Sesión 9

Objetivo

Describir un problema relativo a gestión de un quiosco de prensa y revistas para llevar la contabilidad del stock y los pedidos que deben ser realizados en la clase presencial de la sesión 9 y completar algunas opciones como tares no presencial.

En esta sesión 9 sólo **se realizarán las funciones básicas y se incluirán excepciones**. Las operaciones de acceso a ficheros para lectura y escritura se dejan **para la sesión 10**.

Importa el proyecto `lab09_newsstand_starting_project`

Descripción general

Se desea diseñar e implementar un sistema de gestión de ventas y pedidos de un quiosco (Newsstand) que vende publicaciones de dos tipos: periódicos (newspaper) y revistas (magazine), de los que se guarda la siguiente información:

- **name** (cadena) Nombre del producto
- **inStock** (entero) Número de ejemplares sin vender
- **sales** (entero) Número de ejemplares vendidos

Los periódicos son publicados diariamente y para las revistas se debe guardar información sobre su periodicidad:

- **frequency** (Enumerado)**.** Periodicidad entre dos ediciones consecutivas. Los niveles de periodicidad son: QUARTERLY, MONTHLY, BIMONTHLY, DAILY, WEEKLY

A partir de los datos de ventas guardados en un fichero, se precisa generar los pedidos que deben realizarse y guardar los mismos en un fichero de pedidos.

Información de ventas

El fichero de entrada contiene una línea para cada publicación. Las líneas tienen una representación textual de las publicaciones y cada campo está separado del siguiente por el carácter tabulador ('\t'). Se proporciona el fichero publicaciones.txt con el siguiente contenido por línea:

<tipo de publicación> \t <Nombre> \t <stock>\t <vendidos> [\t <periodicidad>]

newspaper	Nueva España	14	30	
newspaper	El Mundo	4	10	
magazine	Hola	14	30	BIMONTHLY
magazine	PCWord	14	30	QUATERLY
magazine	Diez Minutos	4	10	WEEKLY
magazine	El Mueble	4	10	MONTHLY
magazine	Muy Interesante	8	20	DAILY

Generación de pedidos

La aplicación debe generar pedidos de compra, un pedido por cada publicación con menos de 10 copias en existencia.

Cada pedido debe incluir el nombre de la publicación y la cantidad de unidades solicitadas. Este número es calculado en función del tipo de publicación y cantidad en stock:

a) **Periódico:** Se piden tantos ejemplares como los vendidos más el doble de los ejemplares en stock.
b) **Revista:** Si el stock es menor que 5 se piden 20 ejemplares. Si es mayor o igual que 5, dependerá de la periodicidad. Si es semanal, se piden tantos ejemplares como los vendidos y en otro caso, se piden tantos como los vendidos más los ejemplares en stock.

Interacción con el usuario

La aplicación ofrece un menú de usuario con las siguientes opciones:

1. **Cagar publicaciones desde un fichero**
   Esta opción lee a partir de un fichero de texto los datos de todas las publicaciones del quiosco (newsstand). La aplicación lee el contenido del fichero y lo añade a la lista de publicaciones. No se permitirán publicaciones repetidas.
2. **Mostrar publicaciones**
   Imprime la lista de publicaciones del quiosco.
3. **Añadir una nueva publicación**
   Es posible añadir una nueva publicación con un stock inicial también a través del teclado. Esta opción pide al usuario toda la información necesaria para crear una nueva publicación. Una publicación se identifica por el nombre y no puede haber publicaciones repetidas.
4. **Eliminar una publicación**
   Se ejecuta esta opción cuando el quiosco cesa la venta de una publicación. El programa pregunta al usuario por el nombre de la publicación a borrar de la lista.
5. **Generar pedidos**
   La aplicación creará una lista con los pedidos necesarios para reponer. Cada pedido contendrá el nombre de la publicación y el número de ejemplares a pedir.
6. **Guardar pedidos en un fichero**
   Los pedidos generados serán guardados en un fichero de texto. Se le pide al usuario el nombre del fichero donde guardar la información.
7. **Importar publicaciones desde un fichero zip**
   Similar a la opción 1, pero esta vez carga la lista de publicaciones de un fichero comprimido .gz todas las publicaciones y elimina las anteriores.
8. **Exportar publicaciones en un fichero zip**
   Guarda en fichero comprimido .gz la lista de publicaciones.

Errores (excepciones)

Podrán existir diferentes fuentes de error: **errores de usuario**, incluyendo los del analizador de líneas (parsing errors), **errores del sistema** y **errores de programación**.

- **Errores del usuario**, por ejemplo, si indica un nombre de fichero erróneo, o se confunde al meter por teclado los datos de una publicación. También se consideran errores de usuario

los que pueda haber en el fichero de entrada (llamados parsing errors), por ejemplo, si la línea no tiene todos los campos que debiera o si el tipo de producto es desconocido.
- **Erres del sistema**, por ejemplo, los producidos por un mal funcionamiento de un dispositivo de entrada/salida.
- **Errores de programación**, por ejemplo, si los parámetros no son adecuados, y salta `IllegalArgumentException` o bien alguna otra excepción.

Si nosotros no manejamos tales condiciones de error, la aplicación podría terminar abruptamente.

Debemos manejar los errores que ocurren en la aplicación durante la ejecución y proporcionar mensajes al usuario final más manejables que los emitidos por el sistema. Es decir, debemos poder escribir código que pueda adaptarse a tales situaciones.

Se deben contemplar las siguientes situaciones de error:

1. Errores de Usuario

Deberán ser considerados los siguientes errores:

a) **Errores al analizar las líneas del fichero (parsing errors)**. Una línea inválida en un fichero es aquella que contiene campos inválidos al crear un objeto publicación. Por ejemplo, líneas con número incorrecto de campos, líneas con datos incorrectos en campos numéricos, con tipos de publicaciones desconocidas, líneas con frecuencias desconocidas. **No consideraremos un error que haya líneas en blanco, simplemente en este caso se ignora la línea y se continúa**.

b) **Añadir una publicación repetida**. Bien a partir de la opción 1 al cargar las publicaciones desde un fichero, o bien cuando se añade directamente a través de la opción 3, no estará permitido introducir una publicación con un nombre igual al de alguna que ya está en la lista.

c) **Leer o escribir en un fichero cuyo nombre tiene menos de 5 caracteres**.

d) **Borrar una publicación que no existe**.

e) **Leer de un fichero** que no existe

2. Errores de Programación y Sistema

Cualquier `RuntimeException` (o descendiente) lanzada durante la ejecución del programa (`IllegalArgumentException`, `NullPointerException`, `IndexOutOfBoundsException`, . .) será considerado un **error de programación**.

Entre los errores del sistema que vamos a detectar están los producidos por entrada/salida. Por ejemplo, si falla el dispositivo de lectura. En estos casos el sistema genera `IOException`. Pues bien, cualquier `IOException` (o descendiente) **excepto `FileNotFoundException`** debe considerarse un **error del sistema.**

Manejo de errores

Debemos manejar los errores que ocurren al ejecutar la aplicación y escribir mensajes de error fáciles de usar. Si no logramos manejar dichas condiciones de error, la aplicación podría finalizar abruptamente.

Para cada error, debemos decidir:

1. Cuándo y qué **excepción debería ser lanzada**

2. Dónde **recoger y tratar la excepción**
    - Las excepciones no comprobadas (RuntimeException), se recogen justo antes de finalizar el programa para que lo haga de manera controlada, y se tratan en un manejador común, evitando duplicación de código handleSystemError.
    - Las excepciones comprobadas, se recogen en el método a partir del cual queremos que el programa continúe. Puede ser en la interfaz para que vuelva a interaccionar con el usuario, o bien sin llegar a la interfaz, si hay algún proceso repetitivo de operaciones (por ejemplo, operación de carga de todos los objetos de una lista).

1. Errores de usuario

Lanza una nueva excepción en los siguientes casos:

a) Añadir una publicación repetida.

i) **Añadir publicación repetida a partir de un fichero.**
Lanzamiento. En la clase `Newsstand`, dentro de la operación `addPublication`, se deber comprobar antes de añadir una publicación, que ésta no esté ya en la lista. Si es así, se lanza `NewsstandException` con el mensaje: "`Publicación repetida: <nombre publicacion>`".
Manejo. Si esto sucede, la publicación se descarta y se debe enviar el mensaje de error a un archivo de registro (log), pero **el proceso debe continuar** cargando otras publicaciones hasta que el archivo de entrada se ha cargado por completo.

ii) **Añadir publicación repetida desde la opción 3 del menú.**
Lanzamiento. Esta opción del menú utiliza el mismo método `addPublication` que el caso anterior, puesto que se trata de realizar la misma función.
Manejo. Si esto sucede, la publicación se descarta, se muestra el mensaje de error en consola y se debe volver a presentar el menú al usuario para continuar operando.

b) Intentar leer o escribir nombres de ficheros con menos de 5 caracteres.
Lanzamiento. Este hecho se puede controlar en Newsstand (opciones loadFile (sesión 9), saveOrdersToFile(sesión 10), importFromZip (sesión 10) y exportToZip (sesión 10)). Lanzará una excepción `InvalidNameException` con el mensaje "`Nombre de fichero demasiado corto: <filename>`", que es una excepción no controlada de Java.
Manejo. Se mostrará en consola el mensaje de error, se grabará en el log el mensaje y se continúa mostrando el menú de nuevo para que el usuario continúe operando (como el resto de excepciones de usuario).

c) Intentar borrar una publicación que no existe. Si la publicación no existe, se lanzará `NewsstandException` con el mensaje "`Publicación inexistente: <datos publicación>`". Se mostrará por pantalla y se vuelve a presentar el menú de opciones para elegir una nueva opción.

d) Leer ficheros que no existen. Lanzamiento. Si el fichero no existe, el sistema lanza la excepción `fileNotFoundException` al intentar abrirlo. Esto sucede en la clase fileUtil (lo haremos en la sesión 10).
Manejo. Se mostrará en consola un mensaje de advertencia y se mostrará el menú de nuevo para que el usuario elija otra opción del menú.

Errores de análisis (parsing errors) mientas se carga el fichero

En este caso se lanza una excepción específica del parser, puesto que lo suyo es recoger el error, grabarlo en el log y continuar analizando líneas hasta la finalización del fichero.

e) **Lanzamiento**. Se lanza una nueva excepción personalizada, llamada `InvalidLineFormatException`, que almacene el número de la línea que causa el error y el tipo de error (
1. PALABRA CLAVE DESCONOCIDA,
2. NÚMERO NO VÁLIDO,
3. NÚMERO DE CAMPOS NO VÁLIDO,
4. FRECUENCIA NO VÁLIDA.

Manejo. El analizador debe continuar analizando líneas, pero escribe un mensaje en el fichero de log, ignora la línea y continúa. El mensaje para ser escrito en el fichero de log debe ser:

```
INVALID LINE line-number : message
```

Nota. No se considera error el hecho de recibir una línea en blanco. Simplemente se ignora, pero no hay que guardar mensaje alguno.

2. Errores del Sistema y programación

(a) `RuntimeException` (o descendientes)
Lanzamiento. Excepción controlada, podría ser lanzada en cualquier lugar.
Manejo. El programa debe finalizar la ejecución de manera controlada, informar al usuario de que ha habido un problema interno, enviar un mensaje de error al log y grabar en el log la traza de ejecución. Todas estas operaciones se realizan en un método manejador del sistema por defecto (`HandleSystemError`).

(b) `IOException` (o descendientes) excepto `FileNotFoundException`.
(c) **Lanzamiento**. En las clases `FileUtil.java` y `ZipFileUtil.java.` En las funciones de lectura y escritura se transformar esta excepción (salvo `FileNotFoundException`) en `RuntimeException` puesto que va a tratarse del mismo modo. Para ello se recoge y se lanza la nueva excepción.
(d) **Manejo**. El programa debe finalizar la ejecución de manera controlada, , informar al usuario de que ha habido un problema interno, enviar un mensaje de error al log y grabar en el log la traza de ejecución. Todas estas operaciones se realizan en un método manejador del sistema por defecto (`HandleSystemError`).

Puesta en funcionamiento del programa

En primer lugar, familiarízate con el código. Analiza el código y compáralo con el diagrama UML.

El proyecto inicial no contiene errores de compilación e incluso se puede ejecutar. Sin embargo, algunas opciones fallan con `NotYetImplementedException`. Intenta ejecutar opciones de la 1 a la 5.

1.1 Implementa las opciones del menú: 1 Cargar de un fichero y 2 mostrar publicaciones

1. Comienza implementando la opción 1 para cargar un fichero de texto en una lista. Completa la clase `PublicationParser` para transformar la lista de `<String>` devuelta por la implementación actual (falsa) de la clase `FileUtil` en una lista de publicaciones.

2. Completa el método `getPublications()` para probar el resultado de `loadFile`.

1.2 Implementa las opciones: 3 Añadir publicación, 4 Eliminar publicación y 5 Crear peticiones.

1.3 Añade el manejo de excepciones por defecto para errores del sistema

Primero, haz que tu programa finalice ordenadamente cuando hay un error de programación (o sistema) (RuntimeExcepcion), recoge la excepción y manéjala con un manejador del sistema por defecto, que muestre por pantalla un mensaje indicando que se ponga en contacto con el administrador, grabe el mensaje de error en el log, y grabe también en el log la pila de ejecución.
La gestión de estos errores se puede realizar en UserInterface:show. Ten en cuenta que una llegada de un error de este tipo debe hacer que el programa finalice, por lo que habrá que obligar a salir del bucle.

1.4 Añade el manejo de excepciones por defecto para errores del usuario

Si se produce algún error de usuario que implique volver a mostrar el menú de opciones, llegará hasta el método show una NewsstandException, En este caso habrá que recoger esta excepción y manejarla con un manejador de usuario por defecto, que muestre un mensaje al usuario por consola indicando el error y le proponga un nuevo intento sobre el menú de opciones.

1.5 Añade el manejo de excepciones al parser

Transforma alguna línea en el fichero de entrada en una línea incorrecta o en blanco. Por ejemplo, edita la clase `FileUtil.java` realiza alguna modificación y ejecuta de nuevo.

```
public List<String> readLines(String inFileName) {
    List<String> res = new LinkedList<>();

    res.add("newspaper  La Nueva España 14  30");
    res.add("news   La Nueva España 14  30");
    res.add("newspaper  La Nueva España 1M  30");
    res.add("newspaper  La Nueva España 14  3J");
    res.add("newspaper  La Nueva España 14");
    res.add("magazine   Hola    14  30  BIMONTH");
    res.add("magazine   Hola    30  BIMONTHLY");
    res.add("newspaper  El Mundo    4   10");
    res.add("magazine   Hola    14  30  BIMONTHLY");
    res.add("magazine   PCWord  14  30  QUARTERLY");
    res.add("magazine   Diez Minuntos   4   10  WEEKLY");
    res.add("magazine   El Mueble   4   10  MONTHLY");
    res.add("magazine   Muy Interesante 8   20  DAILY");
    res.add("    ");
    res.add("magazine   Quo 8   10  BIMONTHLY");

    return res;
}
```

Tarea sesión 9

Esta tarea debe estar implementada **24 horas antes** de la siguiente clase de laboratorio.

Antes de la entrega, renombra el proyecto dentro del IDE (usando Refactor- Rename) con:

`apellido1_apellido2_session9_task_newsstand`

Para entregar la tarea es necesario exportar el proyecto y comprimirlo en formato ZIP.

Tarea específica de esta sesión

1. Completa las **opciones del menú de la 1 a la 5**, en caso de que algo no esté terminado.

2. Filtra la entrada del usuario al añadir una nueva publicación a mano y deja que el usuario vuelva a intentarlo una vez más en caso de una entrada incorrecta.
   - Todos los valores numéricos deben estar en un rango [0-50]
   - Los valores de cadena no pueden estar vacíos ni nulos.
   - Los valores de cadena no pueden tener más caracteres que el máximo (por ejemplo 25) para cada campo.
   - La frecuencia debe corresponder con alguno de los valores existentes *(transforma la excepción de valueOf en NewsstandException)*

3. Añade test al método `PublicationParser.parse()` para los casos con periódico. Implementa casos positivos y negativos.

4. Añade test para el método `Newsstand.createOrders()`

- Revisa la sección "Manejo de errores" para comprobar qué casos hay que contemplar
- Usa `getOrders()` para probar las nuevas peticiones creadas. Recuerda devolver una copia de la lista.
- Escribe los casos positivos y negativos.
- **En el proyecto** `lab09_newsstand_starting_proyect` se proporciona un fichero `NewsstandCreateOrdersTest` con los casos que deberán ser implementados.

Escenarios:

Name	In-stock	Sold	Frequency	Order
Newspaper	10	5	-	No order
Newspaper	11	5	-	No order
Newspaper	5	10	-	Newspaper,20
Magazine	10	5	7	No order
Magazine	11	5	7	No order
Magazine	10	5	15	No order
Magazine	11	5	15	No order
Magazine	5	5	7	Magazine,5
Magazine	5	5	15	Magazine,10
Magazine	3	5	7	Magazine,20
Magazine	3	5	15	Magazine.20