PROGRAMACIÓN ORIENTADA A OBJETOS
Persistencia
2025-01
Laboratorio 6/6 [:)]

OBJETIVOS

1. Completar el código de un proyecto considerando requisitos funcionales.
2. Diseñar y construir los métodos básicos de manejo de archivos: abrir, guardar, importar y exportar.
3. Controlar las excepciones generadas al trabajar con archivos.
4. Experimentar las prácticas XP : ⊡ Only one pair integrates code at a time.
   ⊡Use collective ownership.

ENTREGA

- ➔ Incluyan en un archivo .zip los archivos correspondientes al laboratorio. El nombre debe ser los dos apellidos de los miembros del equipo ordenados alfabéticamente.
- ➔ Deben publicar el avance al final de la sesión y la versión definitiva en la fecha indicada en los espacios preparados para tal fin.

DESARROLLO

Preparando

En este laboratorio vamos a extender el proyecto `schelling` adicionando un menú barra con las opciones básicas de entrada-salida y las opciones estándar nuevo y salir.

1. En su directorio descarguen la versión del proyecto realizado por ustedes para el laboratorio 03 y preparen el ambiente para trabajar desde **CONSOLA**
2. Ejecuten el programa, revisen la funcionalidad.

Creando la maqueta

[En lab06.doc, *.asta y *.java] [NO OLVIDEN BDD y MDD]

En este punto vamos a construir la maqueta correspondiente a esta extensión siguiendo el patrón MVC.

1. **MODELO:** Preparen en la clase fachada del dominio los métodos correspondientes a las cuatro opciones básicas de entrada-salida (`open, save, import y export`). Los métodos deben simplemente propagar una `CityException` con el mensaje de "Opción `nombreOpción` en construcción. Archivo `nombreArchivo`". Los métodos deben tener un parámetro `File`.
2. **VISTA :** Construyan un menú barra que ofrezca, además de las opciones básicas de entrada-salida, las opciones estándar de nuevo y salir (`Nuevo, Abrir, Guardar como, Importar, Exportar como, Salir`). No olviden incluir los separadores. Para esto creen el método `prepareElementsMenu.` Capturen la pantalla del menú.
3. **CONTROLADOR:** Construyan los oyentes correspondientes a las seis opciones. Para esto creen el método `prepareActionsMenu` y los métodos base del controlador (`optionOpen, optionSave, optionImport, optionExport, optionNew, optionExit`), En las opciones que lo requieran usen un `FileChooser` y atiendan la excepción. Estos métodos llaman el método correspondiente de la capa de dominio que por ahora sólo lanza una excepción. Ejecuten las diferentes acciones del menú y para cada una de ellas capture una pantalla significativa.

Implementando salir y nuevo

[En lab06.doc, *.asta y *.java] [NO OLVIDEN BDD y MDD]

Las opciones salir y nuevo van a ofrecer los dos servicios estándar de las aplicaciones. El primero no requiere ir a capa de dominio y el segundo sí.

1. Construyan el método `optionExit` que hace que se termine la aplicación. No es necesario incluir confirmación.
2. Construyan el método `optionNew` que crea una nueva ciudad. Capturen una pantalla significativa.

Implementando salvar y abrir

[En lab06.doc, *.asta y *.java] [NO OLVIDEN BDD y MDD]

Las opciones salvar y abrir van a ofrecer servicios de persistencia de la ciudad como objeto. Los nombres de los archivos deben tener como extensión .dat.

1. Copien las versiones actuales de `open` y `save` y renómbrenlos como `open00` y `save00`
2. Construyan el método `save` que ofrece el servicio de guardar en un archivo el estado actual de la ciudad. Por ahora para las excepciones sólo consideren un mensaje de error general. No olviden diseño y pruebas de unidad.
3. Validen este método guardando el estado obtenido después de dos clics como `oneCity.dat`. ¿El archivo se creó en el disco? ¿Cuánto espacio ocupa?
4. Construyan el método `open` que ofrece el servicio de leer una ciudad de un archivo. Por ahora para las excepciones sólo consideren un mensaje de error general. No olviden diseño y pruebas de unidad.
5. Realicen una prueba de aceptación para este método iniciando la aplicación, creando un nuevo estado y abriendo el archivo `oneCity.dat`. Capturen imágenes significativas de estos resultados.

Implementando importar y exportar

[En lab06.doc, *.asta y *.java] [NO OLVIDEN BDD y MDD]

Estas operaciones nos van a permitir importar información de la ciudad desde un archivo de texto y exportarla. Los nombres de los archivos de texto deben tener como extensión .txt

Los archivos texto tienen una línea de texto por cada elemento

En cada línea asociada un elemento se especifica el tipo y la posición.

Person 10 10

Walker 20 20

1. Copien las versiones actuales de `import` y `export` y renómbrenlos como `import00` y `export00`
2. Construyan el método `export` que ofrece el servicio de exportar a un archivo texto, con el formato definido, el estado actual. Por ahora para las excepciones sólo consideren un mensaje de error general. No olviden diseño y pruebas de unidad.
3. Realicen una prueba de aceptación de este método: iniciando la aplicación y exportando como `oneCity.txt`. Editen el archivo y analicen los resultados. ¿Qué pasó?
4. Construyan el método `import` que ofrece el servicio de importar de un archivo texto con el formato definido. Por ahora sólo considere un mensaje de error general. No olviden diseño y pruebas de unidad.
   (Consulten en la clase String los métodos trim y split)
5. Realicen una prueba de aceptación de este par de métodos: iniciando la aplicación exportando a `oneCity.txt`. saliendo, entrando, creando una nueva e importando el archivo `otherCity.txt`. ¿Qué resultado obtuvieron? Capturen la pantalla final.
6. Realicen otra prueba de aceptación de este método escribiendo un archivo de texto correcto en `oneCity.txt`. e importe este archivo. ¿Qué resultado obtuvieron? Capturen la pantalla.

Analizando comportamiento
[En lab06.doc, *.asta y *.java] [NO OLVIDEN BDD y MDD]

1. Ejecuten la aplicación, den tres clics, salven a un archivo cualquiera y ábranlo. Describan el comportamiento
2. Ejecuten la aplicación, tres clics, exporten a un archivo cualquiera e importen. Describan el comportamiento
3. ¿Qué diferencias ven el comportamiento 1. y 2.? Expliquen los resultados.

Perfeccionando salvar y abrir
[En lab06.doc, *.asta y *.java] [NO OLVIDEN BDD y MDD]

1. Copien las versiones actuales de `open` y `save` y renómbrenlos como `open01` y `save01`
2. Perfeccionen el manejo de excepciones de los métodos `open` y `save` detallando los errores. No olviden pruebas de unidad.
3. Realicen una prueba de aceptación para validar uno de los nuevos mensajes diseñados, ejecútenla y capturen la pantalla final.

Perfeccionando importar y exportar.
[En lab06.doc, *.asta , cityErr.txt *.java] [NO OLVIDEN BDD y MDD]

1. Copien las versiones actuales de `import` y `export` y renómbrenlos como `import01` y `export01`
2. Perfeccionen el manejo de excepciones de los métodos `import` y `export` detallando los errores. No olviden pruebas de unidad.
3. Realicen una prueba de aceptación para validar uno de los nuevos mensajes diseñados, ejecútenla y capturen la pantalla final.

Perfeccionando importar. Hacia un minicompilador.
[En lab06.doc, *.asta , cityErr.txt *.java] [NO OLVIDEN BDD y MDD]

1. Copien las versiones actuales de `import` y `export` y renómbrenlos como `import02` y `export02`
2. Perfeccionen el método **`import`** para que, además de los errores generales, en las excepciones indique el detalle de los errores encontrados en el archivo (como un compilador) : número de línea donde se encontró el error, palabra que tiene el error y causa de error.
3. Escriban otro archivo con errores, llámelo cityErr.txt, para ir arreglándolo con ayuda de su "importador". Presente las pantallas que contengan los errores.

BONO. Perfeccionando importar. Hacia un minicompilador flexible.
[En lab06.doc, *.asta , schellingFlex.txt *.java] [NO OLVIDEN BDD y MDD]

1. Copien las versiones actuales de `import` y `export` y renómbrenlos como `import03` y `export03`
2. Perfeccionen los métodos `import` y `export` para que pueda servir para cualquier tipo de elementos creados en el futuro. No olviden pruebas de unidad.
   (Investiguen cómo crear un objeto de una clase dado su nombre)
3. Escriban otro archivo de pruebas, llámelo `cityErrG.txt`, para probar la flexibilidad. Presente las pantallas que contenga un error significativo.

RETROSPECTIVA

1. ¿Cuál fue el tiempo total invertido en el laboratorio por cada uno de ustedes ?(Horas/Hombre)
2. ¿Cuál es el estado actual del laboratorio? ¿Por qué?
3. Considerando las prácticas XP del laboratorio. ¿cuál fue la más útil? ¿por qué?
4. ¿Cuál consideran fue el mayor logro? ¿Por qué?
5. ¿Cuál consideran que fue el mayor problema técnico? ¿Qué hicieron para resolverlo?
6. ¿Qué hicieron bien como equipo? ¿Qué se comprometen a hacer para mejorar los resultados?
7. ¿Qué referencias usaron? ¿Cuál fue la más útil? Incluyan citas con estándares adecuados.