UD4 – Prueba de validación 1 – Herencia y referencias estáticas

Resultados de aprendizaje y criterios de evaluación:

2. Escribe y prueba programas sencillos, reconociendo y aplicando los fundamentos de la programación orientada a objetos.

a) Se han identificado los fundamentos de la programación orientada a objetos.
c) Se han instanciado objetos a partir de clases predefinidas.
d) Se han utilizado métodos y propiedades de los objetos.
e) Se han escrito llamadas a métodos estáticos.
f) Se han utilizado parámetros en la llamada a métodos.
g) Se han incorporado y utilizado librerías de objetos.
h) Se han utilizado constructores.

4. Desarrolla programas organizados en clases analizando y aplicando los principios de la programación orientada a objetos.

a) Se ha reconocido la sintaxis, estructura y componentes típicos de una clase.
b) Se han definido clases.
c) Se han definido propiedades y métodos.
d) Se han creado constructores.
e) Se han desarrollado programas que instancien y utilicen objetos de las clases creadas anteriormente.
f) Se han utilizado mecanismos para controlar la visibilidad de las clases y de sus miembros.
g) Se han definido y utilizado clases heredadas.
h) Se han creado y utilizado métodos estáticos.
i) Se han creado y utilizado conjuntos y librerías de clases.

Recuerda:

- Sólo se corregirán aquellos trabajos entregados a tiempo.
- El código debe entregarse en un zip que contenga el código.
- Está permitido el uso de apuntes y prácticas realizados en clase.
- Está terminantemente prohibido el uso de aplicaciones de mensajería o de inteligencia artificial para la realización de la prueba. El uso de alguna de estas herramientas conllevará el suspenso de la prueba.
- Recuerda utilizar una estructura adecuada de paquetes en un único proyecto que permita agrupar todas las actividades con una sola clase ejecutable.

1. Crea una clase en Java que disponga de los siguientes métodos estáticos y úsalos en la clase ejecutable para demostrar su correcto funcionamiento usando los ejemplos citados.

a) *sumarElevadosN*: Debe recibir un parámetro positivo que será el valor de N y devolver el resultado de sumar todos los números enteros comprendidos entre 1 y N, ambos incluidos, elevados a la cantidad enésima

Por ejemplo: Si el método recibe como parámetro 6, debe devolver el resultado de la siguiente operación: $1^n + 2^n + 3^n + 4^n + 5^n + 6^n$.

Ej1: 3 → 14
Ej2: 6 → 91
Ej3: 73 → 132349

b) *calcularMediaNotas*: Debe recibir un parámetro array de números enteros positivos y debe devolver el resultado de calcular la media de los números enteros comprendidos en dicho array.

Por ejemplo: Si el método recibe como parámetro [10, 4, 6, 1, 7], debe devolver el resultado de la siguiente operación: (10 + 4 + 6 + 1 + 7) / 5.

Ej1: [10, 4, 6, 1, 7] → 5’6
Ej2: [4, 7, 9, 2, 3, 7, 8] → 5’71

2. Atendiendo a los principios de herencia en Java, implementa las siguientes clases:

Crea una clase que represente a un **electrodoméstico** y cumpla con estas características:

a) Debe contener los siguientes atributos:

- Marca
- Modelo
- Peso
- Potencia (en kW)
- Está Encendida

b) Debe disponer de un constructor y los métodos de acceso de todos los atributos, exceptuando el último.

c) Debe disponer de un método que le permita encender el electrodoméstico y otro que le permita apagarlo. Estos métodos deben modificar el valor del atributo correspondiente.

A continuación, crea dos clases **nevera** y **microondas** que cumplan con los siguientes requisitos:

a) Estas clases deben heredar de la clase electrodoméstico anterior.

- La nevera debe tener un atributo adicional que haga referencia a la temperatura que posee.
- El microondas debe tener un atributo adicional que haga referencia a si tiene la puerta abierta.

b) Deben sobrecargar el método que permite encender a los objetos de tipo electrodoméstico.

- En el caso de la nevera, este método debe encender el electrodoméstico y mostrar por pantalla un mensaje “¡Ños, fuerte pelete!”.
- En el caso del microondas, debe encender el electrodoméstico y mostrar por pantalla un mensaje “Algo huele a quemado”.

c) Sobrecarga el método toString de ambas clases de manera que generen una salida idéntica a las siguientes:

Nevera	---------- * Tipo de electrodoméstico: Nevera. * Marca: <Mostrar el valor del atributo correspondiente> * Modelo: <Mostrar el valor del atributo correspondiente> * Peso: <Mostrar el valor del peso con sus unidades> * Potencia: <Mostrar el valor de la potencia con sus unidades> * Temperatura: <Debe mostrarse aquí el valor correspondiente> * Está encendido: <Debe mostrarse aquí “Si” o “No”> ----------

Microondas	----------
	* Tipo de electrodoméstico: Microondas.
	* Marca: <Mostrar el valor del atributo correspondiente>
	* Modelo: <Mostrar el valor del atributo correspondiente>
	* Peso: <Mostrar el valor del peso con sus unidades>
	* Potencia: <Mostrar el valor de la potencia con sus unidades>
	* Puerta: <Debe mostrarse aquí “Abierta” o “Cerrada”>
	* Está encendido: <Debe mostrarse aquí “Si” o “No”>

Por último, en la clase ejecutable, instancia un objeto de cada tipo, enciéndelos y muéstralos por pantalla empleando el comando sout. Finalmente, apaga los objetos de nuevo.

Rúbrica de evaluación:

[G] Indentación del código: **0,5 puntos**

[G] Nombre de variables, métodos y clases: **0,5 puntos**

[G] Estructura de paquetes y nomenclatura: **0,5 puntos**

[G] Clase ejecutable: **1 punto**

[1] Métodos estáticos: **3 puntos**

[2] Herencia de clases: **2 puntos**

[2] Atributos propios no heredados: **0,5 puntos**

[2] Creación y herencia de constructores: **0,5 puntos**

[2] Control de la visibilidad de los miembros de las clases: **0,5 puntos**

[2] Sobrecarga de métodos: **0,5 puntos**

[2] Sobrecarga de toString: **0,5 punto**