Resolución Práctico 7: Herencia y polimorfismo.

Ejercicio 1: En este ejercicio debemos crear una clase padre llamada Vehiculo con los atributos marca y modelo, y el método mostrarInfo(), luego debemos crear una clase Auto que herede de esta clase vehículo y además crearle un atributo adicional cantidadDePuertas, también debemos sobreescribir el método mostrarInfo de la clase padre. Por ultimo debemos instanciar un auto en el main y mostrar su información completa.

Codigo.

Clase Vehiculo:

```
public class Vehiculo {
    // Declaramos los atributos de la clase padre protegidos
    protected String marca;
    protected String modelo;

    // Creamos su constructor
    public Vehiculo(String marca, String modelo) {
        this.marca = marca;
        this.modelo = modelo;
    }

    // Metodo para mostrar la informacionde un vehivulo, el cual vamos a sobreescribir luego
    public void mostrarInfo(){
        System.out.println("Modelo: " + modelo + " ,marca: " + marca);
    }
}
```

Clase Auto:

```
public class Auto extends Vehiculo { // Extendemos de la clase vehiculo
    // Declaramos el metodo adicional de clase auto
    private int cantidadDePuertas;

    // Creamos el constructor llamando tambien al constructor de la clase padre
    public Auto(int cantidadDePuertas, String marca, String modelo) {
        super(marca, modelo);
        this.cantidadDePuertas = cantidadDePuertas;
    }

    // Sobreescribimos al metodo mostrarInfo
    @Override
    public void mostrarInfo(){
        System.out.println("Modelo: " + this.modelo + " ,marca: " + this.marca + ", cantidad de puertas: " + cantidadDePuertas);
    }

}
```

Main:

```
public static void main(String[] args) {
    // Instanciamos un auto
    Auto a = new Auto(5, "Renault", "Sandero");

    // Llamamos al metodo para mostrar su informacion
    a.mostrarInfo();
}
```

Salida por pantalla:

```
Modelo: Sandero ,marca: Renault, cantidad de puertas: 5
------------------------------------------------------------------------
BUILD SUCCESS
```

Ejercicio 2:
En este ejercicio debemos crear una clase abstracta llamada Figura con un atributo nombre y un método calcularArea(). Luego debemos crear dos clases que hereden de Figura: una clase Circulo y una clase Rectangulo, cada una implementando su propia versión del método calcularArea() de acuerdo a su fórmula correspondiente.
Por último, debemos crear un array de figuras, instanciar al menos un círculo y un rectángulo, y mostrar el área de cada figura usando polimorfismo.

Codigo.
Clase Figura:

```java
public abstract class Figura {
    // Declaramos el atributo protegido nombre
    protected String nombre;

    // Creamos el constructor de figura
    public Figura(String nombre) {
        this.nombre = nombre;
    }

    // Creamos el  metodo calcular area
    public void calcularArea(){
    }
}
```

Clase Circulo:

```java
public class Circulo extends Figura{//Extendemos de figura
    // Declaramos el atributo radio
    private double radio;

    // Creamos el contructor
    public Circulo(double radio, String nombre) {
        super(nombre);
        this.radio = radio;
    }

    // Sobreescribimos al metodo caluclar area
    @Override
    public void calcularArea(){
        System.out.println("El area del cirulo " + nombre + " es: " +  (radio *  3.14));
    }
}
```

Clase rectángulo:

```java
public class Rectangulo extends Figura{ // Extendemos de figura
    // Creamos los atributos base y altura
    private double base;
    private double altura;

    // Creamos el constructos
    public Rectangulo(double base, double altura, String nombre) {
        super(nombre);
        this.base = base;
        this.altura = altura;
    }

    // Sobreescribimos al metodo calcular area
    @Override
    public void calcularArea(){
        System.out.println("El area del rectangulo " + nombre + " es: " +  (base * altura));
    }
}
```

Main:

```java
public class Main {

    public static void main(String[] args) {
        // Creamos un array vacio de figuras
        ArrayList<Figura> figuras = new ArrayList<>();

        // Creamos y añadimos 4 figuras al array
        Rectangulo r1 = new Rectangulo(4.0,  4.0, "Rectangulo 1");
        Rectangulo r2 = new Rectangulo(6.0,  4.0, "Rectangulo 2");
        Circulo c1 = new Circulo(10, "Circulo 1");
        Circulo c2 = new Circulo(15, "Circulo 2");

        figuras.add(r1);
        figuras.add(r2);
        figuras.add(c1);
        figuras.add(c2);

        // Recorremos el array y llamamos al metodo calcular area en cada figura
        for(Figura f : figuras){
            f.calcularArea();
        }

    }

}
```

Salida por pantalla:

```
El area del rectangulo Rectangulo 1 es: 16.0
El area del rectangulo Rectangulo 2 es: 24.0
El area del cirulo Circulo 1 es: 31.400000000000002
El area del cirulo Circulo 2 es: 47.1
```

Ejercicio 3:
En este ejercicio debemos crear una clase Empleado con un método calcularSueldo(Empleado e) que reciba un empleado y devuelva un sueldo fijo según el tipo de empleado. También debemos crear dos clases que hereden de Empleado: EmpleadoPlanta y EmpleadoTemporal. Luego, debemos crear una lista de empleados, agregar instancias de ambas subclases, recorrer la lista y para cada empleado invocar el método calcularSueldo() para obtener su sueldo, usando instanceof dentro del método para determinar el tipo de empleado y asignar el sueldo correspondiente.

Codigo.
Empleado:

```
public abstract class Empleado {

    /**
     * Calcula el sueldo correspondiente a un empleado según su tipo.
     *
     * @param e el empleado del cual se desea calcular el sueldo. Puede ser una
     * instancia de EmpleadoPlanta o EmpleadoTemporal.
     * @return el sueldo del empleado: - 900000.0 si es EmpleadoPlanta -
     * 850000.0 si es EmpleadoTemporal - 0.0 si no pertenece a ninguno de los
     * tipos anteriores
     */
    public double calcularSueldo(Empleado e) {
        if (e instanceof EmpleadoPlanta) {
            return 900000.0;
        } else if (e instanceof EmpleadoTemporal) {
            return 850000.0;
        } else {
            return 0;
        }
    }
}
```

EmpleadoPlanta:

```
public class EmpleadoPlanta extends Empleado{ // Extendemos de empleado

}
```

EmpleadoTemporal:

```
public class EmpleadoTemporal extends Empleado{ //Extendemos de empleado

}
```

Main:

```
public class Main {

    public static void main(String[] args) {
        // Inicializamos un array de empleados
        ArrayList<Empleado> empleados = new ArrayList<>();

        // Creamos empleados y los agregamos al array
        EmpleadoPlanta e1 = new EmpleadoPlanta();
        EmpleadoPlanta e2 = new EmpleadoPlanta();
        EmpleadoTemporal e3 = new EmpleadoTemporal();
        EmpleadoTemporal e4 = new EmpleadoTemporal();

        empleados.add(e1);
        empleados.add(e2);
        empleados.add(e3);
        empleados.add(e4);

        int i = 0; // Variable solo para ver el indice en el for, OPCIONAL.
        // Recorremos el array de empleados y llamamos al  metodo para calcular sueldo
        for(Empleado e : empleados) {
            System.out.println("El empleado " + i + " cobra: " + e.calcularSueldo(e));
            i++; // Incrementamos el indice
        }
    }

}
```

Salida por pantalla:

```
El empleado 0 cobra: 900000.0
El empleado 1 cobra: 900000.0
El empleado 2 cobra: 850000.0
El empleado 3 cobra: 850000.0
----------------------------------------
BUILD SUCCESS
```

Ejercicio 4:

En este ejercicio debemos crear una clase llamada Animal con los métodos hacerSonido() y describirAnimal(). Luego, debemos crear tres clases que hereden de Animal: Perro, Gato y Vaca, y sobrescribir el método hacerSonido() en cada una con la anotación @Override para que cada animal emita su sonido característico.

Por último, debemos crear una lista de animales, agregar instancias de Perro, Gato y Vaca, y mostrar los sonidos de cada uno utilizando polimorfismo.

Codigo.

Clase animal:

```
public class Animal {
    // Creamos el metodo hacer sonido
    public void hacerSonido(){

    }

    public void describirAnaimal(){

    }
}
```

Clase Perro:

```
public class Perro extends Animal{ // Extendemos de animal
    // Sobreescribimos hacer sonido
    @Override
    public void hacerSonido(){
        System.out.println("Guaf!!");
    }
}
```

Clase Gato:

```
public class Gato extends Animal{ // Extendemos de animal
    // Sobreescribimos hacer sonido
    @Override
    public void hacerSonido(){
        System.out.println("Miau!!");
    }
}
```

Clase Vaca:

```
public class Vaca extends Animal{ // extendemos de animal
    // Sobreescribimos hacer sonido
    @Override
    public void hacerSonido(){
        System.out.println("MUUU!!");
    }
}
```

Main:

```
public class Main {

    public static void main(String[] args) {
        // Inicializamos array vacio de animales
        ArrayList<Animal> animales = new ArrayList<>();

        // Creamos y agregamos animales al array
        Perro p1 = new Perro();
        Gato g1 = new Gato();
        Vaca v1 = new Vaca();

        animales.add(p1);
        animales.add(g1);
        animales.add(v1);

        // Recorremmos el array y llamamos al metodo hacer sonido
        for (Animal a : animales) {
            a.hacerSonido();
        }

    }

}
```

Salida por consola:

```
Guaf!!
Miau!!
MUUU!!
---------------------------------------
BUILD SUCCESS
```

Ejercicio 5:

En este ejercicio debemos crear una interfaz llamada Pagable con un método pagar(). Luego, debemos crear tres clases que implementen esta interfaz: TarjetaCredito, Transferencia y Efectivo, definiendo en cada una su propia versión del método pagar().

Además, debemos crear un método llamado procesarPago(Pagable medio), que reciba cualquier forma de pago e invoque su método pagar(), permitiendo así procesar distintos medios de pago de forma genérica.

Por último, debemos crear objetos de las distintas clases de pago y procesarlos usando una única función.

Codigo.

Interfaz pagable:

```
public interface Pagable {
    // Declaramos el metodo pagar
    void pagar();
}
```

Clase TarjetaCredito:

```
public class TarjetaCredito implements Pagable{ // Implementamos pagable

    // Sobreescribimos el metodo pagar
    @Override
    public void pagar() {
        System.out.println("Pago realizado con tarjeta de credito");
    }

}
```

Clase Transferencia:

```
public class Transferencia implements Pagable{ // Implementamos pagable

    // Sobreescribimos el metodo pagar
    @Override
    public void pagar() {
        System.out.println("Pago realizado con transferencia");
    }

}
```

Clase Efectivo:

```
public class Efectivo implements Pagable{ // Implementamos pagable

    // Sobreescribimos el metodo pagar
    @Override
    public void pagar() {
        System.out.println("Pago realizado con efectivo");
    }

}
```

Main:

```java
public class Main {

    public static void main(String[] args) {
        // Inicializamos un arrayList del tipo Pagable
        ArrayList<Pagable> formasDePago = new ArrayList<>();

        // Agreamos 3 formas de pago al array
        formasDePago.add(new TarjetaCredito());
        formasDePago.add(new Transferencia());
        formasDePago.add(new Efectivo());

        // Recorremos el array y llamamos al metodo procesarPago
        for (Pagable p : formasDePago) {
            procesarPago(p);
        }
    }

    /**
     * Procesa un pago utilizando cualquier objeto que implemente la interfaz
     * Pagable.
     *
     * @param medio el medio de pago a procesar. Puede ser una instancia de
     * TarjetaCredito, Transferencia, Efectivo, u otra clase que implemente
     * Pagable.
     */
    public static void procesarPago(Pagable medio) {
        medio.pagar();
    }
}
```

Salida por consola:

```
Pago realizado con tarjeta de credito
Pago realizado con transferencia
Pago realizado con efectivo
------------------------------------------------------------
BUILD SUCCESS
------------------------------------------------------------
```