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

METODOLOGÍA DE LA PROGRAMACIÓN **5-FEBRERO-2021 – 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:**

```
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;
        }
}
```

```
public class B {

        public static int valor;
        private C c;

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

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

```
public class C {
        private int k;

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

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

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:

```
C c2 = A.generar();          // linea 4
C c3 = new C();              // linea 5
B.valor = B.valor + 1;       // linea 6
```

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

c) Si los anteriores ficheros fuentes están el directorio `./src`, se compilan dejando los binarios resultantes en el directorio `./bin` y además se usa una biblioteca `./lib/biblioteca-1.0.jar` 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) ¿Cómo ejecutamos en línea de comandos la clase principal `P` desde el directorio actual? (Suponemos que el CLASSPATH está ya configurado).

c.2) ¿Qué estructura de directorios y ficheros concretos deberíamos tener en el directorio actual?

a)

En la clase A:

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

En la clase B:

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

En la clase C:

```
package y.x;
```

b)

```
C c2 = A.generar();    // linea 4 Correcta el método generar es estático y accesible sobre A y devuelve un objeto de tipo C compatible con la variable c2
C c3 = new C();        // linea 5 Correcta el constructor sin argumentos existe por defecto al no existir otros constructores
B.valor = B.valor + 1; // linea 6 Correcta porque valor es una variable estática accesible a nivel de clase que sí puede modificarse.
```

c.1) `java P`

c.2) Nota: `src, bin, lib, x, y, z` son directorios, el resto son ficheros:

```
./src
├──x
│   └──y
│       └──B.java
├──y
│   └──x
│       ├──C.java
│       └──z
│           └──A.java
└──P.java

./bin
├──x
│   └──y
│       └──B.class
├──y
│   └──x
│       ├──C.class
│       └──z
│           └──A.class
└──P.class

./lib
└──biblioteca-1.0.jar
```

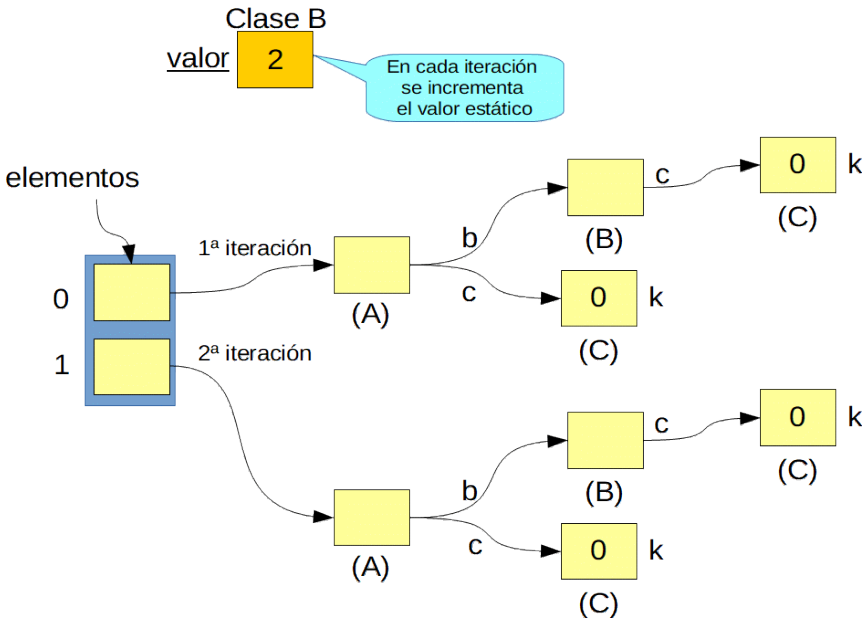
2. **A partir del siguiente código que utiliza las clases `A`, `B` y `C` mostradas en el Ejercicio 1 y suponiendo que la clase `Principal` pertenece al mismo paquete que la clase `A`:**

```
public class Principal {

	public static void main(String[] args) {
		A[] elementos = new A[2];
		int i = 0;
		do {
			elementos[i] = new A(i);
			elementos[i].establecer(A.generar());
			i++;
		}
		while(i < 2);
		// Finalización del bucle
		A clon1 = elementos[0].clone();
		A clon2 = elementos[1].clone();
	}
}
```

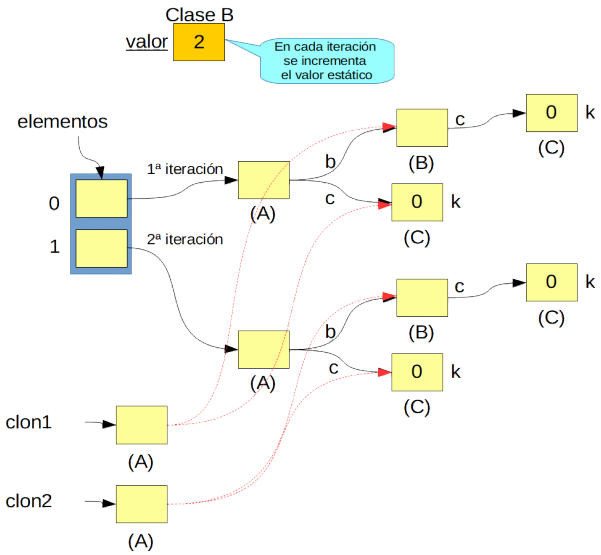
a.1) Explicar de forma razonada e incluyendo un dibujo, cuántos objetos, en qué orden y de qué tipo se han generado, justo al llegar al comentario `// Finalización del bucle`. (0.30 ptos)
a.2) Suponiendo que el método `clone()` de la clase `A` existe y realiza una clonación superficial: explicar cuántos objetos nuevos y de qué tipo se generan en las clonaciones previas. (0.15 ptos)
a.3) Suponiendo ahora que el método `clone()` realiza una clonación profunda: explicar cuántos objetos nuevos y de qué tipo se generan en las clonaciones previas. (0.15 ptos)

a.1)

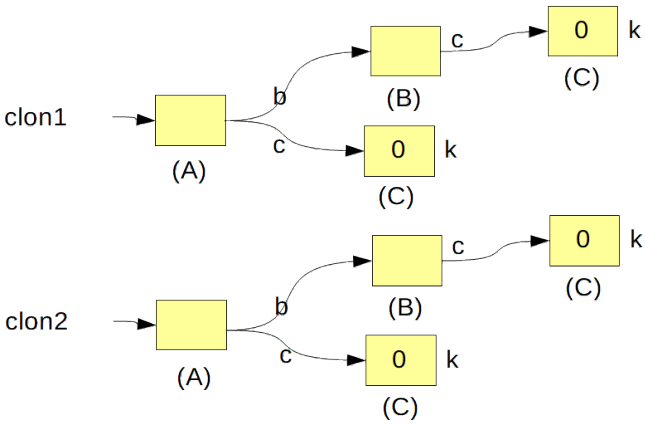


Se crea 1 objeto array `elementos` con dos posiciones para guardar objetos de tipo A.
En cada iteración se crea 1 objeto A que a su vez instancia 1 objeto de tipo B y que transitivamente crea 1 objeto C. Al llamar al método generar se crea 1 objeto de tipo C que se conecta también con el objeto tipo A creado previamente.
1 objeto array
2 objetos A
2 objetos B
4 objetos C

a.2) Al realizarse una clonación superficial se crean 2 nuevos objetos A, pero el resto de objetos se comparten por lo tanto se crean solo 2 objetos A nuevos. Nota: el dibujo no se pide, pero se incluye para aclarar la solución.



a.3) Al realizarse una clonación profunda se clonan TODOS los objetos, siguiendo las referencias en profudidad duplicando el número de objetos a 2 objetos A, 2 objetos B y 4 objetos C nuevos. Estos objetos son independientes de los objetos ya creados previamente, aunque con igual estado. Nota: el dibujo no se pide, pero se incluye para aclarar la solución.



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

3. **Se quiere dar una implementación muy simplificada de una tarjeta de crédito. Sin un límite máximo de saldo a alcanzar y pudiendo extraer dinero, incluso llegando a tener saldo negativo (asumimos una deuda). El saldo negativo puede crecer mientras los intereses del total de deuda asumida estén por debajo del 2% del saldo inicial, con el que se abrió la cuenta. Los intereses a pagar en cada momento se calculan como el 10% de la cantidad negativa en el saldo, o bien cero si el saldo es positivo (no hay deuda). Construir la siguiente clase en Java perteneciente al paquete `mepro.banco`:**

- **Clase `TarjetaCredito` con constructor y métodos públicos que permitan:** (0.70 ptos)
  - **Constructor público que recibe el saldo inicial con el que se crea la cuenta.**
  - **Método `public float calcularIntereses()` calcula los intereses sobre el saldo actual.**
  - **Método `public void retirarDinero(float cantidad)` que decrementa la cantidad del saldo actual, siempre y cuando los intereses de la deuda asumida no superen el 2% del saldo inicial con el que se abrió la cuenta. En caso contrario, no se decrementa la cantidad. Por ejemplo, con una cuenta abierta con saldo inicial de 100€, los intereses del saldo negativo en la cuenta nunca podrán ser superiores a 2€ (no deberíamos deber nunca más de 20€).**
  - **Método `public float obtenerSaldo()`: devuelve el saldo actual en la cuenta.**

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

```java
public class TarjetaCredito {

	private float saldoInicial;

	private float saldo;

	private static final float INTERESES_EN_NEGATIVOS = 0.1F; // 10%

	private static final float LIMITE_SOBRE_SALDO_INICIAL = 0.02F; // 2%

	public TarjetaCredito(float saldoInicial) {
		this.saldoInicial = saldoInicial;
		this.saldo = saldoInicial;
	}

	public float calcularIntereses() {
		if (obtenerSaldo() < 0) {
			return calcularIntereses(obtenerSaldo());
		}
		return 0.0F;
	}

	private float calcularIntereses(float cantidad) {
		return Math.abs(cantidad * INTERESES_EN_NEGATIVOS);
	}

	public void ingresarDinero(float cantidad) {
		saldo += cantidad;
	}

	public void retirarDinero(float cantidadRetirada) {
		float intereses = calcularIntereses(obtenerSaldo()-cantidadRetirada);
		float interesesMaximos = obtenerSaldoInicial()*LIMITE_SOBRE_SALDO_INICIAL;
		if ( intereses < interesesMaximos) {
			this.saldo -= cantidadRetirada;
		}
	}

	public float obtenerSaldo() {
		return saldo;
	}

	private float obtenerSaldoInicial() { // se admite no incluirlo y acceder directamente al atributo
		return saldoInicial;
	}
}
```