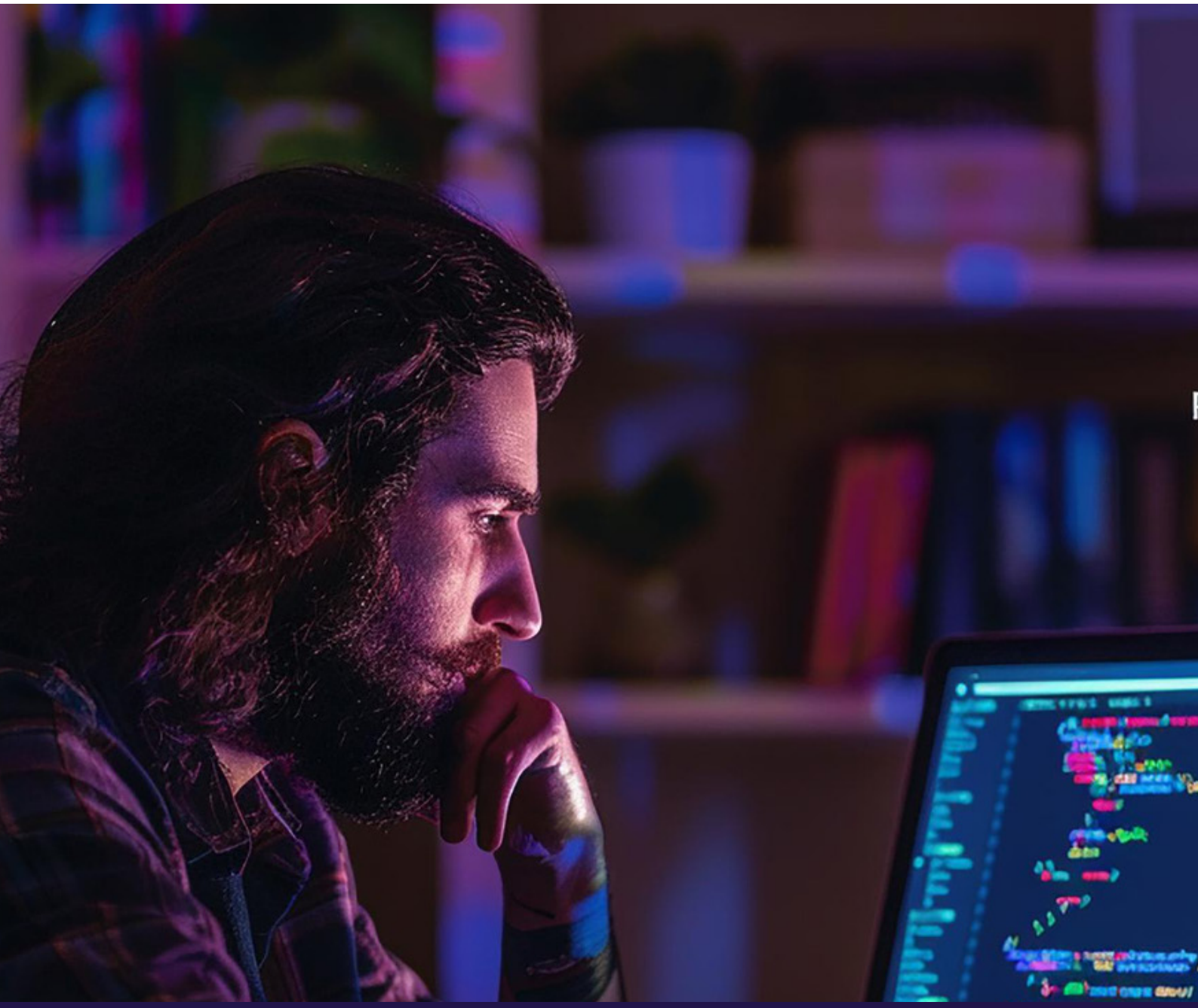
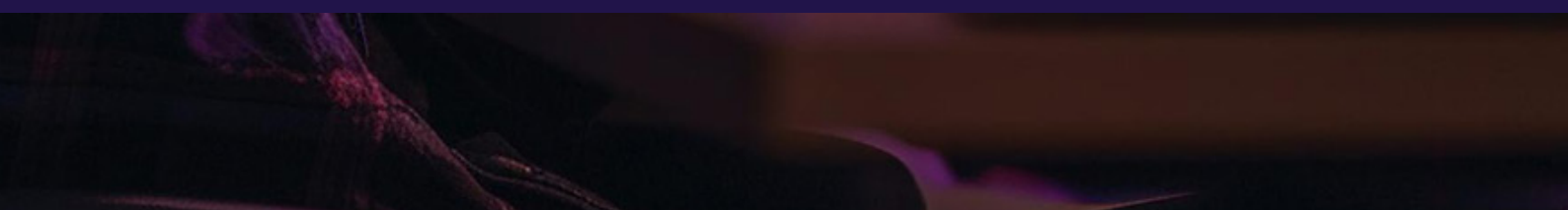
TECNOLÓGICA DEL ORIENTE
INSTITUCIÓN DE EDUCACIÓN SUPERIOR



FUNDAMENTOS DE *SOFTWARE*

DISEÑO ORIENTADO A OBJETOS: CONCEPTOS DE CLASES, OBJETOS Y HERENCIA



DISEÑO ORIENTADO A OBJETOS: CONCEPTOS DE CLASES, OBJETOS Y HERENCIA

El diseño orientado a objetos (OO), es un paradigma que organiza el desarrollo de ***software*** en torno a objetos que representan entidades del mundo real o conceptos abstractos. Este enfoque permite crear sistemas modulares, escalables y fáciles de mantener, destacándose por sus principios como encapsulación, abstracción, herencia y polimorfismo. A continuación, explicaremos los conceptos de clases, objetos y herencia, fundamentales para entender y aplicar este paradigma.

Clases: el plano de construcción de los objetos

Una **clase** es una plantilla o estructura que define las características y comportamientos de un grupo de objetos. Es el modelo conceptual que especifica los **atributos** (datos o propiedades) y **métodos** (funcionalidades) comunes a los objetos que se derivan de ella. La clase no es un objeto en sí misma; es un concepto que describe cómo deben ser los objetos.

Componentes de una clase:

1. Atributos: representan las características o datos de la clase. Se traducen en variables.

   o Ejemplo: en una clase **Persona**, los atributos podrían ser nombre, edad y dirección.

2. Métodos: representan las operaciones o funciones que los objetos pueden realizar.

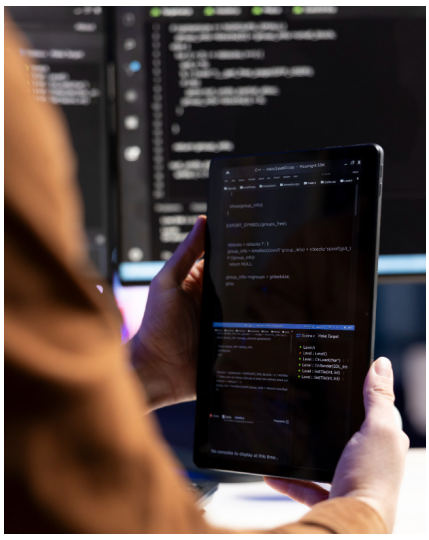
   o Ejemplo: en la clase **Persona**, un método podría ser hablar ().

3. Constructores: métodos especiales que permiten inicializar los objetos, cuando se crean.

   o Ejemplo: en Persona, un constructor puede recibir parámetros como nombre y edad, para establecer sus valores.

4. Modificadores de acceso: controlan la visibilidad de los atributos y métodos.

   o Ejemplos: *public, private, protected.*



Ejemplo de una clase en Código (Java):

```java
public class Persona {
// Atributos
    private String nombre;
    private int edad;

// Constructor
public Persona (String nombre, int edad) {
    this.nombre nombre;
    this.edad edad;
  }

// Métodos
public void hablar() {
    System.out.println(nombre + "está hablando.");
  }
public int obtenerEdad() {
    return edad;
  }
}
```

Aguilar, J. (2024). Clase Java. [Clase]. Creado con IDE Eclipse.

Objetos: instancias de clases

Un objeto es una instancia concreta de una clase. Es la representación en memoria de la estructura definida por la clase, que tiene atributos únicos y puede ejecutar métodos definidos por su clase.

Cada objeto tiene:

- Estado: representado por los valores de sus atributos.
- Comportamiento: determinado por los métodos de la clase.
- Identidad: su referencia única en la memoria.

Relación entre clases y objetos:

La clase es el molde, mientras que el objeto es la figura creada a partir del molde. Por ejemplo, **Persona** es la clase, y Juan es un objeto de la clase **Persona**.

Ejemplo de creación y uso de objetos:

```
public class Main {
public static void main(String[] args) {
// Crear un objeto de la clase Persona
Persona juan = new Persona ("Juan", 30);

// Usar métodos del objeto
juan.hablar(); // Salida: "Juan está hablando."
System.out.println("Edad: " + juan.obtenerEdad()); // Salida: "Edad: 30"
  }
}
```

Aguilar, J. (2024). Uso de objetos en Java. [clase]. Creado con IDE Eclipse.

Herencia: reutilización y especialización de clases

La herencia es un mecanismo que permite que una clase (llamada clase hija o subclase), adquiera los atributos y métodos de otra clase (llamada clase padre o superclase). Este concepto promueve la reutilización de código y la especialización de clases.

Características claves:

1. Relación “es-un”: la subclase es un tipo específico de la superclase.

- Ejemplo: un Estudiante es un tipo de **Persona**.

2. Extensibilidad: las subclases pueden agregar nuevos atributos o métodos y sobrescribir (*override*) los métodos de la superclase.

o Ejemplo: una subclase **Estudiante** puede tener un atributo adicional, matrícula.

3. Polimorfismo: las instancias de una subclase pueden ser tratadas como objetos de la superclase, permitiendo un comportamiento genérico.

Componentes:

- Superclase: clase de la cual se heredan atributos y métodos.

- Subclase: clase que hereda de la superclase y puede extenderse o modificar su comportamiento.

- Palabras claves (depende del lenguaje de programación):

o extends(Java), :(C#), inheritance(Python).

Ejemplo de herencia (Java):

```
// Superclase
public class Persona {
protected string nombre;

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

public void presentarse() {
    System.out.println("Hola, soy" + nombre);
  }
}
```

Aguilar, J. (2024). Herencia en Java. [Clase]. Creado con IDE Eclipse.

```
// Subclase
public class Estudiante extends Persona {
private String matricula;

public Estudiante (string nombre, string matricula) {
     super(nombre); // LLama al constructor de La superclase
     this.matricula = matricula;
   }

   @Override
   public void presentarse() {
     super.presentarse(); // LLama al método de La superclase
     System.out.println("Mi matrícula es " + matricula);
   }
}
```

Aguilar, J. (2024). Clase hija en Java. [Clase]. Creado con IDE Eclipse.

```
// Uso
public class Main {
    public static void main(String[ ]args) {
      Estudiante ana = new Estudiante ("Ana", "A12345");
      ana.presentarse();
    // Salida:
    // Hola, soy Ana
    // Mi matrícula es A12345
  }
}
```

Aguilar, J. (2024). Implementación en Java. [clase]. Creado con IDE Eclipse.