Universidad Tecnológica Nacional
Facultad Regional Avellaneda

RPP - LABORATORIO II

1r CUATRIMESTRE 2024

Requerimientos

En una biblioteca se van a escanear dos tipos de documento: mapas y libros. Necesitamos un programa que nos permita saber en qué paso del proceso tenemos cada uno de los documentos que se van a escanear y también sacar indicadores de tipo cuantitativo.

Los pasos del proceso, es decir, los estados en los que podemos encontrar los documentos (de cualquiera de los dos tipos) son los siguientes:

→ Inicio: el valor por defecto de los documentos.
→ Distribuido: el documento ya está en el escáner que le corresponde.
→ EnEscaner: el documento está siendo escaneado.
→ EnRevision: el documento está en el paso del proceso en el que se revisa si el escaneo no tuvo fallos (problemas de pixelado, páginas faltantes…).
→ Terminado: el documento ya ha sido escaneado y aprobado pues el proceso de revisión fue positivo.

El programa tendrá mecanismos de control para no cometer el error de escanear dos veces el mismo documento.

En el caso de libros, se considerará que son el mismo libro cuando:
→ Tenga el mismo barcode o
→ tenga el mismo ISBN o
→ tenga el mismo título y el mismo autor.

En el caso de los mapas se considerará que son el mismo mapa cuando:
→ Tenga el mismo barcode o
→ tenga el mismo título y el mismo autor y el mismo año y la misma superficie.

El escáner de los mapas (cama plana de gran formato) está en la mapoteca y es distinto al de los libros (escáner con alimentador de páginas) el cual está en la oficina de procesos técnicos.

Se precisa un módulo de informes que nos permita saber, teniendo en cuenta tipo de documento y estado, los siguientes datos:
→ Extensión de lo que hay en cada uno de los estados (número de páginas en el caso de libros, superficie en caso de mapas).
→ Cantidad de ítems en cada uno de los estados.
→ Datos de cada uno.

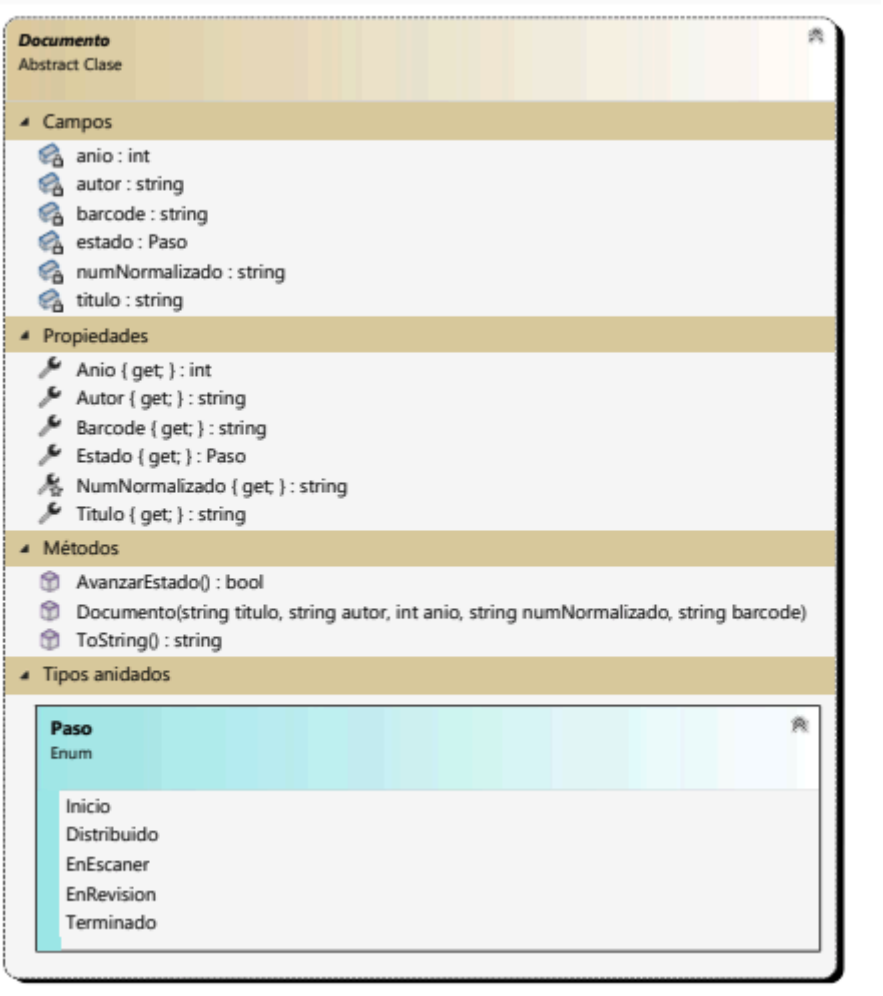
Esquema

La solución debe llamarse “PP_Escaner_ApellidoNombre” por ejemplo: *PP_Escaner_RusMataSilvia* y estar en NET 7.0.

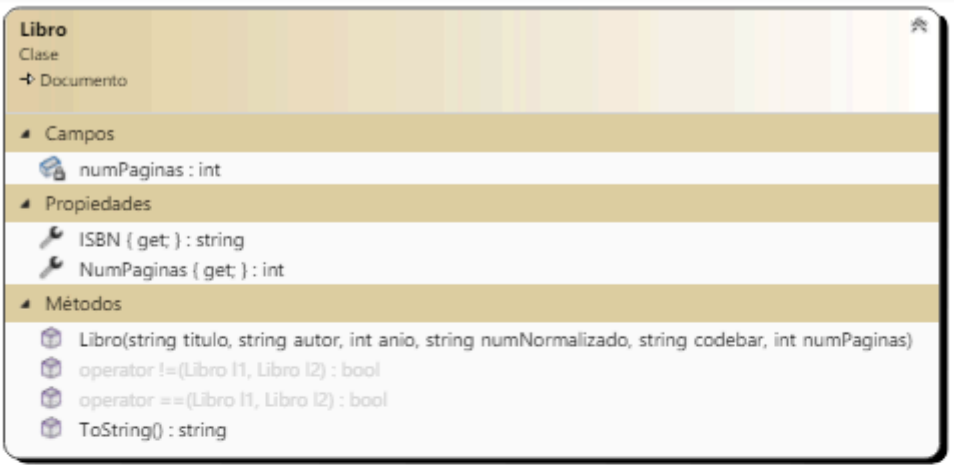
Y debe contener dos proyectos:
→ “Entidades” (Biblioteca de clases)
→ “Test” (Consola)

Biblioteca de clases

→ **Documento y derivados**



- La propiedad de NumNormalizado solo debe poder verse desde las clases derivadas.
- El estado debe inicializarse como “Inicio”.
- El método ToString() debe usar StringBuilder para mostrar todos los datos del documento.
- El método AvanzarEstado() debe pasar al siguiente estado dentro del orden que se estableció en el requerimiento. Debe devolver false si el documento ya está terminado.



- La sobrecarga del operador “==” debe funcionar según lo descripto en el requerimiento para cada uno de los dos tipos de documento.
- La propiedad ISBN en los libros muestra el NumNormalizado.
- Los mapas no tienen NumNormalizado.
- La superficie de los mapas se calcula multiplicando alto por ancho.
- El método ToString() debe mostrar todos los datos del documento en cuestión (los de la clase de la que heredan y los propios, incluida la superficie en el caso de los mapas). Usar StringBuilder.

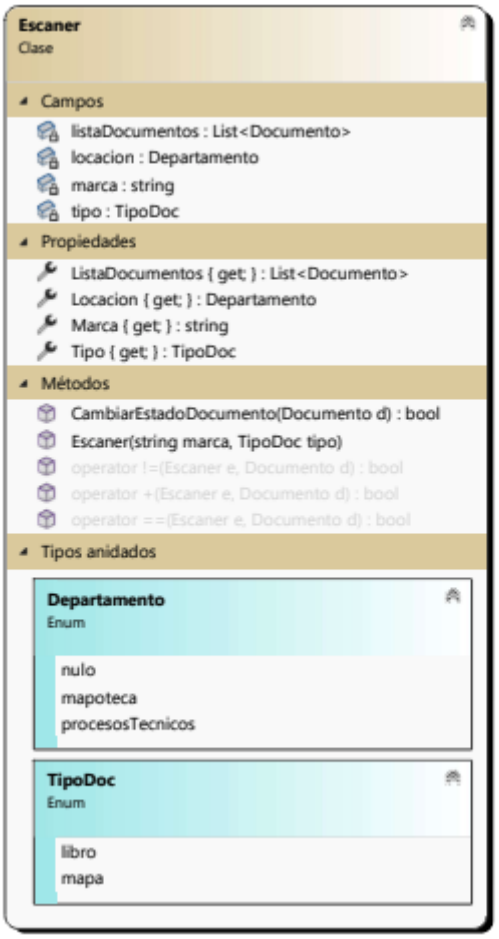
La información sobre los libros debe mostrarse así:

```
Titulo: Yerma
Autor: García Lorca, Federico
Año: 1995
ISBN: 1114
Cód. de barras: 22222
Número de páginas: 27.
```

La información sobre los mapas debe mostrarse así:

```
Autor: Instituto Geográfico de Mendoza
Año: 2008
Cód. de barras: 99990
Superficie: 100 * 30 = 3000 cm2.
```

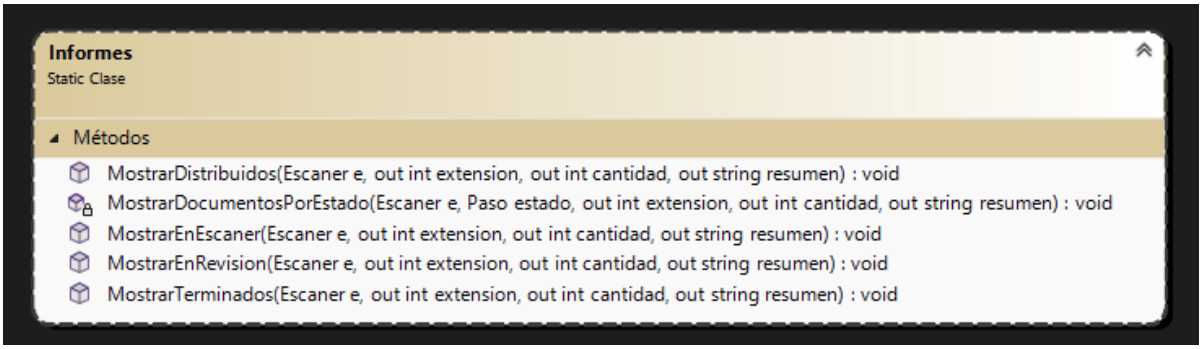
→ **Escáner**



- El constructor debe inicializar la lista de documentos.
- El constructor inicializa la locación según el tipo de documento a escanear (si es “mapa” la locación es “mapoteca” y si es “libro” va a “procesosTecnicos”).
- La sobrecarga del operador “==” comprueba si hay un documento igual en la lista. Devuelve true si encuentra, false si no.

- La sobrecarga del operador “+” añade el documento a la lista de documentos en el caso de que no haya un documento igual ya en ella. También debe añadirlo solo si está en estado “Inicio”. Antes de añadirlo a la lista debe cambiar el estado a “Distribuido”.
- El método “CambiarEstadoDocumento()” cámbiará el estado del documento de dentro de la lista de documentos.

→ **Informes**

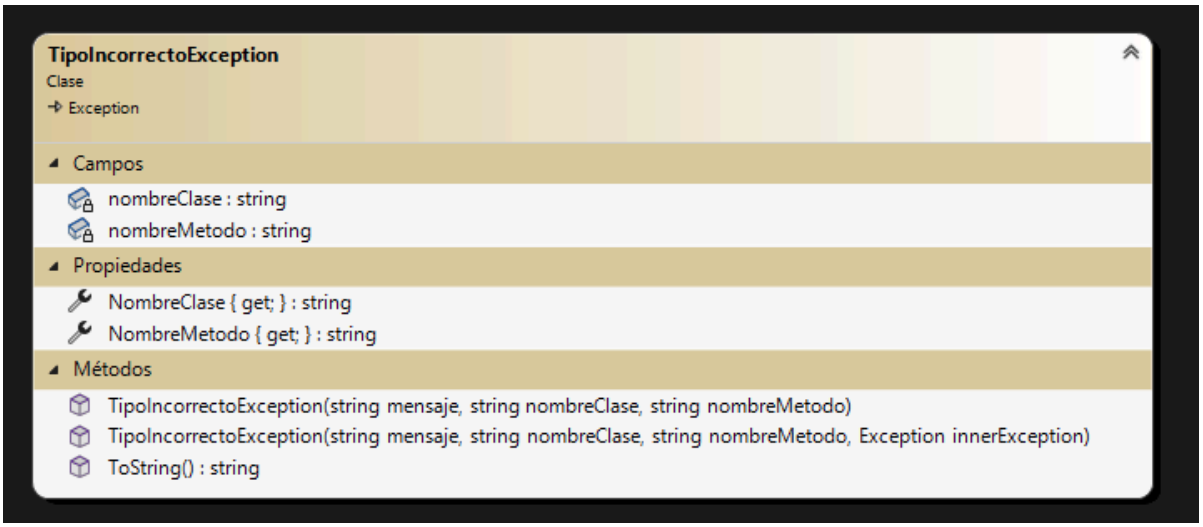


Cada uno de los informes públicos devolverá, dado un escáner y un estado en el que deban encontrarse los documentos tenidos en cuenta, los siguientes datos:

- extensión: el total de la extensión de lo procesado según el escáner y el estado. Es decir, el total de páginas en el caso de los libros y el total de cm2 en el caso de los mapas.
- cantidad: el número total de ítems únicos procesados según el escáner y el estado.
- resumen: se muestran los datos de cada uno de los ítems contenidos en una lista según el escáner y el estado.

Excepción

Se debe crear una excepción que se lance cuando se quiera añadir un libro al escáner de mapas o un mapa al escáner de libros. Debe tener la siguiente estructura:



- El método ToString() debe retornar el siguiente mensaje:
    - Excepción en el método (NOMBRE DEL MÉTODO) de la clase (NOMBRE DE LA CLASE).
    - Algo salió mal, revisa los detalles.
    - Detalles: (EL MENSAJE DE LA INNEREXCEPTION).
- La excepción TipoIncorrectoException será lanzada dentro de la sobrecarga de “==” de Escáner con el mensaje “Este escáner no acepta este tipo de documento”y capturada adentro de “+” donde se vuelve a lanzar como una nueva excepción con el mensaje “El documento no se pudo añadir a la lista”. Utilizar innerException para almacenar la excepción original.
- Utilizar el Main par agregar un mapa a una escáner de libros y un libro al escáner de mapas y mostrar el error por pantalla. Este código en el Main será evaluado también.

Consola

Usar para hacer el testeo necesario de las funcionalidad requerida. Será usada para hacer la corrección.

Criterios de corrección

El examen debe ser entregado mediante Github informado vía form. El día del examen la cátedra proporcionará un código para testear la aplicación mediante el proyecto de consola y la salida esperada.

Motivos de desaprobación directa (nota = 2):
- La solución no compila.
- La solución muestra warnings que no son del tipo:
    - *'class' defines operator == or operator != but does not override Object.Equals(object o)*
    - *'class' defines operator == or operator != but does not override Object.GetHashCode()*
- El último *commit* es posterior a 05/06/2024 a las 13:59.

Requisitos para llegar al 4:
- La salida por terminal después de ejecutar el código de testing proporcionado es similar en un 80% a la salida esperada.
- El código no registra errores graves tales como repetición de varias líneas de código.

Requisitos para llegar al 6:
- La salida por terminal después de ejecutar el código de testing proporcionado es entre un 81 y un 90% igual a la esperada.
- El código no registra errores graves tales como repetición de varias líneas de código.

Requisitos para llegar al 8:
- La salida por terminal después de ejecutar el código de testing proporcionado es entre un 91 y un 99% igual a la esperada.

- El código no registra errores graves tales como repetición de varias líneas de código.

Requisitos para llegar al 10:

- La salida por terminal después de ejecutar el código de testing proporcionado es entre un 100% igual a la esperada.
- El código no registra errores graves tales como repetición de varias líneas de código.