Instrucciones

- Este ejercicio debe estar implementado 24 horas antes de la siguiente clase de laboratorio.
- Importa el lab09_newsstand_startingProject
- Renombra el proyecto dentro del Eclipse (usando Refactor- Rename) como sigue: **apellido1_apellido2_session9_task_newsstand**, en letras minúsculas y sin acentos
- Para entregar la tarea es necesario exportar el proyecto y comprimirlo en **formato ZIP.**

Objetivo de la tarea

Se describe 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 esta semana 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.

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. Existen tipificados los siguientes niveles de perioridad: MONTHLY, DIARY, QUATRIMESTRAL.

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. La sección previa.
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) **Añadir una publicación repetida**. Cuando se añade una nueva publicación repetida (bien a partir de la opción 1 al cargar las publicaciones desde un fichero, o bien cuando se crean a mano a través de la opción 3), no estará permitido introducir un nombre de publicación que ya está en la lista.

b) **Intentar leer de un fichero que no existe o cuyo nombre tiene menos de 5 caracteres**.
c) **Intentar borrar una publicación que no existe**.
d) **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 en blanco, líneas con número incorrecto de campos, líneas con datos incorrectos en campos numéricos, con tipos de publicaciones desconocidas…

2. Errores del sistema y programación

Un error de programación significa que el código tiene errores (tal vez por olvidarse de validar la entrada del usuario, escribir mal un nombre de variable, o algo así). Estas son cosas que siempre se pueden evitar cambiando el código.

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

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 la excepción**
    - Las excepciones no comprobadas, en un manejador común, evitando duplicación de código.
    - Las excepciones comprobadas, cerca del método que lanza la excepción, cuando tu estás en un método en el que sabes qué hacer al llegar la excepción.
3. Cómo **manejar la excepción**

1. Errores de usuario

Lanza una nueva excepción personalizada llamada **NewsstandException** en los 3 primeros tipos de excepciones de usuario definidos anteriormente, a), b) y c).

a) **Añadir una publicación repetida**.
    i) **Carga un fichero con una publicación repetida.**
    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".
    Manejo. Si esto sucede, la publicación se descarta y se debe enviar un mensaje de error a un archivo de registro (log), pero **el proceso debe continuar** hasta que el archivo de entrada se ha cargado por completo.
    ii) Añadir una 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 un mensaje de advertencia en consola y se debe volver a presentar el menú al usuario para continuar operando.

b) **Leer ficheros que no existen o intentar usar nombres de ficheros con menos de 5 caracteres.**
Lanzamiento. Si el fichero no existe, el sistema lanza la excepción al intentar abrirlo. Esto sucede en la clase fileUtil. Para el caso de tamaño del fichero de menos de 5 caracteres se producirá en UserInterface cuando se pida al usuario el nombre del fichero.
Manejo. Se mostrará en consola un mensaje de advertencia y se mostrará el menú de nuevo para que el usuario continúe operando.

c) **Intentar borrar una publicación que no existe**
Manejo. Se mostrará en consola un mensaje de advertencia y se mostrará el menú de nuevo para que el usuario continúe operando.

d) **Errores de análisis (parsing errors) mientas se carga el fichero**
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 (LÍNEA EN BLANCO, PALABRA CLAVE DESCONOCIDA, NÚMERO NO VÁLIDO, NÚMERO DE CAMPOS NO VÁLIDO.

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
```

2. Errores del Sistema y programación

(a) `RuntimeException` (o descendientes)
Lanzamiento. Exepción con 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 por defecto (HandleSystemError).

(b) `IOException` (o descendientes) excepto `FileNotFoundException`.
Lanzamiento. En `uo.mp2122.uit.file.FileUtil.java` y `uo.mp2122.util.file.ZipFileUtil.java`
Manejo. El programa debe finalizar la ejecución de manera controlada, informar al usuario de que ha habido un problema interno y enviar un mensaje de error al log.

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 la opción 3) Añadir publicación, 4) Eliminar publicación y 5) Crear peticiones

1.3 Añade el manejo de excepciones por defecto

Primero, haz que tu programa finalice ordenadamente cuando hay un error, recogiendo la excepción en lugar de simplemente fallar. Al principio, los manejadores predeterminados mostrarán mensajes (si es necesario) e imprimirán el seguimiento de la pila.

Ejecuta de nuevo.

1.4 Añade el manejo de excepciones al parser

Transforma una línea en el fichero de entrada en una línea incorrecta. Por ejemplo, edite uo.mp2122.util.file.FileUtil.java y cambia la primera línea por:

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

    res.add("newspaper  La Nueva España 14  30");
    res.add("newspaper  El Mundo    4   10");
    res.add("");
    res.add("newspaper  El Mundo    4");
    res.add("news   El Mundo    4   10");
    res.add("newspaper  El Mundo    4J  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("magazine   Quo 8   10  BIMONTHLY");

    return res;
}
```

ejecuta de nuevo

Mejores prácticas

- Declara las excepciones comprobadas particulares que puede lanzar tu método

```
public void foo() throws Exception { //Forma incorrecta
}
public void foo() throws NumberFormatException { //Forma correcta
}
```

- Lanza una excepción en el caso default de un witch.
- No ignores excepciones

```
catch(an exception) {
// ignore
}
```

Deberías, al menos, escribir un mensaje de registro (de log):

```
public void logAnException() {
        try {
                // hacer algo
        } catch (NumberFormatException e) {
                log.error("Esto nunca debería ocurrir: " + e);
        }
        }
```

- En una secuencia de captura, cateterizar primero las excepciones más específicas

```
  public void catchMostSpecificExceptionFirst() {
      try {
            doSomething("A message");
      } catch (NumberFormatException e) {
          log.error(e);
      } catch (IllegalArgumentException e) {
          log.error(e)
      }
}
```

- Registra (casi) todo.
- Recoge excepciones inesperadas que apagan la aplicación, escribiendo un mensaje cortés
- Transforma una excepción en otra. Cuando hagas eso, asegúrate de que en el mensaje que emites, queda clara la causa original del error.

```
public void wrapException(String input) throws MyBusinessException {
    try {
        // do something
    } catch (NumberFormatException e) {
        throw new MyBusinessException("A message that describes the error.", e);
    }
 }
```

- Durante la fase de desarrollo, deja que se produzcan excepciones. Escribe los controladores adecuados antes de la implementación.

Trabajo para casa Lab09

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

1. Cada parte de una aplicación debe escribir sus propias validaciones. Filtra la entrada del usuario al agregar 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.

2. Agrega test al método PublicationParser.parse (...) para los casos con periódico. Implementa casos positivos y negativos.

3. Agrega test para el método Newsstand.createOrders ().

- Revisa la sección 4 para comprobar qué casos hay que contemplar
- Usa getOrdersForTesting() para probar las nuevas peticiones creadas. Recuerda devolver una copia de la lista.
- Escribe los casos positivos y negativos
- **Dentro del material de esta sesión se proporciona un fichero 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