



UNIVERSIDAD DE BURGOS

Departamento de Ingeniería Informática

Área de Lenguajes y Sistemas Informáticos

METODOLOGÍA DE LA PROGRAMACIÓN

18-ENERO-2024 – 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:

```
import mepro.exam2.B;

public class A {

    public static int cont;

    private B b;

    public A(){
        b = new B(A.cont);
        A.cont++;
    }

    public B consultarB() {
        return b;
    }

}
```

```
import mepro.exam1.A;
import mepro.exam1.exam4.E;
import mepro.exam2.exam5.R;

public class B {

    private E e;
    private R r;

    public B(int valor) {
        e = E.VALOR_X;
        r = new R(valor * valor);
    }

    public String aTexto() {
        return r.valor() + "/" + A.cont;
    }

}
```

```
import mepro.exam1.A;
import mepro.exam2.exam5.R;

public class P {
    public static void main(String[] args) {
        A a1 = new A(); // línea 1
        B b1 = new B(0); // línea 2
        R r1 = new R(0); // línea 3
    }
}
```

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

```
public record R(int valor) {
}
```

- a) Indicar las líneas imprescindibles a añadir en todas las clases, enumeraciones y registros, para que se produzca el ensamblaje correcto del sistema, SIN añadir, modificar o borrar las importaciones actuales. (0.25 ptos)

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

```
r1.valor(10); // línea 4
A.cont = 10; // línea 5
R r2 = a1.consultarB(); // línea 6
R r3 = b1.r; // línea 7
System.out.println(b1.aTexto()); // línea 8
```

Indicar si son o no correctas en compilación, explicando siempre brevemente el motivo. (0.15 ptos)

- c) Si los anteriores ficheros fuente se compilan, dejando los binarios resultantes en el directorio ./bin y además se necesita para la correcta compilación y ejecución algunos paquetes y clases contenidos en las bibliotecas ./lib/tresenraya-gui-lib-2.0.1.jar, indicar: (0.20 ptos)

c.1) ¿Cómo ejecutamos en línea de comandos la clase principal P desde el directorio actual (.), configurando correctamente la búsqueda de clases por parte de la Máquina Virtual Java?

c.2) ¿Qué estructura completa de ficheros con su extensión deberíamos tener en el directorio ./bin?

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 declaraciones de paquete TODAS las clases, enumeraciones y registros.

En la clase A:

```
package mepro.exam1;
```

En la clase B:

```
package mepro.exam2;
```

En el tipo enumerado E:

```
package mepro.exam1.exam4;
```

En el tipo registro R:

```
package mepro.exam2.exam5;
```

En la clase P:

```
package mepro.exam2;
```

b)

```
r1.valor(10);           // NO se puede asignar valor a un registro.
A.cont = 10;             // CORRECTO acceso a atributo estático de clase.
R r2 = a1.consultarB();   // INCOMPATIBILIDAD de tipos.
R r3 = b1.r;             // Acceso INCORRECTO a atributo privado.
System.out.println(b1.aTexto()); // Acceso CORRECTO a método accesible público.
```

c.1)

```
java -cp ./lib/tresenraya-2.0.1.jar:./bin mepro.exam2.P
```

c.2)

```
./bin
|--- mepro
|
|---exam1
|   |---A.class
|   |---exam4
|       |---E.class
|
|---exam2
|   |---B.class
|   |---P.class
|   |---exam5
|       |---R.class
```





UNIVERSIDAD DE BURGOS

Departamento de Ingeniería Informática

Área de Lenguajes y Sistemas Informáticos

2. A partir del siguiente código que utiliza las clases A, B, tipo enumerado E y tipo registro R del Ejercicio 1:

```
// se omiten la declaración de paquete e importaciones, NO hay que añadirlas
public class Principal {

    public static void main(String[] args) {
        A[][] array = new A[3][2];
        // Línea FB0
        for (int fila = 0; fila < array.length; fila++) {
            for (int columna = 0; columna < array[fila].length; columna++) {
                if ( (fila + columna) % 2 == 0) {
                    array[fila][columna] = new A();
                    B b1 = array[fila][columna].consultarB();
                    System.out.printf("Texto: %s %n", b1.aTexto());
                }
            }
            System.out.printf("Contador: %d %n", A.cont);
        }
        // Línea FB1
        array[0][0] = new A();
        B b2 = array[1][1].consultarB();
        b2 = null;
        // Línea FB2
    }
}
```

- 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 FB0** y posteriormente al llegar a **// Línea FB1**. (0.35 ptos)
- a.2) Al llegar a la línea **// Línea FB1** mostrar literalmente la salida a pantalla generada. (0.15 ptos)
- a.3) Al llegar a la línea **// Línea FB2** y ANTES de finalizar la ejecución del método **main**, indicar razonadamente, sobre el dibujo previo, qué objetos pasan a ser inalcanzables y el motivo. (0.20 ptos)

Nota: no se solicita una explicación tan detallada por parte del alumnado, pero se extiende la explicación, para no solo resolverlo, sino intentar aclarar todos los detalles del ejercicio, explicando pormenorizadamente también los dibujos incluidos.

a.1) En el main se inicia con un objeto de tipo *array* de cadenas, que llegará vacío sin elementos. Al llegar a la línea FB0 se tiene 1 *array* de dos dimensiones en Java que realmente contiene un *array* de 3 elementos de tipo *array* y otros 2 *arrays* de longitud 2 para referencias a objetos de tipo *A*. Inicialmente en esas seis posiciones tenemos valores nulos, puesto que solo se ha reservado memoria para el *array*.

Al llegar a FB1, en dicho *array* las posiciones [0][1], [1][0] y [2][1] tiene valor nulo (la suma de sus índices es impar). Las otras tres posiciones contienen objetos de tipo *A*. Cada objeto de tipo *A* tiene a su vez una referencia a un objeto *B*. Cada objeto *B* tiene una referencia al mismo valor del tipo enumerado *E*. *VALOR_X*, compartido por todos ellos, y una referencia a un objeto de tipo registro *R*, inicializado con distintos valores en cada caso.

A nivel de clase (estático) tenemos:

- el valor estático de tipo primitivo, *cont* en *A* que es compartido por todos los objetos.
- los dos valores del tipo enumerado *E* que generan dos objetos inmutables con *VALOR_X* y *VALOR_Y* respectivamente que no pueden ser modificados.

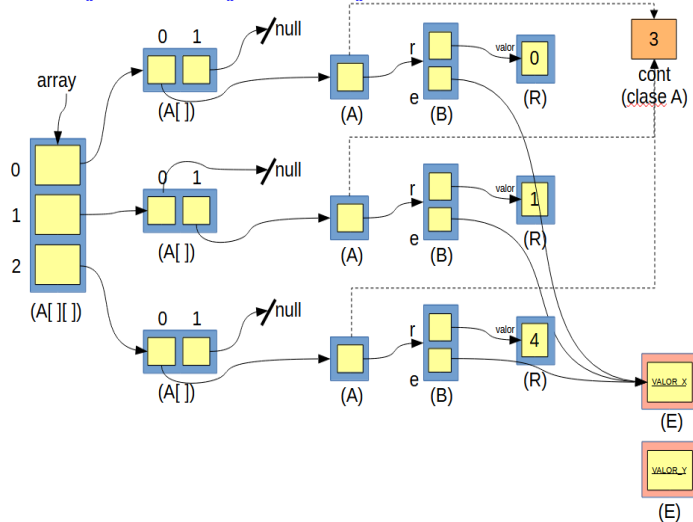
Por lo tanto, tenemos, y sin considerar el argumento *args*, del método *main*:

- 1 *array* con 3 referencias a *arrays* de una dimensión.
- 3 *arrays* de una dimensión, con dos referencias a objetos *A* cada uno.



- 3 objetos A.
- 3 objetos B.
- 3 objetos de tipo registro R.
- A nivel de clase tenemos un valor compartido entero, pero no es un objeto
- Dos objetos inmutables compartidos de tipo enumerado E.

Si dibujamos el conjunto de objetos en memoria tendríamos algo similar a lo siguiente:



a.2) Salida a pantalla:

```
Texto: 0/1
Contador: 1
Texto: 1/2
Contador: 2
Texto: 4/3
Contador: 3
```

a.3)

Al realizar la instanciación y nueva asignación, los objetos a los que se apuntaba desde la posición [0][0] quedan desconectados, pasando a ser inalcanzables: 1 objeto A, 1 objeto B y 1 objeto tipo registro R (como se ve en la siguiente figura a la izquierda en la zona sombreada).

Como resultado de la nueva instanciación, se genera a su vez 1 nuevo objeto A, 1 nuevo objeto B y 1 nuevo objeto R con valor $3*3=9$ y finalmente incrementado el valor estático de `cont` a 4 (se muestra la instantánea en la figura de la derecha en la zona verde). Esto no se pedía explícitamente en el enunciado y no se requería para sumar puntos.

El efecto de la asignación a una variable local como `b2` del objeto al que apuntaba `array[1][1]` y su posterior asignación a un valor nulo, no tiene mayor efecto, puesto que aunque esa referencia se destruye, el objeto sigue siendo alcanzable a través de la variable `array`, a través del objeto de tipo A conectado en `array[1][1]` (ver en la figura de la derecha, la parte inferior).

Si dibujamos el conjunto de objetos antes y después de estas operaciones se tendría lo siguiente:

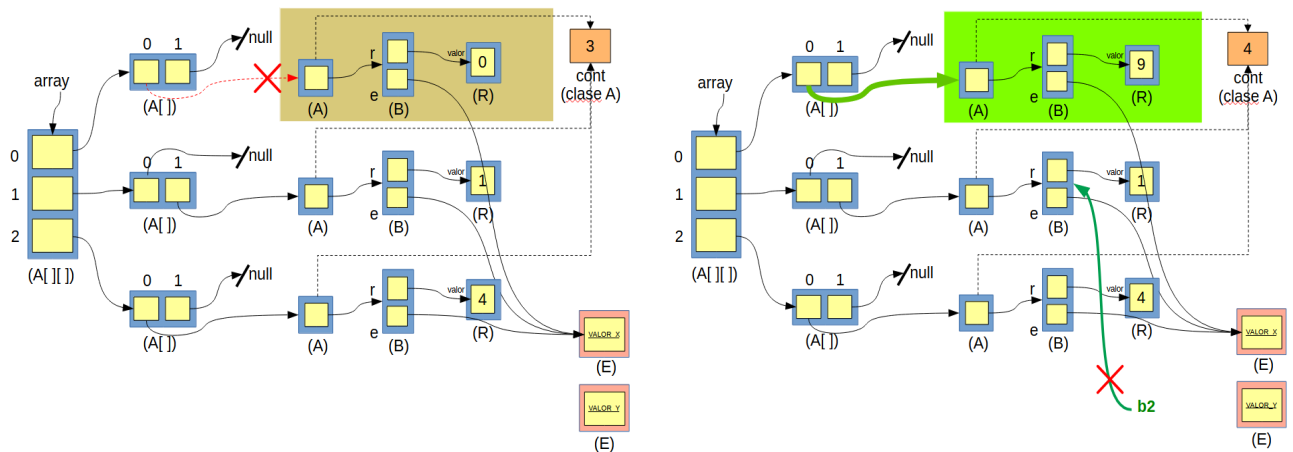




UNIVERSIDAD DE BURGOS

Departamento de Ingeniería Informática

Área de Lenguajes y Sistemas Informáticos



3. Se quiere dar una implementación simplificada de un calefactor, que puede estar encendido o apagado. El estado está definido en una enumeración `mepro.Estado` con valores `ON`, `OFF`, ya resuelta. Se debe construir solo la siguiente clase en Java, perteneciente al paquete `mepro`:

- Clase `Calefactor` con: (0.70 ptos)
 - Constructor público que permita iniciar el calefactor con el coste de facturación de euros por minuto. Dicho valor puede tener decimales. El calefactor siempre se inicia apagado por defecto.
 - Método público `cambiarEstado`, sin argumentos, que permita alternar el estado entre apagado (`ON`) y encendido (`OFF`).
 - Método público `calentar` que recibe como argumento el número de minutos que deben acumularse al total de minutos de funcionamiento, hasta el momento. Solo se acumulan los minutos, si el calefactor está encendido. En caso contrario, se ignora el valor recibido. No retorna ningún valor.
 - Método público `consultarCosteAcumulado`, no recibe argumentos y devuelve el coste en euros hasta el momento del total de minutos acumulados.
 - Método público raíz `main` para poder probar la clase. Realizando las siguientes acciones en este orden: instancia un calefactor, lo pone a calentar X minutos, lo enciende, lo pone a calentar Y minutos, lo apaga y muestra el coste acumulado en euros en pantalla. ¿Qué valor se mostrará en pantalla según los valores elegidos discrecionalmente para el coste en euros por minutos, X e Y?

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

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



```
package mepro;

public class Calefactor {

    private Estado estado;

    private int minutos;

    private double precioHora;

    public Calefactor(double precioHora) {
        this.precioHora = precioHora;
        this.estado = Estado.OFF;
    }

    public void cambiarEstado() {
        estado = estado == Estado.OFF ? Estado.ON : Estado.OFF; // o bien con un if...
    }

    public void calentar(int minutos) {
        if (estado == Estado.ON) {
            this.minutos += minutos;
        }
    }

    public double consultarCosteAcumulado() {
        return minutos * precioHora;
    }

    public static void main(String[] args) {
        Calefactor calefactor = new Calefactor(0.13D); // se elige 0.13D como coste
        calefactor.calentar(10); // se elige X=10
        calefactor.cambiarEstado();
        calefactor.calentar(20); // se elige Y=20
        calefactor.cambiarEstado();
        System.out.printf("Coste: %2.2f€", calefactor.consultarCosteAcumulado());
        // se mostraría 2.60€ en pantalla, puesto que los primeros 10 minutos no cuentan
        // para el coste, puesto que el calefactor estaba inicialmente apagado (OFF).
    }
}
```

