
			elementos[i] = new B();
			System.out.println(elementos[i].obtener());
			i++;
		}
		while(i < 4);
		// Finalización del bucle
		B clon1 = elementos[0].clone(); // superficial
		elementos[1] = null;
		B clon2 = elementos[2].clone(); // superficial
		elementos[3] = null;
		// Justo antes de abandonar el main
	}
}
```

a.1) Explicar de forma razonada e incluyendo un dibujo, cuántos objetos, en qué orden y de qué tipo se han generado, justo al llegar al comentario `// Finalización del bucle`. (0.20 ptos)
a.2) Mostrar la salida en pantalla generada por dicho codigo. (0.10 ptos)
a.3) Justo antes de finalizar el método `main` ¿qué objetos quedan inalcanzables? (0.10 ptos)
a.4) Suponiendo que el método `clone()` de la clase `B` existe y realiza una clonación superficial. Explicar cuántos objetos nuevos y de qué tipo se generan en las clonaciones previas. ¿Y si la clonación fuese profunda? Explicar incluyendo un dibujo. (0.30 ptos)

a.1)



a.2)

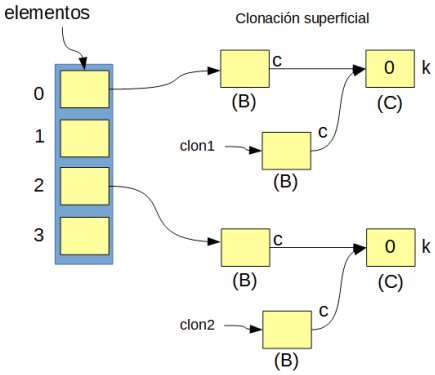
```
1
2
3
4
```

a.3) 4 objetos (ver figura previa)

Los objetos `B` y `C` instanciados y conectados en `elementos[1];`
Los objetos `B` y `C` instanciados y conectados en `elementos[3];`

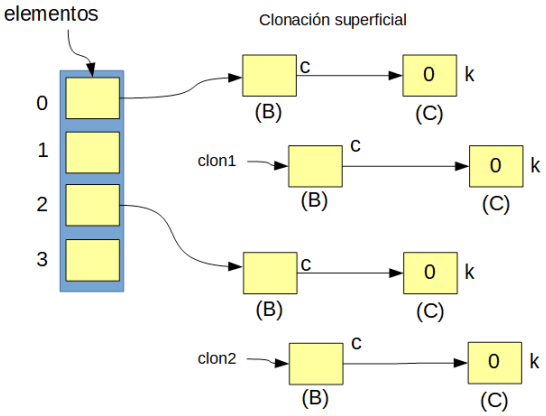
a.4)
Cuando es superficial:

- **Al clonar `elementos[0]` se genera un nuevo objeto `B` y se comparte el objeto `C`.**
- **Al clonar `elementos[1]` se genera un nuevo objeto `B` y se comparte el objeto `C`.**
- **Por lo tanto se crean 2 nuevos objetos.**



Cuando es profunda:

- **Al clonar `elementos[0]` se genera un nuevo objeto `B` y un nuevo objeto `C`.**
- **Al clonar `elementos[1]` se genera un nuevo objeto `B` y un nuevo objeto `C`.**
- **Por lo tanto se crean 4 nuevos objetos.**



3. 

Se quiere dar una implementación muy simplificada de una tarjeta de débito. Con un límite máximo de saldo a acumular y no pudiendo extraer más dinero que el saldo existente actual. Supondremos, por simplificación, que las cantidades siempre son números positivos. Construir la siguiente clase en Java, perteneciente al paquete `mepro.banco`:

- **Clase `TarjetaDebito` con constructor y métodos públicos que permitan:** (0.60 ptos)
  - **Constructor público que recibe el saldo inicial y el límite de saldo. Si el saldo inicial es mayor que el limite dado, se toma como límite dicho saldo inicial.**
  - **Método `public void ingresarDinero(float cantidad)` que añade la cantidad al saldo actual. Si se sobrepasa el límite de saldo, el saldo actual se acota a dicho máximo.**
  - **Método `public void retirarDinero(float cantidad)` que elimina la cantidad del saldo actual. Si no hay dinero suficiente, se deja el saldo a cero.**
  - **Método `public float obtenerSaldo()`: devuelve el saldo actual en la cuenta.**

Se pueden incluir métodos privados si se considera adecuado para su resolución.

```
package mepro.banco;

public class TarjetaDebito {

	private float saldo;

	private float limiteSaldo;

	public TarjetaDebito(float saldoInicial, float limiteSaldo) {
		this.saldo = saldoInicial;
		if (saldoInicial < limiteSaldo) {
			this.limiteSaldo = limiteSaldo;
		}
		else {
			this.limiteSaldo = saldoInicial;
		}
	}

	public void ingresarDinero(float cantidad) {
		if (cantidad + saldo <= limiteSaldo) {
			saldo += cantidad;
		}
		else {
			saldo = limiteSaldo;
		}
	}

	public void retirarDinero(float cantidad) {
		if (saldo - cantidad > 0) {
			saldo -= cantidad;
		}
		else {
			saldo = 0;
		}
	}

	public float obtenerSaldo() {
		return saldo;
	}
}
```