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

METODOLOGÍA DE LA PROGRAMACIÓN **30-OCTUBRE-2019 – 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 {
        private 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.B;
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`. (0.35 ptos)

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

```
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 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/bib-1.0.jar`, y `./lib/bib-2.0.jar` indicar: (Nota: suponemos que el sistema operativo es GNU/Linux o Mac, pero se puede dar la solución para Windows). (0.20 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`?

a)

Clase A:

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

Clase B:

```
package paquete2;
import paquete1.C;
```

Clase C:

```
package paquete1;
```

b)

```
A.valor = 3;              // linea 4   atributo estático no accesible
int x = a1.obtener();     // linea 5   método no accesible obtener()
c1.k = c1.obtener();      // linea 6   SIN ERROR
```

c)

c.1) `set o export CLASSPATH = ./bin;./lib/bib-1.0.jar;./lib/bib-2.0.jar`

Otra alternativa: `CLASSPATH = ./bin;./lib/*`

c.2) `java paquete1.M`

c.3)

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

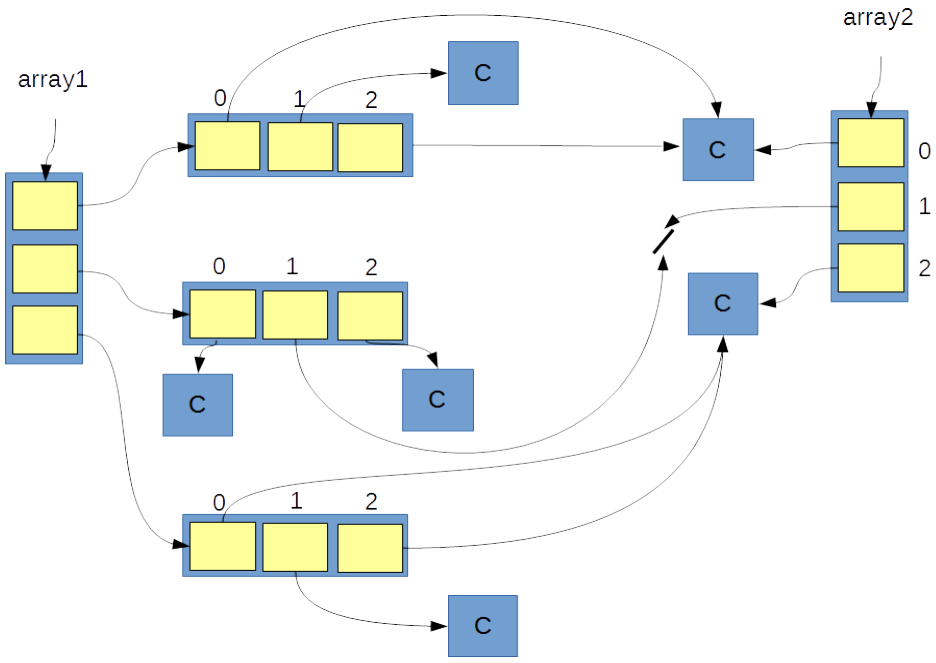
UNIVERSIDAD DE BURGOS
Departamento de Ingeniería Civil
Área de Lenguajes y Sistemas Informáticos

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(), null, 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 == 0) { // % operación resto
                    array1[fila][columna] = array2[fila];
                }
                else {
                    array1[fila][columna] = new C();
                }

            }
        }
        // Finalización del bucle
        array2 = 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`. (0.45 ptos)
a.2) Después de ejecutar la siguiente línea, (`array2 = null`), explicar de forma razonada qué objetos pasarían a estar inalcanzables, justo al finalizar dicha asignación. (0.15 ptos)

a.1)



Explicación:

Se genera una referencia a un array de dos dimensiones (`array1`).

Se genera una referencia a un array de una dimensiones (`array2`), se crea el objeto array y se crean dos objetos de tipo `C`.

Al iniciarse el `main`:

Se reserva memoria para 3x3 referencias a objetos de tipo `C`, creando un objeto array de tres posiciones, donde en cada posición se instancia un nuevo array de 3 posiciones para objetos de tipo `C`. Todavía no se han instanciado objetos de tipo `C` para este array.

En los bucles for:

a) para las posiciones en que la suma de fila y columna es par, se conecta la referencia del `array1` con el objeto ya existente en el `array2`.

- **En la primera iteración/fila se conecta con el objeto `C` ya existente en `array2[0]`**
- **En la segunda iteración/fila se asocia la posición `array1[1][1]` con el valor `null` del `array2[1]`**
- **En la tercera iteración/fila se conecta con el objeto `C` ya existente en el `array2[2];`**
- **No se crean objetos nuevos si no que se conecta con los objetos `C` ya previamente creados.**

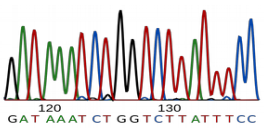
b) para las posiciones en que la suma de fila y columnas es impar, se crea un nuevo objeto de tipo `C` en el `array1`

- **En la primera iteración/fial se crea un objeto nuevo de tipo `C` en la posición `array1[0][1];`**
- **En la segunda iteración/fila se crean dos objetos `C` nuevos y se conecta en `array1[1][0]` y `array1[1][2].`**
- **En la tercera iteración/fila se crea un objeto nuevo de tipo C en la posición `array1[2][1];`**
- **En total se crean 4 nuevos objetos `C`**

Todos los objetos de tipo `C` contienen un atributo k con valor por defecto cero.

a.2) Si `array2 = null;` todos los objetos siguen siendo alcanzables, salvo el array `array2`, puesto que aunque la referencia "desaparece", los dos objetos `C` que estaban almacenados en dicho array, siguen siendo referenciados ahora desde distintas posiciones del `array1`. Es más, son alcanzados por 2 referencias en ambos casos. Por lo tanto, NINGÚN objeto `C` pasa a ser inalcanzable. El `array1` y los `arrays` a los que apuntan también son alcanzables.

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 <u>secuencia ordenada de nucleótidos</u>. 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`**. (0.10 Ptos)**

- **Clase** `Secuencia` **con constructor y métodos que permitan: (0.60 ptos)**
  - **Constructor público que recibe un *array* con la <u>secuencia ordenada</u> de nucleótidos iniciales. Nota: NO se permite el uso de `ArrayList` para su resolución.**
  - **Método** `public void reemplazar(Nucleotido previo, Nucleotido posterior)` **que modifica la cadena actual de ADN, reemplazando todos los nucléotidos de un cierto tipo (previo) por otro (posterior)**
  - **Método** `public int contar(Nucleotido nucleotido)` **devuelve el número de apariciones de un nucleótido concreto en la secuencia.**
  - **Método** `public Secuencia invertir()`**: devuelve una nueva secuencia de nucleótidos en orden inverso.**

```java
package mepro.adn;

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

```java
package mepro.adn;

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 int contar(Nucleotido nucleotido) {
		int contador = 0;
		for (Nucleotido nuc : secuencia) {
			if (nuc == nucleotido) {
				contador++;
			}
		}
		return contador;
	}

	public Secuencia invertir() {
		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 reemplazar(Nucleotido previo, Nucleotido posterior) {
		for (int i = 0; i < secuencia.length; i++) {
			if (secuencia[i] == previo) {
				secuencia[i] = posterior;
			}
		}
	}
}
```