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

METODOLOGÍA DE LA PROGRAMACIÓN **3-FEBRERO-2023 – 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
package mepro.exam2.exam3;

public class A {

	B b;
	private C c;

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

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

```java
package mepro.exam2.exam3.exam4;

public enum C {
	VALOR_X,
	VALOR_Y;
}
```

```java
package mepro.exam3;

public class B {

	public static int valor;
	private C c;

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

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

```java
package mepro.exam1.exam2;

public class P {
	public static void main(String[] args) {

		A a1 = new A();		// línea 1
		B b1 = new B();		// línea 2
		C c1 = null;		// línea 3
	}
}
```

a) Indicar <u>las líneas imprescindibles a añadir en todas las clases</u>, para que se produzca el ensamblaje correcto del sistema, SIN modificar las declaraciones de paquetes actuales, del código actual. (0.20 ptos)

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

```
String texto = b1.obtener();		// línea 4
a1.b = null;				// línea 5
B.valor = 10;				// línea 6
int i = a1.establecer();		// línea 7
C c2 = "VALOR_X";			// línea 8
```

Indicar si son o no correctas <u>en compilación</u>, <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 compilación y ejecución, algunos paquetes y clases contenidos en las bibliotecas `./lib1/junit-5.0.0.jar` y `./lib2/quantik-gui-3.0.0.jar` indicar: (0.30 ptos)

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

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

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

a) Se solicita añadir TODAS las importaciones en TODAS las clases.

En la clase A:

```
import mepro.exam2.exam3.exam4.C;
import mepro.exam3.B;
```

En la clase B:

```
import mepro.exam2.exam3.exam4.C;
```

En la clase P:

```
import mepro.exam2.exam3.A;
import mepro.exam2.exam3.exam4.C;
import mepro.exam3.B;
```

b)

```
String texto = b1.obtener();      // acceso INCORRECTO a método amigable
a1.b = null;                      // acceso INCORRECTO a atributo amigable
B.valor = 10;                     // acceso CORRECTO a atributo público
int i = a1.establecer();          // acceso INCORRECTO, método void no devuelve un valor entero
C c2 = "VALOR_X";                 // INCORRECTO no se puede asignar un String a un tipo enumerado
```

c.1)

```
java -cp ./lib1/junit-5.0.0.jar:./lib2/quantik-gui-3.0.0.jar:./bin
mepro.exam1.exam2.P
```

c.2)

```
./src
   |--- mepro
          |
          |---exam1
          |     |---exam2
          |           |---P.java
          |
          |---exam2
          |     |---exam3
          |           |---A.java
          |           |---exam4
          |                 |---C.java
          |
          |---exam3
                |---B.java
./bin
   |--- mepro
          |
          |---exam1
          |     |---exam2
          |           |---P.class
          |
          |---exam2
          |     |---exam3
          |           |---A.class
          |           |---exam4
          |                 |---C.class
          |
          |---exam3
                |---B.class

./lib1
   |--- junit-5.0.0.jar

./lib2
   |--- quantik-gui-lib-3.0.0.jar
```


2. A partir del siguiente código que utiliza las clases `A`, `B` y enumeración `C`, del Ejercicio 1:

```
package mepro.exam2;
// se omiten las importaciones de otras clases y enumeraciones…
public class Principal {

	public static void main(String[] args) {
	   A[][] array1 = new A[2][2];
	   B[] array2 = new B[3];
	   for (int fila = 0; fila < array1.length; fila++) {
	       for (int columna = 0; columna < array1[fila].length; columna++) {
	             if (columna > 0) {
	                   array1[fila][columna] = new A();
	                   System.out.println(B.valor);
	             }
	       }
	       array2[fila] = new B();
	       System.out.println(B.valor);
	   }
	   // Linea FB1
	   array1[0][1] = null;
	   array1[1][1] = null;
	   array2[1] = null;
	   // Linea FA1
	}
}
```

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`. (0.35 ptos)
a.2) Mostrar y explicar brevemente el motivo de la salida a pantalla generada al llegar al comentario `// Línea FB1`. (0.15 ptos)
a.3) 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.1)

1 *array* de dos dimensiones en Java denominado `array1`, que realmente contiene un *array* de 2 elementos de tipo *array* y otros 2 *array* de longitud 2 para referencias a objetos de tipo `A`.

En dicho *array* las posiciones `[0][0]` y `[1][0]` tienen valores nulos. Las otras dos posiciones contienen objetos de tipo `A`. Cada objeto de tipo `A` tiene a su vez una referencia a un objeto `B` y una referencia a un tipo `C` apuntando a `null`. Cadaa uno de los objetos `B` tiene a su vez un tipo `C` apuntando al valor compartido del tipo enumerado VALOR_X.

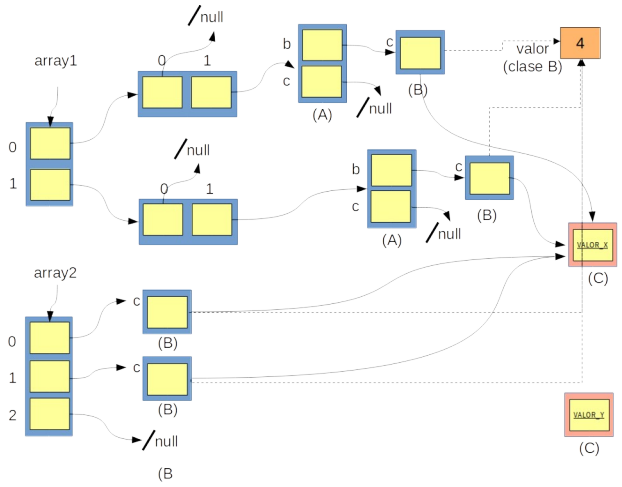
En `array2` tenemos 1 *array* de 3 elementos, con las dos primeras posiciones con objetos de tipo `B` que apuntan a su vez un tipo `C` apuntando al valor compartido del tipo enumerado `VALOR_X`. El tercer valor es `null` porque nunca llega a inicializarse.

A nivel de clase (estático) tenemos:

- el valor estático de `B` que es compartido por todos los objetos
- los dos valores del tipo enumerado `C` que generan dos objetos inmutables con `VALOR_X` y `VALOR_Y` respectivamente que no pueden ser modificados.

Por lo tanto, tenemos:

- 1 *array* con dos referencias a *arrays* de una dimensión.
- 2 *arrays* de una dimensión, con dos referencias a objetos A cada uno.
- 1 array de una dimensión con tres referencias a objetos B.
- 2 objetos `A`.
- 4 objetos `B`.
- 2 objetos de tipo `C`.
- A nivel de clase tenemos un valor compartido pero no es un objeto sino un valor de tipo primitivo `int`.



a.2)

Salida a pantalla:

1
2
3
4

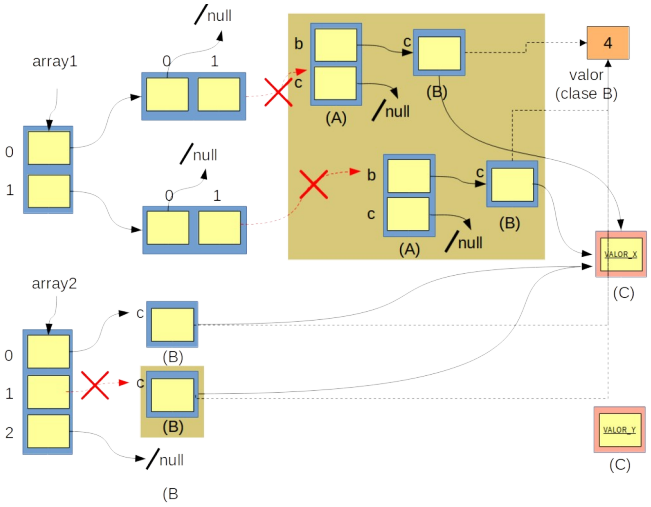
Explicación: al iterar en los casos que `columna > 1` se instancia un objeto `A` que a su vez provoca la invocación al constructor de la clase `B`, donde se incrementa el valor compartido valor. Como esto ocurre dos veces, se mostraría el incremento de dicho valor en pantalla dos veces. Adicionalmente en cada iteración del bucle externo se instancia un objeto B que también provoca un incremento, otras 2 veces. Haciendo un total de 4.

a.3)

En el `array1`, los objetos a los que se apuntaba desde las posiciones `[0][1]` y `[1][1]` quedan desconectados al asignarse un valor `null`, pasando a ser inalcanzables esos 2 objetos de tipo `A`. Indirectamente los dos objetos `B` a los que apuntaban respectivamente, también pasan a ser inalcanzables por lo que tendríamos:

- **2 objetos `A` inalcanzables**
- **2 objetos `B` inalcanzables**

En el `array2`, el objeto `B` en la posición `[1]` también pasa a ser inalcanzable al asignarse un valor null en esa posición.



3. Construir una clase `mepro.examen.TextoSinVocales` que nos permita almacenar texto a partir de un *array* de caracteres. La clase proporcionará: (0.60 ptos)

- **Constructor público que recibe un *array* de caracteres. Supondremos que las letras siempre van en minúsculas, y puede contener cualquier otro tipo de carácter (ej: números, tabuladores, etc.).**
- **Método `public String obtenerTextoSin()` que obtiene la cadena de texto resultante de eliminar todas las vocales.**
- **Método `public String cobtenerTextoOriginal()` que devuelve el texto completo tal y como se inicializó el objeto inicialmente.**

Ejemplo de uso en un método `main` con dicha clase:

```
String texto = "estamos haciendo el examen de mepro";
TextoSinVocales tsv = new TextoSinVocales(texto.toCharArray()); // toCharArray convierte a char[]
System.out.println(tsv.obtenerTextoSin()); // stms hcnd l xmn d mpr
System.out.println(tsv.obtenerTextoOriginal()); // estamos haciendo el examen de mepro
```

Es decisión del alumnado el uso de atributos y métodos adicionales. No se pueden usar los tipos `java.util.List` ni `java.util.ArrayList`, solo *arrays* de Java. Se recuerda que la clase `String` proporciona un constructor con la signatura: `public String(char[] value)`.

Nota: no se corrigen preguntas con tachones o a lápiz.

```java
package mepro.examen;

public class TextoSinVocales {

	private char[] caracteres;

	public TextoSinVocales(char[] caracteres) {
		this.caracteres = new char[caracteres.length];
		for(int i = 0; i < caracteres.length; i++) {
			this.caracteres[i] = caracteres[i];
		}
	}

	public String obtenerTextoSin() {
		char[] tmp = new char[caracteres.length];
		int contador = 0;
		for (char c  : caracteres) {
			if (!esVocal(c)) {
				tmp[contador] = c;
				contador++;
			}
		}
		char[] resultado = new char[contador];
		for (int i = 0; i < contador; i++) {
			resultado[i] = tmp[i];
		}
		return new String(resultado);
	}

	private boolean esVocal(char c) {
		switch(c) {
		case 'a':
		case 'e':
		case 'i':
		case 'o':
		case 'u':
			return true;
		}
		return false;
	}

	public String obtenerTextoOriginal() {
		return new String(caracteres);
	}
}
```