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

**METODOLOGÍA DE LA PROGRAMACIÓN** **21-ENERO-2020 – 2ª 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 {
        public static int valor;
        B b;

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

        public static A generar(int i){
                return new A(i);
        }

        int obtener() {
                return valor * b.obtener();
        }
}
```

```java
public class B {
        private int j;
        private C c;

        public B(int j) {
                this.j = j;
                c = new C();
        }

        public int obtener() {
                return j * c.obtener();
        }
}
```

```java
public class C {
        int k;

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

```java
package paquete1;
import paquete1.paquete2.A;
import paquete2.paquete1.C;
public class M {
        public static void main(String[] args) {
                A a1 = new A(0);
                B b1 = new B(0);
                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 `M`.** (0.25 ptos)

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

```java
A.valor = 3;                    // linea 4
int x = a1.obtener();           // linea 5
c1.k = c1.obtener();            // linea 6
```

**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/bib.jar` y `./lib/junit.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) ¿Qué valor tenemos que dar al CLASSPATH para una correcta ejecución?**

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

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

**a)**

**Clase A**

```
package paquete1.paquete2;
import paquete1.B;
```

**Clase B:**

```
pacakge paquete1;
import paquete2.paquete1.C;
```

**Clase C:**

```
package paquete2.paquete1;
```

**b)**

```
A.valor = 3;              // Ok se accede a un atributo público estático
int x = a1.obtener();     // ERROR, no se puede acceder a método amigable obtener()
c1.k = c1.obtener();      // ERROR k es amigable y no se puede acceder. En este caso
no hay error en acceder a obtener() por ser público.
```

**c)**

**c.1)**

```
set CLASSPATH=.\bin;.\lib\*
```

**o bien**

```
set CLASSPATH=.\bin;.\lib\bib.jar;.\lib\junit.jar
```

**c.2)**

```
java  paquete1.M
```

**c.3)**

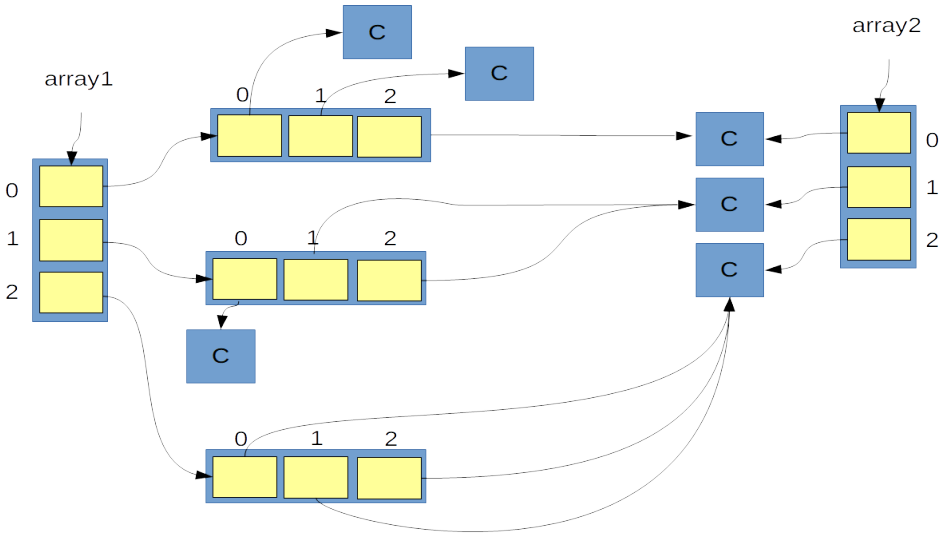
```
.\bin
    |--paquete1
    |        |--- M.class
    |        |--- B.class
    |        |--- paquete2
    |                  |--- A.class
    |
    |
    |--paquete2
           |-----paquete1
                    |--------C.class
```

2. **A partir del siguiente código que utiliza la clase `C` mostrada en el Ejercicio 1:**

```
public class Main {

  private static C[][] array1;

  private static C[] array2 = {new C(), new C(), new C()};

  public static void main(String[] args) {

        array1 = new C[array2.length][array2.length];

        for (int fila = 0; fila < array1.length; fila++) {
            for (int columna = 0; columna < array1[fila].length; columna++) {

                if ((fila + columna) >= 2) {
                   array1[fila][columna] = array2[fila];
                }
                else {
                   array1[fila][columna] = new C();
                }

            }
        }
        // Finalización del bucle
        array1 = null;
  }
}
```

**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) ¿Cuántas referencias apuntan a cada objeto del array2 en ese momento? ¿Cómo se denomina a esta situación y qué problemas puede generar? (0.15 ptos)**
**a.3) Después de ejecutar la última línea, (`array1 = null`), explicar de forma razonada qué y cuántos objetos pasarían a estar inalcanzables, antes de finalizar la ejecución del método `main`. (0.15 ptos)**

**a.1) Se crea un primer `array2` con 3 objetos de tipo C. Luego se crea y reserva memoria para un array de 3x3, inicialmente vacío (todo nulos).**
**En el bucle: si la suma del índice `fila+columna` es 2 o 3 se comparte el objeto (la referencia apunta al mismo objeto de la fila correspondiente en el `array2`), en caso contrario se crea un objeto nuevo de tipo `C`. Por lo tanto, se crean 3 objetos nuevos. Ver figura.**



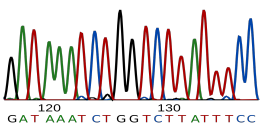

**a.2)**

**En la posición 0, al objeto C apuntan 2 referencias (`array2[0]` y `array1[0][2]`).**
**En la posición 1, al objeto C apuntan 3 referencias (`array2[1]` y `array1[1][1]` y `array1[1][2]`).**
**En la posición 2, al objeto C apuntan 4 referencias (`array2[2]` y `array1[2][0]`,`array1[2][1]`,`array1[2][2]`).**

**A este fenómeno se lo denomina *aliasing dinámico*. Dos o más referencias apuntan al mismo objeto, con el problemade que el estado de un objeto se modifique por una referencia, y se observe el resultado con otra, siendo difícil de intuir el motivo y generando problemas en el código y depuración del mismo.**

**a.3) Pasan a estar inalcanzables los 3 objetos de tipo C creados solo en los bucles con el `array1` en las posiciones `[0][0]`, `[0][1]` y `[1][0]`. Puesto que al perder la referencia a `array1` (pasa a valer `null`) no hay ninguna otra referencia "viva" que nos permita acceder a esos objetos. El resto siguen siendo accesible s a través del `array2` y de las propias referencias en el array que comparten dichos objetos.**

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

3. 


**Se quiere dar una implementación de secuencias de ADN que se componen de una secuencia ordenada de nucleótidos. Construir la siguiente enumeración y clase en Java pertenecientes al paquete `mepro.adn`: (0.70 Ptos)**

- **Enumeración `Nucleotido` que contiene los valores constantes `ADENINA`, `CITOSINA`, `GUANINA` y `TIMINA`. (0.05 Ptos)**
- **Clase `Secuencia` con constructor y métodos que permitan: (0.65 ptos)**
  - **Constructor público que recibe un *array* con la secuencia ordenada (array) de nucleótidos iniciales. Nota: NO se permite el uso de `ArrayList` para su resolución.**
  - **Constructor público que recibe un entero como argumento, rellenando la secuencia de nucleótidos elegidos al azar, tantos como indique dicho número.**
  - **Método `public Secuencia obtenerSecuenciaInvertida()`: devuelve una nueva secuencia de nucleótidos, en orden inverso a la secuencia actual.**
  - **Método `public void invertir()`: cambia el estado de la secuencia actual, invirtiendo todos sus elementos de orden.**

```
public enum Nucleotido {
	ADENINA,
	CITOSINA,
	GUANINA,
	TIMINA;
}

public class Secuencia {

	private Nucleotido[] secuencia;

	public Secuencia(Nucleotido[] valorInicial) {
		secuencia = new Nucleotido[valorInicial.length];
		for(int i = 0; i < valorInicial.length; i++) {
			secuencia[i] = valorInicial[i];
		}
	}

	public Secuencia(int tamaño) {
		secuencia = new Nucleotido[tamaño];
		for (int i = 0; i < tamaño; i++) {
			secuencia[i] = generarAlAzar();
		}
	}

	public Secuencia obtenerSecuenciaInvertida() {
		Nucleotido[] inversa = new Nucleotido[secuencia.length];
		for (int i = secuencia.length -1, j = 0; i >= 0; i--, j++) {
			inversa[j] = secuencia[i];
		}
		return new Secuencia(inversa);
	}

	public void invertir() {
		Secuencia inversa = obtenerSecuenciaInvertida();
		for (int i = 0; i < secuencia.length; i++) {
			secuencia[i] = inversa.secuencia[i];
		}
	}

	private Nucleotido generarAlAzar() {
		return Nucleotido.values()[(int) (Math.random() * Nucleotido.values().length)];
	}

}
```