Ejercicios prácticos

Unidades: 7

Guía

- Se plantean diferentes ejercicios para realizarlos en java para trabajar con herencia, interfaces y clases abstractas.
- Se recomienda utilizar un IDE como Eclipse o Netbeans
- Puede haber más de una solución en un ejercicio, buscar la solución óptima, aquella que requiera menos líneas de código y no se abuse de estructuras de control
- Para facilitar la impresión textual de un objeto implementa el método toString() en cada clase.

El método toString() en Java es una función incorporada en la clase Object. Esta función se usa para crear una representación en cadena de texto (String) de un objeto.

Cuando intentas imprimir un objeto directamente, por ejemplo, usando System.out.println(obj), Java automáticamente invoca el método toString() para obtener una representación del objeto en formato String.

Una forma de implementarlo es utilizar esta estructura:

```
@Override
public String toString() {
        return "NOMBRE DE LA CLASE[" + "atributo1='" + atributo1 + ", atributo2=" + atributo2 + ",
                atributoN=" + atributoN + "]";
}
```

Ejemplo

Elabora un programa para gestionar aspectos médicos de órganos de pacientes. Por ahora solo se van a disponer información de dos órganos, el corazón y el pulmón. Como no sabemos si más adelante se debe disponer de más órganos, se va a desarrollar una superclase Órgano con el id del paciente y el estado en que se encuentra.

El corazón debe poder guardar la frecuencia cardíaca y tener un método para comprobar si la frecuencia se encuentra en una situación normal o anormal. Se considera normal si está comprendida entre 60-100.

El pulmón tiene como atributo la capacidad pulmonar.

Todas las clases deben tener un método denominado *mostrarDetalles* para imprimir por pantalla los datos de sus atributos.

```java
public class Organo {
	private int idPaciente;
	private String estado;

	public Organo(int idPaciente, String estado) {
		this.idPaciente = idPaciente;
		this.estado = estado;
	}

	public void mostrarDetalles() {
		System.out.println("ID Paciente: " + idPaciente + " Estado: " +
			estado);
	}
}

public class Corazon extends Organo {
	private int frecuenciaCardiaca;

	public Corazon(int idPaciente, String estado, int heartRate) {
		super(idPaciente, estado);
		this.frecuenciaCardiaca = heartRate;
	}

	@Override
	public void mostrarDetalles() {
		super.mostrarDetalles(); //
			System.out.println("Frecuencia cardíaca del corazón: " +
			frecuenciaCardiaca);
	}

	public void comprobarFrecuenciaCardiaca() {
		if (frecuenciaCardiaca >= 60 && frecuenciaCardiaca <= 100) {
			System.out.println("La frecuencia cardíaca es normal");
		} else {
			System.out.println("La frecuencia cardíaca es anormal.");
		}
	}
}

public class Pulmon extends Organo {
	private float capacidad;

	public Pulmon(int idPaciente, String estado, float capacidadPulmonar) {
		super(idPaciente, estado);
		this.capacidad = capacidadPulmonar;
	}

	@Override
	public void mostrarDetalles() {
		super.mostrarDetalles();
		System.out.println("Capacidad pulmonar: " + capacidad);
	}
}
```

Ejercicio 1.

Desarrolla una aplicación que tenga una representación de un sistema de gestión de inventario para un almacén de componentes informáticos.

Deberás crear una clase base llamada "Componente" que tiene tres atributos: referencia (tipo long), marca (tipo String) y modelo (tipo String). Deberás incluir en esta clase un constructor que inicialice estos tres atributos, métodos de acceso (getters y setters) para cada uno de ellos, y el método toString().

Además, se pide que desarrolles dos clases que heredan de la clase "Componente", estas clases son "DiscoDuro" y "Microprocesador".

La clase "DiscoDuro" debe tener un atributo adicional: capacidad (tipo int). Deberás incluir un constructor en esta clase que inicialice los cuatro atributos (los tres de la clase "Componente" y el nuevo atributo), métodos de acceso para este nuevo atributo y el método toString().

La clase "Microprocesador" debe tener un atributo adicional: frecuencia (tipo int). Deberás incluir un constructor en esta clase que inicialice los cuatro atributos (los tres de la clase "Componente" y el nuevo atributo), métodos de acceso para este nuevo atributo, y el método toString().

Finalmente, crea una clase principal denominada “Ej1” que contenga un método main, crea instancias de estas clases, invoca sus métodos y muestra los resultados.

Ejercicio 2.

Realiza una aplicación para gestionar vehículos que permita saber si están arrancado o no. Para ello crea una clase “Vehículo” que tenga los atributos marca, modelo, precio y estaArrancado. Esta clase debe tener:

- Un constructor que tome tres argumentos para inicializar estos valores y estaArrancado como false.
- Un método arrancar() y parar() que cambie el estado de estaArrancado y que imprima si el vehículo está arrancado o apagado. Estos métodos deben ser capaz de reconocer que clase hija le llama para mostrar un mensaje diferente, según el tipo de vehículo. Por ejemplo, “El [coche/moto/camión/etc..] está arrancando o parado”
- El método toString().

Crea dos subclases: Coche y Moto. El coche solo tiene un sistema de arranque eléctrico, en cambio, la moto puede ser arrancada manualmente o de forma eléctrica, la lógica de la moto debe ser capaz de distinguir estos dos tipos de arranque.

Ambas clases deben tener:

- Un constructor que tome los mismos atributos que tiene la clase Vehículo.

- Métodos arrancar() según los sistemas de arranque que tengan. Deben llamar al método arrancar() de Vehículo y mostrar un mensaje inicial con el mensaje "(El coche/La moto) ha utilizado el encendido (manual/eléctrico)"
- Un método parar(). Debe llamar al método parar() de Vehículo.
- El método toString().

Finalmente, crea una clase principal denominada "Ej2" que contenga un método main, crea instancias de estas clases, invoca sus métodos y muestra los resultados.

Ejercicio 3.

Desarrolla una aplicación que permite calcular el área y perímetro de diferentes figuras geométricas para ello se pide crear una interfaz llamada "FiguraGeometrica" que declare los siguientes métodos: "calcularArea()" y "calcularPerimetro()".

Crea tres clases que implementen esta interfaz: "Circulo", "Rectangulo" y "Triangulo". Cada clase debe tener los atributos necesarios para poder calcular el área y el perímetro (por ejemplo, un círculo necesita el radio, un rectángulo necesita largo y ancho, y un triángulo necesita base y altura para el área, y los tres lados para el perímetro).

Implementa los métodos calcularArea() y calcularPerimetro() en cada clase de forma acorde a la figura que representan.

Crea una clase denominada "Ej3" que tenga la función main, crea instancias de estas clases, invoca sus métodos y muestra los resultados.

Ejercicio 4.

Desarrolla una aplicación que permita gestionar instrumentos musicales. Crear una interfaz llamada "InstrumentoMusical" que declare los siguientes métodos: "tocar()", "afinar()" y "tipoInstrumento()".

Crea tres clases que implementen esta interfaz: "Guitarra", "Piano" y "Trompeta". Cada una de estas clases debe implementar los métodos de la interfaz de la siguiente manera:

- tocar(): debe imprimir un mensaje que diga "Estoy tocando [instrumento]".
- afinar(): debe imprimir un mensaje que diga "Estoy afinando [instrumento]".
- tipoInstrumento(): debe retornar una cadena de texto que describa el tipo de instrumento (puedes ser creativo aquí, o simplemente retornar el nombre del instrumento).

Crea una clase denominada "Ej4" con la función main, crea instancias de estas clases, invoca sus métodos y muestra los resultados.

Ejercicio 5.

Desarrolla una aplicación para representar un sistema de archivos básico. En este sistema de archivos, hay dos tipos de objetos con los que puedes interactuar: archivos y directorios.

Cada archivo tiene un nombre y un tamaño en bytes. Cada directorio tiene un nombre y puede contener una cantidad arbitraria de archivos y directorios. Además, cada directorio tiene un tamaño, que se define como la suma del tamaño de todos los archivos y directorios contenidos dentro de él.

Se te pide diseñar este sistema y proporcionar una manera de calcular el tamaño total de un directorio dado. Ten en cuenta que los archivos pueden ser añadidos a un directorio y cada objeto dentro del sistema de archivos tiene un nombre.

Crea una clase denominada "Ej5" con la función main, crea instancias de estas clases, invoca sus métodos y muestra los resultados.

Ejercicio 6.

Desarrolla una aplicación de un sistema de gestión de productos para una tienda que vende varios tipos de productos: electrónicos, alimentos y ropa. Todos los productos tienen algunas características en común: un código de identificación, un nombre y un precio. Sin embargo, cada tipo de producto tiene diferentes maneras de calcular su precio final para el cliente.

Para los productos electrónicos, se agrega una tarifa de reciclaje al precio; para los alimentos, se agrega un impuesto sobre las ventas; y para la ropa, si es de una determinada marca, se añade un precio adicional.

Crea una clase abstracta Producto con las características comunes de todos los productos y un método abstracto calcularPrecioFinal. Luego, crea las clases Electronico, Alimento y Ropa que extienden a Producto y proporciona una implementación para el método calcularPrecioFinal en cada una de estas subclases.

Finalmente, escribe un programa de prueba que cree al menos un objeto de cada subclase, calcule el precio final para cada producto e imprima la información del producto junto con su precio final.

Ejercicio 7.

Desarrolla una aplicación de un sistema para un banco que ofrece varios tipos de cuentas: cuenta de ahorro, cuenta corriente y cuenta de inversión. Todas las cuentas tienen un número de cuenta, un titular y un saldo. Sin embargo, cada tipo de cuenta tiene diferentes maneras de calcular el interés.

Crea una clase abstracta Cuenta con las características comunes de todas las cuentas y un método abstracto calcularInteres. Luego, crea las clases CuentaAhorro, CuentaCorriente y CuentaInversion que extienden a Cuenta y proporciona una implementación para el método calcularInteres en cada una de estas subclases.

Finalmente, escribe un programa de prueba que cree al menos un objeto de cada subclase, calcule el interés para cada cuenta e imprima la información de la cuenta junto con el interés calculado.

Ejercicio 8.

Vamos a diseñar una simulación de un jardín. En este jardín, hay una variedad de organismos, incluyendo varias especies de plantas, insectos y aves. Por otro lado, podemos añadir distintas casas o casetas para que puedan vivir los organismos.

Una particularidad del jardín es que cualquier objeto que exista, sea un organismo o casa, puede tener la capacidad de moverse por sí mismo o que se pueda ser trasladado por acción nuestra. La forma de mover o trasladar será a partir de recibir dos coordenadas, x e y que representa una ubicación del jardín. Con lo cual, debe de existir dos métodos denominados, mover(ubicación) y trasladar(ubicación) comunes a todas nuestras clases, y que permita modificar su funcionalidad según la necesidad de cada una de ellas. Cabe destacar que, por las necesidades de nuestra aplicación, pensamos que la funcionalidad de la movilidad se pueda extender a futuros requerimientos que no sea solo del jardín.

Cada organismo tiene un conjunto de características comunes: un nombre y una ubicación en el jardín, como se ha indicado en el anterior párrafo, las ubicaciones se representan con coordenadas, x e y. Según los requisitos, no es posible tener organismos genéricos, solo podemos tener plantas, insectos, aves u otro tipo que se presente en un futuro.

Dentro de los organismos tenemos las plantas que tienen como característica propia la fecha en la que se plantó y las acciones de crecer y realizar la fotosíntesis. A diferencia de los demás organismos, las plantas no pueden moverse por sí mismo, solo se pueden trasladar por acción nuestra.

Otro de los organismos, los insectos no tienen características propias. Pueden moverse libremente por el jardín según la ubicación que quieran y tienen la capacidad de alimentarse de plantas y de otros insectos, pero no del resto de organismos. Aún que los insectos son libres, los podemos atrapar y trasladarlos a otra ubicación por nuestra cuenta.

Las aves son otro organismo que no tienen características propias. Pueden moverse libremente por el jardín según la ubicación que quieran, aún que esta acción no existe como tal, si no, que su acción se denomina volar y no es posible trasladar las aves a otra ubicación.

Hay dos estructuras construidas en el jardín: una caseta para pájaros y una casa de insectos. Ambas estructuras comparten características, como el nombre, el material con el que están construidas y la ubicación. En cambio, se diferencia en que la caseta de pájaros permite saber la cantidad aves que puede tener y la casa de insectos permite

conocer el tipo de insectos que alberga. Cabe indicar que nuestra aplicación si permite tener casas genéricas que no sean de pájaros ni de insectos. Aún que las casas no tienen ninguna acción peculiar, todas se comportan igual, realizando acciones de mover y trasladar:

- Cuando se mueva, se mostrará un mensaje indicando: "La casa no puede moverse por sí sola."
- Cuando se traslade, se mostrará un mensaje indicando: "La casa se traslada a la ubicación x,y".

Además, debes implementar un método denominado muestraCaracteristicas() en cada clase que imprima un mensaje mostrando los atributos y los valores que tiene de cada organismo o casa. En el caso, que en un futuro se generen nuevas clases que representen objetos del jardín, tenemos que forzar que se implemente este método con el mismo nombre.

Tu tarea es diseñar las clases necesarias para representar a los diferentes organismos y estructuras del jardín, pensando si es necesario crear clases abstractas o interfaces, así como, definir las acciones que pueden realizar. No es necesario realizar la funcionalidad de las acciones, solo es suficiente con mostrar un mensaje. Por ejemplo, la acción de mover puede mostrar un mensaje como el siguiente: "El objeto se mueve a la ubicación x e y". Debes asegurarte de que tu diseño es lo suficientemente flexible como para poder añadir nuevas especies de organismos y nuevas acciones en el futuro.

Por último, debes implementar un método en tu programa principal que utilice tus clases para crear una pequeña simulación del jardín. Este método debe crear una variedad de organismos y casas, y luego debe utilizar los métodos de tus clases para hacer que los organismos realicen varias acciones