Trabajo Práctico N º 7

“Herencia y polimorfismo”

Alumnos: Alemis, Juan Cruz – juancruzalemis@gmail.com

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.

Vechiculo

```java
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;
    }

    // Método para mostrar la información de un Vehiculo
    public void mostrarInfo() {
        System.out.println("Modelo: " + modelo + ", Marca: " + marca);
    }
}
```

Auto

```java
public class Auto extends Vehiculo {

    // Atributo adicional
    private int cantidadDePuertas;

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

    // Sobrescritura del método mostrarInfo
    @Override
    public void mostrarInfo() {
        System.out.println("Modelo: " + this.modelo +
                           ", Marca: " + this.marca +
                           ", Cantidad de puertas: " + cantidadDePuertas);
    }
}
```

Main

```java
public class Main {
    public static void main(String[] args) {

        // Instanciamos un auto
        Auto a = new Auto(3, "Merdecez Benz", "Sprinter");

        // Llamamos al método para mostrar su información
        a.mostrarInfo();
    }
}
```

Pantalla

```
Modelo: Sprinter, Marca: Merdecez Benz, Cantidad de puertas: 3
BUILD SUCCESSFUL (total time: 0 seconds)
```

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.

Figura

```
public abstract class Figura {

    protected String nombre;

    public Figura(String nombre) {
        this.nombre = nombre;
    }

    public abstract double calcularArea();

    public void mostrarArea() {
        System.out.println("El área del " + nombre + " es: " + calcularArea());
    }
}
```

Ciculo

```
public class Circulo extends Figura {

    private double radio;

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

    @Override
    public double calcularArea() {
        return Math.PI * Math.pow(radio, 2);
    }
}
```

Rectangulo

```java
public class Rectangulo extends Figura {

    private double alto;
    private double ancho;

    public Rectangulo(double alto, double ancho, String nombre) {
        super(nombre);
        this.alto = alto;
        this.ancho = ancho;
    }

    @Override
    public double calcularArea() {
        return alto * ancho;
    }
}
```

Main

```java
public class Main {

    public static void main(String[] args) {

        ArrayList<Figura> figuras = new ArrayList<>();

        figuras.add(new Rectangulo(10, 4, "Rectangulo 1"));
        figuras.add(new Rectangulo(2, 6, "Rectangulo 2"));
        figuras.add(new Circulo(10, "Circulo 1"));
        figuras.add(new Circulo(3, "Circulo 2"));

        for (Figura f : figuras) {
            f.mostrarArea();
        }
    }
}
```

Pantalla

```
El  rea del Rectangulo 1 es: 40.0
El  rea del Rectangulo 2 es: 12.0
El  rea del Circulo 1 es: 314.1592653589793
El  rea del Circulo 2 es: 28.274333882308138
BUILD SUCCESSFUL (total time: 0 seconds)
```

Ejercicio3:

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.

Empleado

```
public class Empleado {

    public double calcularSueldo(Empleado e) {
        if (e instanceof EmpleadoPlanta) {
            return 900000;
        } else if (e instanceof EmpleadoTemporal) {
            return 500000;
        }
        return 0;
    }
}
```

EmpleadoPlanta

```
public class EmpleadoPlanta extends Empleado {
}
```

EmpleadoTemporal

```
public class EmpleadoTemporal extends Empleado {
}
```

Main

```
    public static void main(String[] args) {

        // Inicializamos un array de empleados
        ArrayList<Empleado> empleados = new ArrayList<>();

        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;


        for (Empleado e : empleados) {
            System.out.println("El empleado " + i + " cobra: " + e.calcularSueldo(e));
            i++;
        }
    }
}
```

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 SUCCESSFUL (total time: 0 seconds)
```

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.

Animal

```java
public class Animal {

    public void hacerSonido() {
        System.out.println("Sonido genérico...");
    }

    public void describirAnimal() {
        System.out.println("Soy un animal");
    }
}
```

Perro

```java
public class Perro extends Animal {

    @Override
    public void hacerSonido() {
        System.out.println("Guau!!");
    }
}
```

Gato

```java
public class Gato extends Animal {

    @Override
    public void hacerSonido() {
        System.out.println("Miau!!");
    }
}
```

Vaca

```java
public class Vaca extends Animal {

    @Override
    public void hacerSonido() {
        System.out.println("Muuu!!");
    }
}
```

Main

```java
public class Main {

    public static void main(String[] args) {

        ArrayList<Animal> animales = new ArrayList<>();

        animales.add(new Perro());
        animales.add(new Gato());
        animales.add(new Vaca());

        for (Animal a : animales) {
            a.hacerSonido();
        }
    }
}
```

Pantalla

```
Guau!!
Miau!!
Muuu!!
BUILD SUCCESSFUL (total time: 0 seconds)
```

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.

Pagable

```java
public interface Pagable {
    void pagar();
}
```

TarjetaCredito

```java
public class TarjetaCredito implements Pagable {

    @Override
    public void pagar() {
        System.out.println("Pago realizado con tarjeta de crédito");
    }
}
```

Transferencia

```java
public class Transferencia implements Pagable {

    @Override
    public void pagar() {
        System.out.println("Pago realizado con transferencia");
    }
}
```

Efectivo

```java
public class Efectivo implements Pagable {

    @Override
    public void pagar() {
        System.out.println("Pago realizado con efectivo");
    }
}
```

Main

```java
public class Main {

    public static void main(String[] args) {

        ArrayList<Pagable> pagos = new ArrayList<>();

        pagos.add(new TarjetaCredito());
        pagos.add(new Transferencia());
        pagos.add(new Efectivo());

        for (Pagable medio : pagos) {
            procesoPago(medio);
        }
    }

    public static void procesoPago(Pagable medio) {
        medio.pagar();
    }
}
```

Pantalla

```
Pago realizado con tarjeta de cr�dito
Pago realizado con transferencia
Pago realizado con efectivo
BUILD SUCCESSFUL (total time: 0 seconds)
```