TECNOLÓGICA DEL ORIENTE
INSTITUCIÓN DE EDUCACIÓN SUPERIOR

PROGRAMACIÓN ORIENTADA A OBJETOS

CÓDIGOS EDITABLES

CÓDIGOS EDITABLES DE LOS EJEMPLOS DE POO

1.1. Ejemplo de definición de clase en Java:

```
public class Libro {
    private String titulo;
    private String autor;
    private String isbn;
    public Libro(String titulo, String autor, String isbn) {
        this.titulo = titulo;
        this.autor = autor;
        this.isbn = isbn;
    }
        // Otros métodos y propiedades...
}
```

1.2. Ejemplo de instanciación de objeto en Java:

```
Libro miLibro = new Libro("El Gran Gatsby", "F. Scott Fitzgerald", "978-0743273565");
```

1.3. Ejemplo de definición de atributos y métodos en Java:

```
public class CuentaBancaria {
    private String numeroCuenta;
    private double saldo;
    public CuentaBancaria(String numeroCuenta, double saldoInicial) {
        this.numeroCuenta = numeroCuenta;
        this.saldo = saldoInicial;
    }
    public void depositar(double cantidad) {
        saldo += cantidad;
    }
    public void retirar(double cantidad) {
        if (saldo >= cantidad) {
            saldo -= cantidad;
        } else {
            System.out.println("Fondos insuficientes");
        }
    }
    public double obtenerSaldo() {
        return saldo;
    }
}
```

1.4. Ejemplo de un diagrama de clases UML básico:

```
classDiagram
    class Estudiante {
        -id: int
        -nombre: string
        +matricularse(Curso)
    }
    class Curso {
        -codigo: string
        -nombre: string
        +agregarEstudiante(Estudiante)
    }
    class Profesor {
        -id: int
        -nombre: string
        +enseñar(Curso)
    }
    Estudiante "m" --> "n" Curso : se matricula en >
    Curso "1" --> "1" Profesor : es enseñado por >
```

1.5. Ejemplo en Java:

```
abstract class Forma {
    }
}
abstract double calcularArea();
    abstract double calcularPerimetro();
}
class Rectangulo extends Forma {
    double longitud;
    double ancho;

    Rectangulo(double longitud, double ancho) {
        this.longitud = longitud;
        this.ancho = ancho;
    }
    double calcularArea() {
        return longitud * ancho;
    }
    double calcularPerimetro() {
        return 2 * (longitud + ancho);
```

```
    }
}
class Circulo extends Forma {
    double radio;
    Circulo(double radio) {
        this.radio = radio;
    }
    double calcularArea() {
        return Math.PI * radio * radio;
    }
    double calcularPerimetro() {
        return 2 * Math.PI * radio;
```

2.1. Ejemplo en Java:

```
public class CuentaBancaria {
    private double saldo;  // Sólo accesible dentro de la clase
    String titular;       // Default: accesible dentro del paquete
    protected int tipo;    // Accesible dentro del paquete y por subclases
    public int numeroCuenta;  // Accesible desde cualquier lugar
        public void depositar(double cantidad) {
        saldo += cantidad;
    }
        private boolean tieneFondosSuficientes(double cantidad) {
        return saldo >= cantidad;
    }
        // Otros métodos...
}
```

2.2. Ejemplo en Java:

```
public class Persona {
    private String nombre;
    private int edad;
    public String getNombre() {
        return nombre;
    }
    public void setNombre(String nombre) {
        this.nombre = nombre;
    }
    public int getEdad() {
        return edad;
    }
```

```
    public void setEdad(int edad) {
        if (edad >= 0) {
            this.edad = edad;
        } else {
            throw new IllegalArgumentException("La edad no puede ser negativa");
        }
    }
}
```

2.3. Ejemplo de abstracción en Java:

```
java
abstract class Figura {
    abstract double calcularArea();
    abstract double calcularPerimetro();
}
class Rectangulo extends Figura {
    private double ancho;
    private double alto;
    public Rectangulo(double ancho, double alto) {
        this.ancho = ancho;
        this.alto = alto;
    }
    double calcularArea() {
        return ancho * alto;
    }
    double calcularPerimetro() {
        return 2 * (ancho + alto);
    }
}
class Circulo extends Figura {
    private double radio;
    public Circulo(double radio) {
        this.radio = radio;
    }
    double calcularArea() {
        return Math.PI * radio * radio;
    }
    double calcularPerimetro() {
        return 2 * Math.PI * radio;
    }
}
```

```
class Coche2 {
    private Arrancable motor;

    public Coche2(Arrancable motor) {
        this.motor = motor;
    }

    public void arrancar() {
        motor.encender();
    }
}

// Uso
MotorElectrico motorElectrico = new MotorElectrico();
Coche2 coche = new Coche2(motorElectrico);
coche.arrancar();
```

2.4. Ejemplo en Java:

```
interface Imprimible {
    void imprimir();
}
class Documento implements Imprimible {
    private String contenido;
    public Documento(String contenido) {
        this.contenido = contenido;
    }
    public void imprimir() {
        System.out.println("Imprimiendo documento: " + contenido);
    }
}
class Imagen implements Imprimible {
    private String ruta;
    public Imagen(String ruta) {
        this.ruta = ruta;
    }
    public void imprimir() {
        System.out.println("Imprimiendo imagen: " + ruta);
    }
}
// Uso
Imprimible documento = new Documento("Hola, mundo!");
Imprimible imagen = new Imagen("/ruta/a/la/imagen.jpg");

documento.imprimir(); // Imprimiendo documento: Hola, mundo!
imagen.imprimir();    // Imprimiendo imagen: /ruta/a/la/imagen.jpg
```

2.5. Ejemplo en Java:

```
public class CuentaBancaria {
    private double saldo;
    private String titular;
      public CuentaBancaria(String titular) {
        this.titular = titular;
        this.saldo = 0;
    }
        public void depositar(double cantidad) {
        saldo += cantidad;
    }
```

```
    public void retirar(double cantidad) {
    if (saldo >= cantidad) {
        saldo -= cantidad;
    } else {
        System.out.println("Fondos insuficientes");
    }
  }
    public double getSaldo() {
    return saldo;
  }
}
```

3.1. Ejemplo en Java:

```
class Estudiante {
  private String nombre;
  private List<Curso> cursos;
  public Estudiante(String nombre) {
    this.nombre = nombre;
    this.cursos = new ArrayList<>();
  }
  public void inscribirEnCurso(Curso curso) {
    cursos.add(curso);
    curso.agregarEstudiante(this);
  }
}
class Curso {
  private String nombre;
  private List<Estudiante> estudiantes;
  public Curso(String nombre) {
    this.nombre = nombre;
    this.estudiantes = new ArrayList<>();
  }
  public void agregarEstudiante(Estudiante estudiante) {
    estudiantes.add(estudiante);
  }
}
// Uso
Estudiante estudiante1 = new Estudiante("John");
Estudiante estudiante2 = new Estudiante("Jane");
Curso curso1 = new Curso("Matemáticas");
Curso curso2 = new Curso("Historia");
```

```
estudiante1.inscribirEnCurso(curso1);
estudiante1.inscribirEnCurso(curso2);
estudiante2.inscribirEnCurso(curso1);
```

3.2. Ejemplo en Java:

```
class Inventario {
    private Map<String, Integer> stock;
    public Inventario() {
        stock = new HashMap<>();
        stock.put("Producto1", 10);
        stock.put("Producto2", 5);
    }
    public boolean comprobarDisponibilidad(String producto, int cantidad) {
        if (stock.containsKey(producto) && stock.get(producto) >= cantidad) {
            return true;
        }
        return false;
    }
    public void reducirStock(String producto, int cantidad) {
        if (comprobarDisponibilidad(producto, cantidad)) {
            stock.put(producto, stock.get(producto) - cantidad);
        }
    }
}
class Envio {
    public void organizarEnvio(String producto, String direccion) {
        System.out.println("Organizando envío de " + producto + " a " + direccion);
    }
}
class Pedido {
    private Inventario inventario;
    private Envio envio;
    public Pedido(Inventario inventario, Envio envio) {
        this.inventario = inventario;
        this.envio = envio;
    }
    public void procesarPedido(String producto, int cantidad, String direccion) {
        if (inventario.comprobarDisponibilidad(producto, cantidad)) {
            inventario.reducirStock(producto, cantidad);
            envio.organizarEnvio(producto, direccion);
        } else {
```

```
            System.out.println("Producto no disponible: " + producto);
        }
    }
}
// Uso
Inventario inventario = new Inventario();
Envio envio = new Envio();
Pedido pedido = new Pedido(inventario, envio);
pedido.procesarPedido("Producto1", 2, "123 Calle Principal");
```

3.3. Ejemplo en Java:

```
// Acoplamiento alto
class Motor {
    public void encender() {
        // Lógica para encender el motor
    }
}
class Coche {
    private Motor motor;
    public Coche() {
        motor = new Motor();
    }
    public void arrancar() {
        motor.encender();
    }
}
// Acoplamiento bajo
interface Arrancable {
    void encender();
}
class MotorElectrico implements Arrancable {
    public void encender() {
        // Lógica para encender el motor eléctrico
    }
}
class Coche2 {
    private Arrancable motor;
    public Coche2(Arrancable motor) {
        this.motor = motor;
    }
    public void arrancar() {
```

```
        motor.encender();
    }
}
// Uso
MotorElectrico motorElectrico = new MotorElectrico();
Coche2 coche = new Coche2(motorElectrico);
coche.arrancar();
```

3.4. Ejemplo en Java:

```
// Composición
class Universidad {
    private List<Departamento> departamentos;
    public Universidad() {
        departamentos = new ArrayList<>();
    }
        public void agregarDepartamento(Departamento departamento) {
        departamentos.add(departamento);
    }
}
class Departamento {
    private String nombre;
    public Departamento(String nombre) {
        this.nombre = nombre;
    }
}
// Agregación
class Estudiante {
    private String nombre;
    public Estudiante(String nombre) {
        this.nombre = nombre;
    }
}
class Universidad2 {
    private List<Estudiante> estudiantes;
    public Universidad2() {
        estudiantes = new ArrayList<>();
    }
        public void matricularEstudiante(Estudiante estudiante) {
        estudiantes.add(estudiante);
    }
}
```

```
// Uso
Universidad universidad = new Universidad();
Departamento departamento1 = new Departamento("Ciencia de la Computación");
Departamento departamento2 = new Departamento("Matemáticas");
universidad.agregarDepartamento(departamento1);
universidad.agregarDepartamento(departamento2);
Estudiante estudiante1 = new Estudiante("John");
Estudiante estudiante2 = new Estudiante("Jane");
Universidad2 universidad2 = new Universidad2();
universidad2.matricularEstudiante(estudiante1);
universidad2.matricularEstudiante(estudiante2);
```

3.5. Ejemplo en Java:

```
class Producto {
    private String nombre;
    private double precio;
    public Producto(String nombre, double precio) {
        this.nombre = nombre;
        this.precio = precio;
    }
    public String getNombre() {
        return nombre;
    }
    public double getPrecio() {
        return precio;
    }
}
class LineaItem {
    private Producto producto;
    private int cantidad;
    public LineaItem(Producto producto, int cantidad) {
        this.producto = producto;
        this.cantidad = cantidad;
    }
    public Producto getProducto() {
        return producto;
    }
    public int getCantidad() {
        return cantidad;
    }
      public double getTotal() {
```

```
        return producto.getPrecio() * cantidad;
    }
}
class Recibo {
    private List<LineaItem> items;
    private double total;
    public Recibo(List<LineaItem> items) {
        this.items = items;
        this.total = items.stream().mapToDouble(LineaItem::getTotal).sum();
    }
    public double getTotal() {
        return total;
    }
}


class Cesta {
    private List<LineaItem> items;

    public Cesta() {
        items = new ArrayList<>();
    }
    public void addItem(Producto producto, int cantidad) {
        LineaItem item = new LineaItem(producto, cantidad);
        items.add(item);
    }
    public Recibo checkout() {
        return new Recibo(items);
    }
}

// Uso
Producto producto1 = new Producto("Leche", 2.50);
Producto producto2 = new Producto("Pan", 1.75);

Cesta cesta = new Cesta();
cesta.addItem(producto1, 2);
cesta.addItem(producto2, 1);

Recibo recibo = cesta.checkout();
System.out.println("Total: " + recibo.getTotal()); // Total: 6.75
```