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

**METODOLOGÍA DE LA PROGRAMACIÓN** **26-NOVIEMBRE-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:**

```java
package mepro.exam1;

public class A {

	B b;
	private C c;

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

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

```java
package mepro.exam2;

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

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

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

```java
package mepro.exam2;

public class C {
	private int k;

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

```java
package mepro.exam1;

public class P {
	public static void main(String[] args) {
		A a1 = new A();
		B b1 = new B();
		C c1 = new C(); // línea 3
	}
}
```

**a) Indicar <u>solo las líneas imprescindibles a añadir</u> en la clase `P`, para que se produzca el ensamblaje correcto del sistema, SIN modificar el resto de clases.** (0.10 ptos)

**b) Realizadas las modificaciones previas, y suponiendo que a continuación de la línea 3 del método `main`, añadimos las siguientes líneas:**

```
a1.b = null;                          // linea 4
b1.c = c1;                            // linea 5
System.out.println(b1.obtener());     // linea 6
c1 = A.generar();                     // linea 7
B.valor = 42;                         // linea 8
```

**Indicar si son o no correctas en compilación, <u>explicando siempre brevemente el motivo</u>.** (0.20 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/Jcolor-5.0.0.jar` y `./lib/kamisado-gui-1.0.0.jar` indicar:** (0.40 ptos)

**c.1) ¿Cómo ejecutamos en línea de comandos la clase principal `P` desde el directorio actual configurando ademas correctamente la opción `-cp` o `-classpath`?**

**c.2) ¿Qué estructura completa de directorios y ficheros deberíamos tener en el directorio raíz de nuestro proyecto, conteniendo <u>todos los ficheros necesarios</u> mencionados previamente?**

**Nota: suponemos que el sistema operativo es GNU/Linux o Mac, pero se puede dar la solución para Windows.**

**a) Solo se solicitan añadir las líneas de P, no en el resto de clasess (se aclaró en la realización de la prueba).**

```
import mepro.exam2.B;
import mepro.exam2.C;
```

**b)**

```
a1.b = null;                          // acceso a atributo amigable CORRECTO desde el mismo paquete
b1.c = c1;                            // acceso INCORRECTO a atributo private c
System.out.println(b1.obtener());     // acceso INCORRECTO a método amigable desde otro paquete
c1 = A.generar();                     // acceso a método público estático con la clase CORRECTO
B.valor = 42;                         // acceso INCORRECTO a atributo estático amigable valor
```

**c.1)**

```
java -cp ./lib/*:./bin mepro.exam1.P
```

**o**

```
java -cp ./lib/Jcolor-5.0.0.jar:./lib/kamisado-gui-lib-1.0.0.jar:./bin mepro.exam1.P
```

**o**
**cambiando -cp por -classpath**

**c2.)**

```
./src
   |--- mepro
        |---exam1
        |    |---A.java
        |    |---P.java
        |---exam2
             |---B.java
             |---C.java

./bin
   |--- mepro
        |---exam1
        |    |---A.class
        |    |---P.class
        |---exam2
             |---B.class
             |---C.class
./lib
   |--- Jcolor-5.0.0.jar
   |--- kamisado-gui-lib-1.0.0.jar
```

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

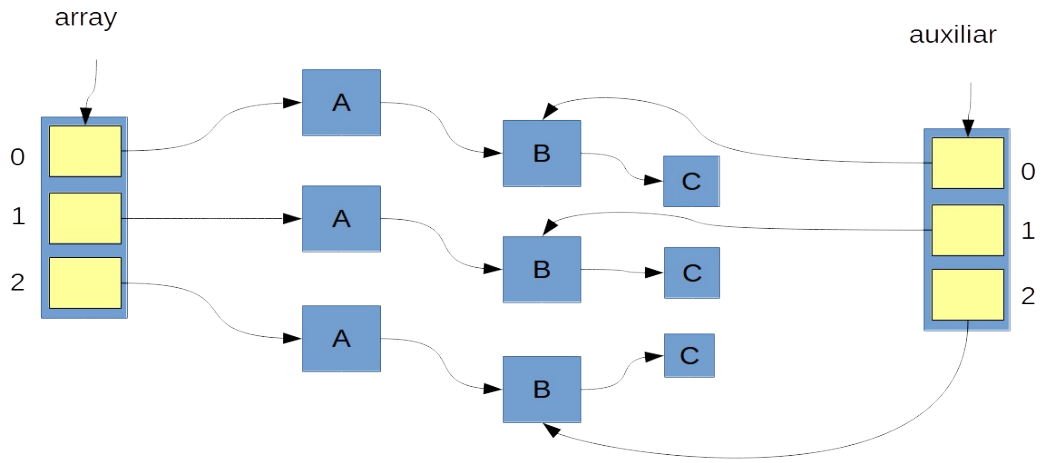
```
// se omiten la declaración de paquete e importaciones
public class Principal {

	public static void main(String[] args) {
		A[] array    = new A[3];
		B[] auxiliar = new B[3];
		for (int i = 0; i < array.length; i++) {
			array[i] = new A();
			auxiliar[i] = array[i].b;
		}
		// Línea FB1
		array[0] = null;
		array[2] = null;
		auxiliar[1]  = null;
		// Línea FA1
		A[] otroArray = array.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 `// Línea FB1`.** (0.25 ptos)
**a.2) Al llegar a la línea `// Línea FA1` indicar razonadamente sobre el dibujo previo, qué objetos pasan a ser inalcanzables y el motivo.** (0.20 ptos)
**a.3) Suponiendo que el método `clone()` realiza una clonación profunda del *array*, explicar cuántos objetos nuevos y de qué tipo se generan.** (0.15 ptos)

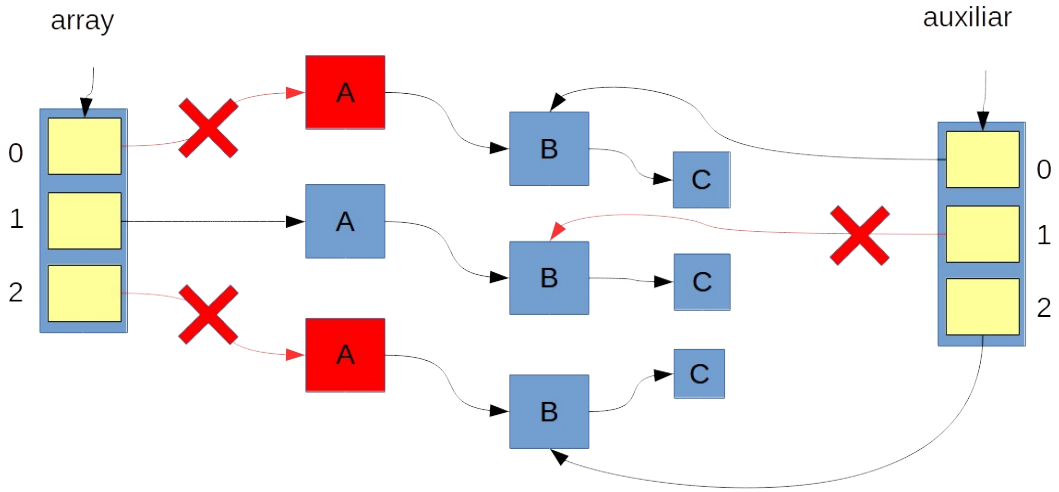
[1]**a.1) Se crean dos objetos de tipo array (array y auxiliar) y se reserva la memoria para las referencias a sus elementos. En el bucle for se crean en 3 iteraciones objetos de tipo A que a su vez instancian objetos de tipo B y C. A la par se inicializa en auxiliar con los objetos B creados previamente.**

**Por lo tanto tenemos 2 arrays, 3 objetos A, 3 objetos B y 3 objetos C, tal y como se muestra a continuación.**




1 Las explicaciones son más extensas de lo solicitado y no es necesario incluir todos los dibujos. Se incluyen para que la solución sea lo más explicativa posible.

**a.2) Al llegar a la línea FA1 tenemos la siguiente situación en cuanto a los objetos:**

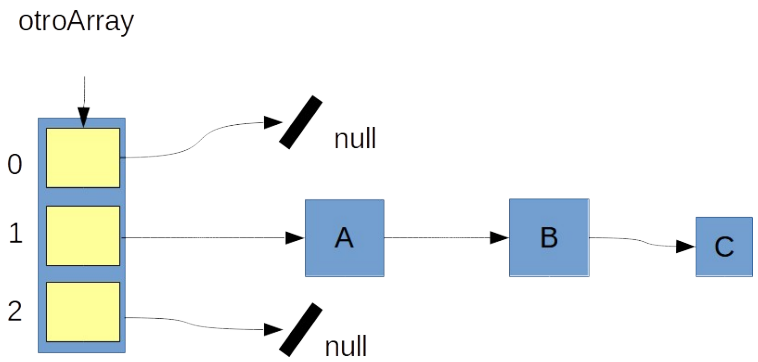



**En este ámbito las referencia vivas de inicio, son las de los dos arrays (`array` y `auxiliar`), y los objetos alcanzables se calculan a partir de dichas referencias.**

**Las referencias con una cruz roja "desparecen" al asignarse valor `null`, y los 2 objetos de tipo A apuntados por `array[0]` y `array[2]` pasan a ser inalcanzables.**

**El resto de objetos siguen siendo todos alcanzables, incluso el objeto `B` al que apuntaba `auxiliar[1]`, puesto que sigue siendo alcanzable a través de `array[1]` y el objeto `A` al que apunta dicha referencia.**

**a.3) Al clonar en profundidad array tendríamos un nuevo array donde en las posiciones 0 y 1 no tendríamos objetos (porque teníamos valores null originalmente y no hay nada que clonar), mientras que en la posición array[1] se clonaría un objeto A, otro B y otro C como teníamos en el array original.**



3. **Se quiere dar una implementación de una secuencia ordenada de colores. Por ejemplo:**


**Los colores están definidos en una enumeración `mepro.colores.Color` con valores `ROJO`, `AZUL`, `AMARILLO`, ya resuelta. Construir solo la siguiente clase en Java perteneciente al paquete `mepro.colores`:**

- **Clase `Secuencia` con constructor y métodos públicos que permitan:** (0.70 ptos)
  - **Constructor público que recibe un *array* de objetos de tipo `Color`, y asigna cada uno de los colores en la posición correspondiente de la secuencia ordenada.**
  - **Método `public int contarCambios()` que cuenta el número de cambios de color que se producen entre posiciones consecutivas, recorriendo de izquierda a derecha la secuencia. Se considera un cambio de color cuando el color actual es distinto al color anterior.**
  - **Método `public void reemplazar(Color previo, Color nuevo)` que reemplaza las apariciones del color previo en la secuencia, con el color nuevo, modificando el estado de la secuencia.**
  - **Método `public Secuencia invertir()` que devuelve una nueva secuencia con los colores en orden inverso a la secuencia actual.**

**Se pueden incluir métodos privados si se considera adecuado para su resolución, pero no es obligatorio.**

```java
package mepro.colores;

public class Secuencia {

	private Color[] colores;

	public Secuencia(Color[] colores) {
		this.colores = new Color[colores.length];
		for (int i = 0; i < colores.length; i++) {
			this.colores[i] = colores[i];
		}
	}

	public int contarCambios() {
		int contador = 0;
		for (int i = 0; i < colores.length-1; i++) {
			if (this.colores[i] != colores[i+1]) {
				contador++;
			}
		}
		return contador;
	}

	public void reemplazar(Color previo, Color nuevo) {
		for (int i = 0; i < colores.length; i++) {
			if (this.colores[i] == previo) {
				this.colores[i] = nuevo;
			}
		}
	}

	public Secuencia inversa() {
		Color[] nuevosColores = new Color[colores.length];
		for (int i = 0; i < colores.length; i++) {
			nuevosColores[i] = colores[colores.length-1-i];
		}
		return new Secuencia(nuevosColores);
	}
}
```