Sistema de Gestión de Biblioteca Digital

Descripción General

Desarrollar un sistema básico de gestión de biblioteca digital que permita administrar libros y realizar operaciones básicas de una biblioteca.

Requisitos Funcionales

RF1: Gestión de Libros

- El sistema debe permitir crear libros con:
    - Título (texto, obligatorio)
    - Autor (texto, obligatorio)
    - Número de páginas (entero positivo, obligatorio)
- Cada libro debe poder mostrar su información completa

RF2: Gestión de Biblioteca

- La biblioteca debe poder almacenar múltiples libros
- Debe permitir agregar nuevos libros
- Debe poder listar todos los libros existentes
- Debe mostrar el total de libros almacenados

RF3: Operaciones del Sistema

1. Crear al menos 3 libros de ejemplo
2. Crear una biblioteca
3. Agregar los libros a la biblioteca
4. Mostrar catálogo completo
5. Mostrar estadísticas básicas

Reglas del problema:

- Un libro no puede tener 0 o menos páginas
- La biblioteca debe evitar duplicados (mismo título y autor)
- La información debe mostrarse de forma legible

Estructura de Clases

1. Clase Libro

Responsabilidad: Representar un libro individual

Atributos privados:

- titulo (string): Título del libro
- autor (string): Autor del libro
- paginas (int): Número de páginas

Métodos públicos:

- __construct(string $titulo, string $autor, int $paginas)
    - Inicializa el libro con datos básicos
    - Valida que páginas sea positivo
- getInfo(): string
    - Devuelve información formateada del libro
- Getters para cada atributo (opcional pero buena práctica)

2. Clase Biblioteca

Responsabilidad: Gestionar una colección de libros

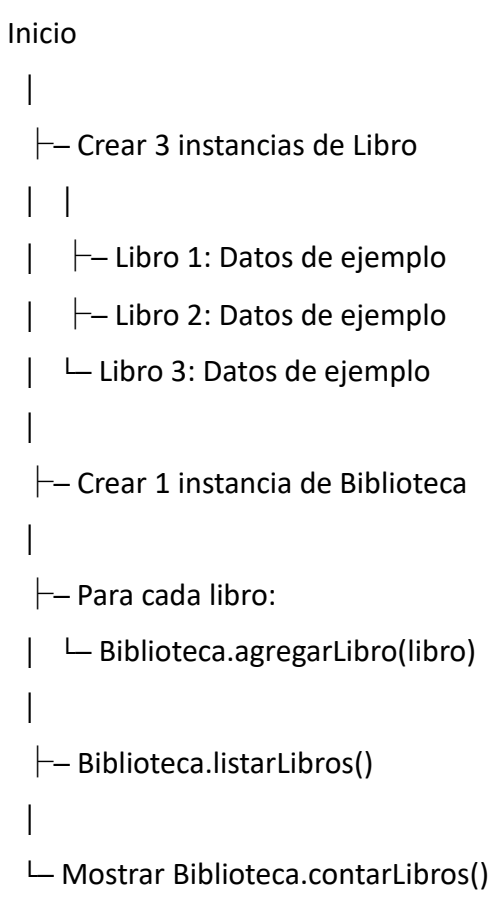
Atributos privados:

- libros (array): Lista de objetos Libro

Métodos públicos:

- __construct()
    - Inicializa array vacío
- agregarLibro(Libro $libro): void
    - Añade un libro a la colección
    - Muestra confirmación
- listarLibros(): void
    - Muestra todos los libros con formato
    - Maneja caso biblioteca vacía
- contarLibros(): int
    - Devuelve cantidad total de libros

Flujo del Programa



Sistema de Validación de Usuarios con Excepciones

Descripción General

Desarrollar un sistema de registro de usuarios que implemente manejo de excepciones personalizadas para validar datos de entrada. El sistema debe demostrar el uso de excepciones para control de errores en lugar de valores de retorno.

Requisitos Funcionales

RF1: Registro de Usuario

- El sistema debe permitir registrar usuarios con:
    - Nombre de usuario (string, 3-20 caracteres)
    - Email (formato válido de email)
    - Edad (entero entre 18 y 120)
    - Contraseña (mínimo 8 caracteres)

RF2: Validaciones con Excepciones

Crear excepciones personalizadas para cada tipo de error:

1. LongitudInvalidaException - Para longitud incorrecta de nombre/contraseña
2. EmailInvalidoException - Para formato de email incorrecto
3. EdadInvalidaException - Para edad fuera de rango
4. UsuarioExistenteException - Para usuario ya registrado

RF3: Gestión del Sistema

- Clase Usuario para representar cada usuario
- Clase SistemaRegistro para gestionar múltiples usuarios
- Método registrarUsuario() que lance excepciones específicas
- Bloque try-catch en el programa principal para manejar errores

Reglas de Negocio

- **Nombre de usuario**: 3-20 caracteres, solo letras y números
- **Email**: Formato válido, debe contener "@" y dominio
- **Edad**: 18-120 años
- **Contraseña**: Mínimo 8 caracteres
- No pueden existir dos usuarios con mismo nombre o email

Práctica:

1. Excepciones Personalizadas

Crear 4 clases de excepción que extiendan de Exception:

- Cada una con mensaje específico de error
- Posibilidad de código de error personalizado

2. Clase Usuario

Atributos: username, email, edad, password
Métodos: Constructor, getters, validación básica

3. Clase SistemaRegistro

Atributos: Array de usuarios registrados
Métodos:

- registrarUsuario(Usuario $usuario): Valida y registra, lanza excepciones
- validarDatos(Usuario $usuario): Realiza validaciones
- usuarioExiste(Usuario $usuario): Verifica duplicados

4. Programa Principal

- Crear instancias de SistemaRegistro
- Intentar registrar varios usuarios (algunos válidos, otros con errores)
- Usar bloques try-catch para manejar cada tipo de excepción
- Mostrar resultados claros de cada intento