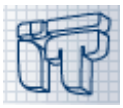


NOMBRE:
APELLIDOS:
NIA:
GRUPO:

Primer parcial

2ª Parte: Problemas (7 puntos sobre 10)

Duración: 70 minutos
Puntuación máxima: 7 puntos
Fecha: 25 de marzo de 2021

Instrucciones para el examen:

- No se permite el uso de libros o apuntes, ni tener teléfonos móviles u otros dispositivos electrónicos encendidos. Incumplir cualquiera de estas normas puede ser motivo de expulsión inmediata del examen.

Ejercicio 1 (2,5 / 7 puntos)

Una tienda de música quiere gestionar su inventario de instrumentos y para ello ha encargado a nuestro equipo de desarrolladores un proyecto Java para realizar esa tarea. Como parte del equipo de desarrollo, se te pide lo siguiente:

Apartado 1.1. Clase Instrumento. Declaración de clase atributos y constructores (0,7 puntos)
Implementa la clase Instrumento teniendo en cuenta la siguiente especificación

- Tiene un atributo (tipo) de tipo carácter accesible sólo dentro de la propia clase que representa el tipo de instrumento. Crea, a modo de ejemplo, una de las 4 constantes que representan los valores posibles de dicho atributo: c - CUERDA, v - VIENTO, p - PERCUSIÓN o e - ELECTRICO. Puedes asumir que las tres restantes ya están creadas.
- Añade atributos para representar el nombre del instrumento (nombre), precio del instrumento (precio) y un número de inventario (idInventario), dicho identificador de inventario se tiene que generar a partir de un atributo de clase (numInstrumentos) que lleva la cuenta de los instrumentos que han entrado en inventario.
- Crea un constructor completo que reciba los parámetros tipo, nombre, y precio y asigne valor a todos los atributos de la clase. Para implementar el constructor puedes asumir que ya están creados los métodos set y get de todos los atributos.

Apartado 1.2. Clase Instrumento. Método set y toString (0,3 puntos)

- Crea los métodos set y get para el atributo tipo, teniendo en cuenta que solo puede tomar los valores; c - CUERDA, v - VIENTO, p - PERCUSIÓN o e - ELECTRICO, y que en caso de no ser un valor válido hay que lanzar una excepción
- Crea un método toString que devuelve un string con el siguiente formato: idInventario;tipo;nombre;precio (Ej.: “1;c;Guitarra;600.0)

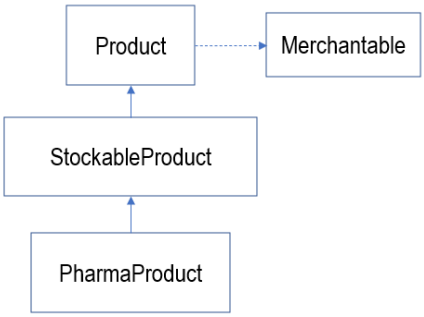
Apartado 1.3. Clase Instrumento. Método main inventario de instrumentos (1 punto)
Crea el método main desde el cual se genere un informe del inventario de instrumentos. Para ello deberás

- Crear un array con al menos 3 instrumentos.
- Calcular el valor del inventario (suma del precio de todos los instrumentos)
- Representarlo en el formato que se indica en la figura. Donde cada instrumento se representa según el formato indicado en el método toString()

```
INVENTARIO
==========
1;c;Guitarra;600.0
2;v;Flauta;300.0
3;p;Tambon;130.0
----------------------------
VALOR TOTAL DEL INVENTARIO: 1030.0
```

Apartado 1.4. Testing (0,5 puntos)
Dado el método setType de la clase Instrumento. Escribe un test unitario como los vistos que clase que cree un instrumento y compruebe que se lanza una excepción en caso de introducir como parámetro un valor incorrecto como por ejemplo el carácter 'm'.

Ejercicio 2 (3 / 7 puntos)

El proyecto de gestión del almacén con el que has estado trabajando ha decidido crear una nueva línea de productos farmaceúticos almacenables llamada PharmaProduct. Esta nueva clase (PharmaProduct) supone una especialización de los productos anteriores (StockableProduct) según la jerarquía que se muestra en la figura. Este tipo de productos tienen características especiales respecto a su fechas de caducidad y temperatura de conservación. La temperatura de conservación afecta además al precio de comercialización del producto. Se pide:	

Apartado 2.1. Merchantable (0,5 puntos)
Implementa la interfaz Merchantable (comercializable) teniendo en cuenta la siguiente especificación

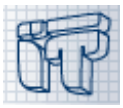
- Tiene un método finalCost que recibe como parámetro el número de unidades solicitadas de un producto (units) y devuelve como resultado el coste final del pedido. Ese resultado dependerá en cada caso del tipo de producto solicitado.

Apartado 2.2. Clase StockableProduct (0,5 punto)

- Declara la clase StockableProduct que implemente la interfaz Merchantable.
- No es necesario que implementes el contenido de la clase únicamente la declaración.

Apartado 2.3. Clase PharmaProduct (1,5 puntos)

- Declara la clase PharmaProduct según la jerarquía indicada en la figura
- Declara los atributos para representar la fecha de caducidad (expiryDate) y la temperatura de conservación (temperature) del producto de modo que no sean accesibles ni modificables desde otras clases. La fecha tendrá formato AAAA-MM-DD, (Ej.: "2022-03-02") y la temperatura se representará con un número decimal.
- Crea un constructor para la clase PharmaProduct que reciba todos los parámetros necesarios para asignar valor a todos los atributos (incluidos los que hereda de su clase padre).



- NOTAS. Puedes suponer que ya existen sin necesidad de implementarlos los siguientes métodos:
    - El constructor de la clase padre: public StockableProduct(String name, String brand, char category, boolean isContable, String measurementUnit, int numUnits, double costPerUnit, double pricePerUnit)
    - Los métodos set/get de los diferentes atributos.

Apartado 2.4. Clase PharmaProduct. Método finalCost (0,5 puntos)

Implementa el método finalCost de la clase PharmaProduct teniendo en cuenta que:

- El precio unitario de cada uno de estos productos es como el precio (pricePerUnit) de un producto cualquiera (StockableProduct) añadiendo una tasa adicional de transporte (transportFee) en función de la temperatura de conservación de cada producto. Por encima de 8ºC no se aplicará tasa alguna, entre 0ºC y 8ºC se aplicará un 20% y por debajo de 0ºC la tasa será del 30%.
- Para calcular el coste total del pedido hay que tener en cuenta el número de unidades de cada producto (units)

Ejercicio 3 (1,5 / 7 puntos)

Apartado 3.1. Recursión (1,5 puntos)

Programa el método recursivo public static int countDigits(int num) que dado un número entero positivo (num) devuelve como resultado la suma de todos sus dígitos. Por ejemplo la llamada en la línea 6 a countDigits (565) daría como resultado 3 y countDigits(1234) daría como resultado 10.

```
3  public class CountDigits {
4      public static void main(String[] args) {
5          int number = Integer.parseInt(args[0]);
6          System.out.println(countDigits(number));
7      }
8
9      public static int countDigits(int num) {
10         //todo
11     }
12 }
```

NOTAS:

- No se permite la conversión de entero a String para resolver el problema.
- Puedes utilizar el hecho de que el número está escrito en base decimal.

SOLUCIÓN DE REFERENCIA (Varias soluciones son posibles)

Ejercicio 1 (2,5 / 7 puntos)

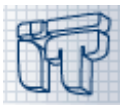
Apartado 1.1. Clase Instrumento. Declaración de clase atributos y constructores (0,7 puntos)

- 0,1 Declaración de clase
- 0,1 Declaración correcta de los atributos privados (tipo, nombre, precio idInventario
- 0,1 por la declaración correcta del atributo de clase numInstrumentos
- 0,4 declaración correcta del constructor
    - Considerar correcto llamada a set o this.atributo = atributo para los que no tienen comprobaciones
    - En el caso de atributo tipo considerar correcto sólo si se llama a set

```
public class Instrumento {
    private static int numInstrumentos = 1;
    private char tipo; // c - CUERDA, v - VIENTO, p - PERCUSIÓN o e - ELECTRICO

    public static final char CUERDA = 'c';
    public static final char VIENTO = 'v';
    public static final char PERCUSION = 'p';
    public static final char ELECTRICO = 'e';

    private String nombre;
    private double precio;
    private int idInventario;

    public Instrumento(char tipo, String nombre, double precio) {
        try {
            setTipo(tipo);
        } catch (Exception e) {
            e.printStackTrace();
        }
        this.nombre = nombre;
        this.precio = precio;
        this.idInventario = Instrumento.numInstrumentos++;
    }
}
```

Apartado 1.2. Clase Instrumento. Método set get y toString (0,3 puntos)

- 0,1 cada método



```
public void setTipo(char tipo) throws Exception {
    switch (tipo) {
    case CUERDA:
    case VIENTO:
    case PERCUSION:
    case ELECTRICO:
        this.tipo = tipo;
        break;
    default:
        throw new Exception("Tipo de instrumento no válido");
    }
}

public char getTipo() {
    return this.tipo;
}

public String toString() {
    return this.idInventario + ";" + this.tipo + ";" +
                this.nombre + ";" + this.precio;
}
```

Apartado 1.3. Método main inventario de instrumentos (1 punto)

- 0,2 declaración correcta del main
- 0,2 creación del array
- 0,2 asignación de elementos al array
- 0,2 calcular el precio total
- 0,2 Imprimir en formato correcto.

```
public static void main(String[] args) {
    Instrumento[] inventario = new Instrumento[3];
    inventario[0] = new Instrumento('c', "Guitarra", 600);
    inventario[1] = new Instrumento('v', "Flauta", 300);
    inventario[2] = new Instrumento('p', "Tambon", 130);
    System.out.println("INVENTARIO");
    System.out.println("==========");
    double valorTotal = 0.0;
    for (int i = 0; i < inventario.length; i++) {
        System.out.println(inventario[i]);
        valorTotal = valorTotal + inventario[i].getPrecio();
    }
    System.out.println("----------------------------");
    System.out.println("VALOR TOTAL DEL INVENTARIO: " + valorTotal);
}
```

Apartado 1.4. Testing (0,5 puntos)

- 0 si no tiene sentido.
- 0,1 por declaración correcta del método de test.
- 0,1 por instanciación del objeto *Instrumento*.
- 0,1 por invocar el método *setTipo()* de modo que genere una excepción.
- 0,2 utilizar *assertThrows()* correctamente.

```
import static org.junit.jupiter.api.Assertions.*;
import org.junit.jupiter.api.Test;

class InstrumentoTest {

    @Test
    void testSetTipo() {
        Instrumento miInstrumento = new Instrumento('c', "Guitarra", 600);
        assertThrows(Exception.class, ()-> {miInstrumento.setTipo('m');});
    }
}
```

Ejercicio 2 (3 / 7 puntos)

Apartado 2.1. Merchantable (0,5 puntos)

Implementa la interfaz

- 0 si no demuestra conocimiento de interfaz porque pone abstract en la declaración o implementa el método o pone {} en vez de ;
- No penalizar si se pone *public* en el método aunque al ser una interfaz no es necesario porque todos sus métodos lo son.

```
public interface Merchantable {
    abstract double finalCost(int units);
}
```

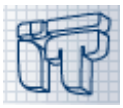
Apartado 2.2. Clase StockableProduct (0,5 punto)

- 0 si no pone abstract o no demuestra conocimiento de lo que es una clase abstracta. No es necesario que ponga el método de la interfaz. Pero no penalizar si lo pone de forma correcta.
- Penalizar con -0,1 si pone abstract pero no pone implements

```
public abstract class StockableProduct extends Product implements Merchantable {
```

Apartado 2.3. Clase PharmaProduct (1,5 puntos)

- 0,3 Declaración de la clase PharmaProduct.(abstract, extends, implements)
  - 0 si no se extiende la clase StockableProduct.
- 0,2 Declaración de atributos privado expiryDate y temperature
  - 0 si hay algún fallo en los modificadores o tipos de datos.
- 0,3 declaración del constructor
  - -0.2 si no incluye parámetros de la clase padre
  - -0.1 si no incluye parámetros para dar valor a los atributos propios
- 0,4 Llamada al constructor de la clase padre
- 0,3 Asignación de atributos propios. Se considera correcto si lo hacen con this y si lo hacen con set porque les hemos dicho que supongan que existen pero no les hemos dicho que requieran comprobaciones.



```
public class PharmaProduct extends StockableProduct {
    private String expiryDate;
    private double temperature;

    public PharmaProduct(String name, String brand, char category,
                         boolean isContable, String measurementUnit,
                         int numUnits, double costPerUnit, double pricePerUnit,
                         String expiryDate, double temperature) {

        super(name, brand, category, isContable, measurementUnit,
              numUnits, costPerUnit, pricePerUnit);

        this.expiryDate = expiryDate;
        this.temperature = temperature;
    }
}
```

Apartado 2.4. Método finalCost (0,5 puntos)

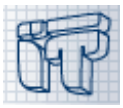
- Declaración e implementación del método finalCost() de acuerdo a la definición para PharmaProduct
  - Penalizar con -0,2 si se aplica la tasa sobre el coste por unidad en lugar de sobre el precio por unidad.

```
public double finalCost(int units) {
    double cost = 0;
    double transportFee = 0;

    if (this.temperature > 0 && this.temperature <= 8) {
        transportFee = 0.2;
    } else if (this.temperature < 0) {
        transportFee = 0.3;
    }

    cost = (this.getPricePerUnit() + (this.getPricePerUnit() * transportFee)) * units;

    return cost;
}
```

Ejercicio 3 (1,5 / 7 puntos)

Apartado 3.1. Recursión (1,5 puntos)

- (0,25) Caso trivial correcto (if num<10).
  - Penalizar con un 0 si no se pone el caso trivial correcto.
- (1,25) Caso Recursivo correcto.
  - Si se retorna el valor correctamente, es decir, el 1 (0,25)
  - Si se llama a la función de modo recursivo “ContarCifras” correctamente (1)
  - Si no se sabe hacer en general el caso recursivo un 0.



```java
public class CountDigits {
    public static void main(String[] args) {
        int number = Integer.parseInt(args[0]);
        System.out.println(countDigits(number));
    }

    public static int countDigits(int num) {
        if (num < 10) {
            return 1;
        } else {
            return 1 + countDigits(num / 10);
        }
    }
}
```