UNIVERSIDAD TECNOLÓGICA DEL PERÚ

Facultad de Ingeniería

Programacion Orientada a Objetos

Sesión 7: Implementación de programas usando polimorfismo

Recordando:

¿Que vimos la clase pasada?

Logro de la sesión

Al finalizar la sesión, el estudiante aplica el polimorfismo en la solución de problemas e implementa soluciones que utilicen eficientemente la programación orientada a objetos en java.

Utilidad

Implementar sistemas utilizando polimorfismo no solo hace que el código sea más flexible y reutilizable, sino que también promueve buenas prácticas de diseño de software, como la abstracción, la encapsulación y la aplicación de patrones de diseño

Agenda

- Implementación de programas usando polimorfismo.

Sistema de Empleados

```
// Clase base abstracta Empleado
abstract class Empleado {
   protected String nombre;

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

   public abstract double salario();
}

// Clase derivada EmpleadoFijo
class EmpleadoFijo extends Empleado {
   private double salarioFijo;

   public EmpleadoFijo(String nombre, double salarioFijo) {
      super(nombre);
      this.salarioFijo = salarioFijo;
   }

   @Override
   public double salario() {
      return salarioFijo;
   }
}
```

```
// Clase derivada EmpleadoPorHora
class EmpleadoPorHora extends Empleado {
   private double tarifaPorHora;
   private int horas;

   public EmpleadoPorHora(String nombre, double tarifaPorHora, int horas) {
      super(nombre);
      this.tarifaPorHora = tarifaPorHora;
      this.horas = horas;
   }

   @Override
   public double salario() {
      return tarifaPorHora * horas;
   }
}

public class TestEmpleados {
   public static void main(String[] args) {
      Empleado e1 = new EmpleadoFijo("Juan", 1500);
      Empleado e2 = new EmpleadoPorHora("Ana", 20, 80);

      System.out.println(e1.nombre + " tiene un salario de: " + e1.salario());
      System.out.println(e2.nombre + " tiene un salario de: " + e2.salario());
   }
}
```

Explicación:

Aquí, la clase abstracta Empleado tiene un método abstracto salario(). Tenemos dos tipos de empleados: EmpleadoFijo, que tiene un salario fijo, y EmpleadoPorHora, que se le paga según las horas trabajadas. Ambas clases derivadas implementan el método salario() según su propia lógica.

Sistema de Pagos

```
// Clase base abstracta Pago
abstract class Pago {
   protected double monto;

   public Pago(double monto) {
      this.monto = monto;
   }

   public abstract void realizarPago();
}

// Clase derivada PagoTarjeta
class PagoTarjeta extends Pago {
   private String numeroTarjeta;

   public PagoTarjeta(double monto, String numeroTarjeta) {
      super(monto);
      this.numeroTarjeta = numeroTarjeta;
   }

   @Override
   public void realizarPago() {
      System.out.println("Realizando pago de " + monto + " con tarjeta " + numeroTarjeta);
   }
}
```

```
// Clase derivada PagoPaypal
class PagoPaypal extends Pago {
   private String email;

   public PagoPaypal(double monto, String email) {
      super(monto);
      this.email = email;
   }

   @Override
   public void realizarPago() {
      System.out.println("Realizando pago de " + monto + " con PayPal al email " + email);
   }
}

public class TestPagos {
   public static void main(String[] args) {
      Pago pago1 = new PagoTarjeta(100.50, "1234-5678-9012-3456");
      Pago pago2 = new PagoPaypal(50.25, "usuario@example.com");

      pago1.realizarPago();
      pago2.realizarPago();
   }
}
```

Explicación:

En este programa, la clase abstracta Pago tiene un método abstracto realizarPago(). Las clases derivadas PagoTarjeta y PagoPaypal implementan este método para mostrar cómo se realiza el pago según el método seleccionado.

Ambos ejemplos demuestran cómo el polimorfismo permite tratar objetos de clases derivadas como objetos de la clase base y cómo las implementaciones específicas de los métodos en las clases derivadas se invocan en tiempo de ejecución.

Practica Guiada

Implementa los ejercicios anteriormente realizados en el programa java.

UTP Universidad Tecnológica del Perú

Conclusiones

- Al utilizar polimorfismo, puedes reutilizar el mismo código para operar sobre diferentes tipos de objetos. Esto reduce la duplicación de código y facilita el mantenimiento del sistema, ya que los cambios solo necesitan hacerse en un lugar.

Cierre

¿Que hemos aprendido hoy?

Universidad Tecnológica del Perú

Bibliografía

- MORENO PÉREZ, J. Programación orientada a objetos.RA-MA Editorial. https://tubiblioteca.utp.edu.pe/cgi-bin/koha/opac-detail.pl?biblionumber=31933
- Vélez Serrano, José. Diseñar y programar, todo es empezar: una introducción a la Programación Orientada a Objetos usando UML y Java. Dykinson. https://tubiblioteca.utp.edu.pe/cgi-bin/koha/opac-detail.pl?biblionumber=36368

Universidad Tecnológica del Perú