UTEC
Universidad Tecnológica

# Interfaces

**Ejercicio 1: Calculadora de Áreas**

Crea una interfaz llamada `Calculable` con los métodos `calcularArea` y `calcularPerimetro`. Implementa esta interfaz en las clases `Circulo`, `Rectangulo` y `Triangulo`. En cada clase, implementa los cálculos específicos para el área y el perímetro de la figura geométrica correspondiente. Luego, crea instancias de cada clase, llama a los métodos de la interfaz y muestra los resultados.

**Ejercicio 2: Ordenamiento de Números**

Define una interfaz llamada `Ordenable` que tenga un método llamado `ordenarNumeros` que toma un arreglo de enteros y lo ordena de menor a mayor. Implementa esta interfaz en una clase llamada `OrdenadorNumeros` utilizando un algoritmo de ordenamiento como el de burbuja. Crea un programa que inicialice un arreglo de números, utilice la implementación de la interfaz para ordenarlo y finalmente muestre el resultado ordenado.

**Ejercicio 3: Sistema de Nómina**

Crea una interfaz llamada `Empleado` con los métodos `calcularSalario` y `mostrarInformacion`. Implementa esta interfaz en las clases `EmpleadoAsalariado` y `EmpleadoPorHoras`. En la clase `EmpleadoAsalariado`, el salario se calculará mensualmente, mientras que en la clase `EmpleadoPorHoras`, se calculará por horas trabajadas. Crea instancias de ambas clases, llama a los métodos de la interfaz y muestra la información de los empleados.

**Ejercicio 4: Conexión a Bases de Datos**

Define una interfaz llamada `ConexionBD` que tenga métodos para `conectar`, `desconectar` y `consultarBD`. Implementa esta interfaz en las clases `MySQLConexion` y `OracleConexion`. En cada clase, simula la conexión y desconexión, y realiza una consulta ficticia a la base de datos correspondiente. Luego, crea instancias de ambas clases, llama a los métodos de la interfaz y muestra mensajes indicando el estado de la conexión y el resultado de la consulta.


**Ejercicio 5: Juego de Cartas Mejorado**

Crea una interfaz llamada `JuegoCartas` con métodos para `barajar` y `repartirCartas`. Implementa esta interfaz en clases como `BarajaEspanola` y `BarajaPoker`. Mejora el ejercicio anterior permitiendo que el juego pueda tener varios jugadores. Crea instancias de las clases, realiza las operaciones de barajado y reparto, y muestra las cartas asignadas a cada jugador.

# Clases Abstractas

**Ejercicio 1: Sistema de Archivos**

Crea una clase abstracta llamada `Archivo` con atributos como `nombre` y `tamaño`. Luego, define clases concretas como `Documento` y `Carpeta` que hereden de `Archivo`. La carpeta debe contener una lista de archivos. Implementa un método abstracto `calcularEspacioOcupado` en la clase `Archivo` que sea implementado en las clases hijas.

**Ejercicio 2: Juego de Rol**

Crea una clase abstracta llamada `Personaje` con atributos como `nombre` y `nivel`. Define clases concretas como `Guerrero` y `Mago` que hereden de `Personaje`. Implementa métodos abstractos como `atacar` y `defender` en la clase `Personaje` que sean implementados en las clases hijas.

**Ejercicio 3: Figuras 3D**

Define una clase abstracta llamada `Figura3D` que tenga un método abstracto `calcularVolumen`. Luego, crea clases concretas como `Esfera`, `Cubo` y `Cono` que hereden de `Figura3D`. Implementa el método `calcularVolumen` en cada clase concreta.

**Ejercicio 4: Empleados con Jerarquía**

Crea una clase abstracta llamada `Empleado` con atributos como `nombre` y `salarioBase`. Luego, define clases concretas como `Gerente` y `EmpleadoBase` que hereden de `Empleado`. Implementa métodos abstractos como `calcularSalario` en la clase `Empleado` que sean implementados en las clases hijas.

**Ejercicio 5: Animales Marinos**

UTEC
Universidad Tecnológica

Crea una clase abstracta llamada `AnimalMarino` con atributos como `nombre` y `profundidadMaxima`. Define clases concretas como `PezPayaso` y `Tiburon` que hereden de `AnimalMarino`. Implementa métodos abstractos como `nadar` en la clase `AnimalMarino` que sean implementados en las clases hijas.

# Manejo de archivos y excepciones

**Ejercicio 1: Registro de Estudiantes con Manejo de Excepciones**

Crea una aplicación de registro de estudiantes que permita ingresar información como nombre, edad y calificación. Implementa manejo de excepciones para asegurar que los datos ingresados sean válidos. Guarda la información de cada estudiante en un archivo de texto llamado "estudiantes.txt". Utiliza una estructura de bucle para permitir al usuario ingresar varios estudiantes. Maneja las excepciones específicas para la entrada de datos incorrecta.

**Ejercicio 2: Manejo de Excepciones en Lectura de Archivo de Configuración**

Desarrolla una aplicación que lea un archivo de configuración en formato JSON. El archivo de configuración debe contener información como nombre del servidor, puerto y credenciales de conexión a una base de datos. Utiliza la biblioteca `Gson` para el manejo de JSON y maneja excepciones específicas como `FileNotFoundException` y `JsonSyntaxException`. Si ocurre alguna excepción, muestra un mensaje amigable para el usuario indicando el problema.

**Ejercicio 3: Calculadora con Historial y Excepciones Extendidas**

Amplía la calculadora del ejercicio anterior para incluir operaciones más avanzadas como raíz cuadrada y potencia. Además, guarda un historial de todas las operaciones realizadas en un archivo de texto llamado "historial.txt". Implementa excepciones personalizadas para casos como intentar calcular la raíz cuadrada de un número negativo o realizar una potencia con un exponente no entero.

**Ejercicio 4: Sistema de Reservas de Habitaciones con Validación de Fechas**

Desarrolla un sistema de reservas de habitaciones de hotel. Permite a los usuarios reservar habitaciones ingresando su nombre, número de habitación y fecha de llegada y salida.

Universidad Tecnológica

Guarda la información de las reservas en un archivo de texto llamado "reservas.txt". Implementa manejo de excepciones para validar que las fechas de llegada y salida sean válidas y que la habitación no esté reservada en esas fechas.

Nota: para pasar de un tipo String a un tipo Date (fecha) se deberá realizar una conversión. Puedes ayudarte con la siguiente función, suponiendo que el formato de ingreso de la fecha es dd/MM/yyyy:

```
private static Date parseFecha(String fecha) throws ParseException {
      SimpleDateFormat dateFormat = new SimpleDateFormat("dd/MM/yyyy");
      return dateFormat.parse(fecha);
}
```

**Ejercicio 5: Manejo de Excepciones en Lectura de Archivo CSV**

Crea una clase llamada `CsvReader` que pueda leer un archivo CSV (valores separados por comas) y convertirlo en objetos Java. Maneja excepciones específicas para asegurarte de que el formato del archivo sea correcto y pueda ser procesado adecuadamente. Utiliza la biblioteca `OpenCSV` para facilitar la lectura del archivo CSV.