Trabajo Práctico Integrador

[Sistema de Gestión de Recursos Académicos]

Objetivo General

Desarrollar un sistema integral de gestión académica como un Trabajo Práctico Integrador (TPI) que demuestre dominio de los principales conceptos de Programación Orientada a Objetos en Java, integrando herencia, polimorfismo, interfaces funcionales, manejo de excepciones y organización de paquetes.

Contexto

La **UniversidadRequiriente** necesita un sistema de gestión académica versátil que permita administrar diferentes tipos de recursos, usuarios y actividades académicas.

Estructura del Proyecto

1. Herencia y Polimorfismo

Jerarquía de Recursos Académicos

Crear una jerarquía de clases para representar diferentes recursos académicos:

- Clase abstracta `RecursoAcademico`:
  - Atributos comunes:
    * `String identificador`
    * `String titulo`
    * `LocalDate fechaCreacion`
    * `String autor`
  - Métodos abstractos:
    * `double calcularRelevancia()`
    * `void mostrarDetalles()`
- Clases hijas que hereden de `RecursoAcademico`:
  1. `Libro`
     - Atributos adicionales:
       * `int numeroPaginas`

* `String editorial`

- Implementación de métodos abstractos

- Método específico: `boolean esLibroDigital()`

2. `Articulo`

- Atributos adicionales:

* `List<String> palabrasClave`

* `String revista`

- Implementación de métodos abstractos

- Método específico: `int contarPalabrasClave()`

3. `TrabajoInvestigacion`

- Atributos adicionales:

* `List<String> autores`

* `String lineaInvestigacion`

- Implementación de métodos abstractos

- Método específico: `boolean tieneFinanciamiento()`

2. Abstracción e Interfaces

Interfaces de Gestión

- Interfaz `Clasificable`:

```
public interface Clasificable {
    String[] obtenerCategoriasClasificacion();
    void asignarCategoria(String categoria);
}
```

- Interfaz `Evaluable`:

```
public interface Evaluable {
    double obtenerPuntaje();
    void realizarEvaluacion(Evaluador evaluador);
}
```

3. Interfaces Funcionales y Expresiones Lambda

Sistema de Filtrado y Análisis de Recursos

- Interfaz funcional personalizada `FiltroRecurso`:

```
@FunctionalInterface
public interface FiltroRecurso {
    boolean evaluar(RecursoAcademico recurso);
}
```

- Clase `GestorRecursos` con métodos que utilicen expresiones lambda:
  - Métodos para:
    * Filtrar recursos por diferentes criterios
    * Ordenar recursos
    * Aplicar transformaciones
    * Generar informes estadísticos

4. Excepciones y Paquetes

Manejo de Excepciones Personalizadas

- Paquetes propuestos:

* `com.universidad.recursos`
* `com.universidad.excepciones`
* `com.universidad.gestores`
* `com.universidad.interfaces`

- Excepciones personalizadas:

1. `RecursoNoEncontradoException`
2. `CategoriaInvalidaException`
3. `LimiteRecursosException`

Requerimientos Adicionales

Funcionalidades del Sistema

1. Registro de recursos académicos
2. Clasificación de recursos
3. Evaluación de relevancia
4. Generación de informes
5. Filtrado avanzado de recursos

Entregables

1. Código fuente completo
2. Diagrama de clases UML
3. Ejemplos de uso de cada componente (en la clase Principal)
4. Informe que explique las decisiones de diseño (*en la defensa del TPI*)

Criterios de Evaluación

- Correcta implementación de herencia y polimorfismo
- Uso adecuado de interfaces
- Manejo eficiente de excepciones
- Implementación de interfaces funcionales
- Organización y estructura del código
- Claridad y legibilidad
- Aplicación de buenas prácticas de programación

Consideraciones Finales

- Fecha de entrega: 26 de noviembre de 2024 (18.30 hs)
- Trabajo individual
- Presentación oral y defensa del proyecto (martes 26 y miércoles 27 del noviembre de 2024)

Bonus Track

Implementar una interfaz gráfica simple (En Java FX) para demostrar el funcionamiento del sistema. (*Requisito no obligatorio, si para los Finales de diciembre o febrero*)