Ciências ULisboa | Informática

Desenvolvimento Centrado em Objetos

Trabalho Prático 1

2024/2025

0 Nota Prévia

O objetivo deste trabalho é que exercitem a utilização dos princípios de programação OO, padrões e técnicas lecionados na disciplina. Dada a natureza do trabalho e o contexto em que é realizado, não é considerado um bom instrumento para avaliar o que aprenderam e, por essa razão, não contribui muito para a nota final. No entanto, a realização deste trabalho é importante para atingirem diversos objetivos de aprendizagem e vai vos ajudar a desenvolver e melhorar diversas capacidades práticas essenciais a um engenheiro informático.

1 Introdução

O trabalho consiste numa aplicação para organizar recursos de leitura digitais, com muitas das funcionalidades que se encontram generalizadas em aplicações deste género.

A aplicação, a que foi dado o nome de **LEIBooks**, vai ser desenvolvida incrementalmente e iterativamente. Isto significa que a aplicação vai ser construída em torno de pequenas partes funcionais (*incrementos*) e, para cada incremento, vão ser realizadas várias iterações em que se vai alargando sucessivamente o conjunto dos casos de uso suportados e vão sendo incorporadas melhorias e feedback dos clientes nos casos de uso já implementados.

O primeiro incremento do LEIBooks foca-se num conjunto de funcionalidades básicas em torno da biblioteca de documentos e de prateleiras. A primeira iteração deste primeiro incremento foca-se num pequeno conjunto de casos de uso básicos, descritos de seguida.

Considerou-se que a aplicação deve permitir carregar documentos guardados localmente em disco para a biblioteca *(Library)* e eventualmente depois apagá-los da biblioteca. É possível procurar documentos, ler os documentos da biblioteca, colocar marcas nas suas páginas (para posteriormente facilitar a navegação dentro do documento) e também fazer anotações. A aplicação oferece a possibilidade de criar prateleiras para melhor organizar os documentos. As prateleiras podem ser normais ou inteligentes. Nas primeiras a adição e remoção dos documentos é feita explicitamente pelo utilizador, enquanto nas segundas a arrumação é feita automaticamente tendo em conta o critério com que foram definidas. Foi decidido que existem algumas prateleiras inteligentes pré-definidas (e não apagáveis) como *Bookmarked*, com apenas os documentos com marcas, e *Recents*, com apenas os documentos mais recentes.

Decidiu-se não endereçar por enquanto aspetos de persistência, ou seja, não é guardada memória de nenhuma ação realizada na aplicação. Por outro lado, tendo em conta a natureza e formato diverso dos documentos que se pretendem suportar, considerou-se dar importância a este aspeto e ao impacto que ele tem em diversos elementos da aplicação, desde a visualização e navegação nos documentos até à obtenção de metadados.

2 Solução de Desenho

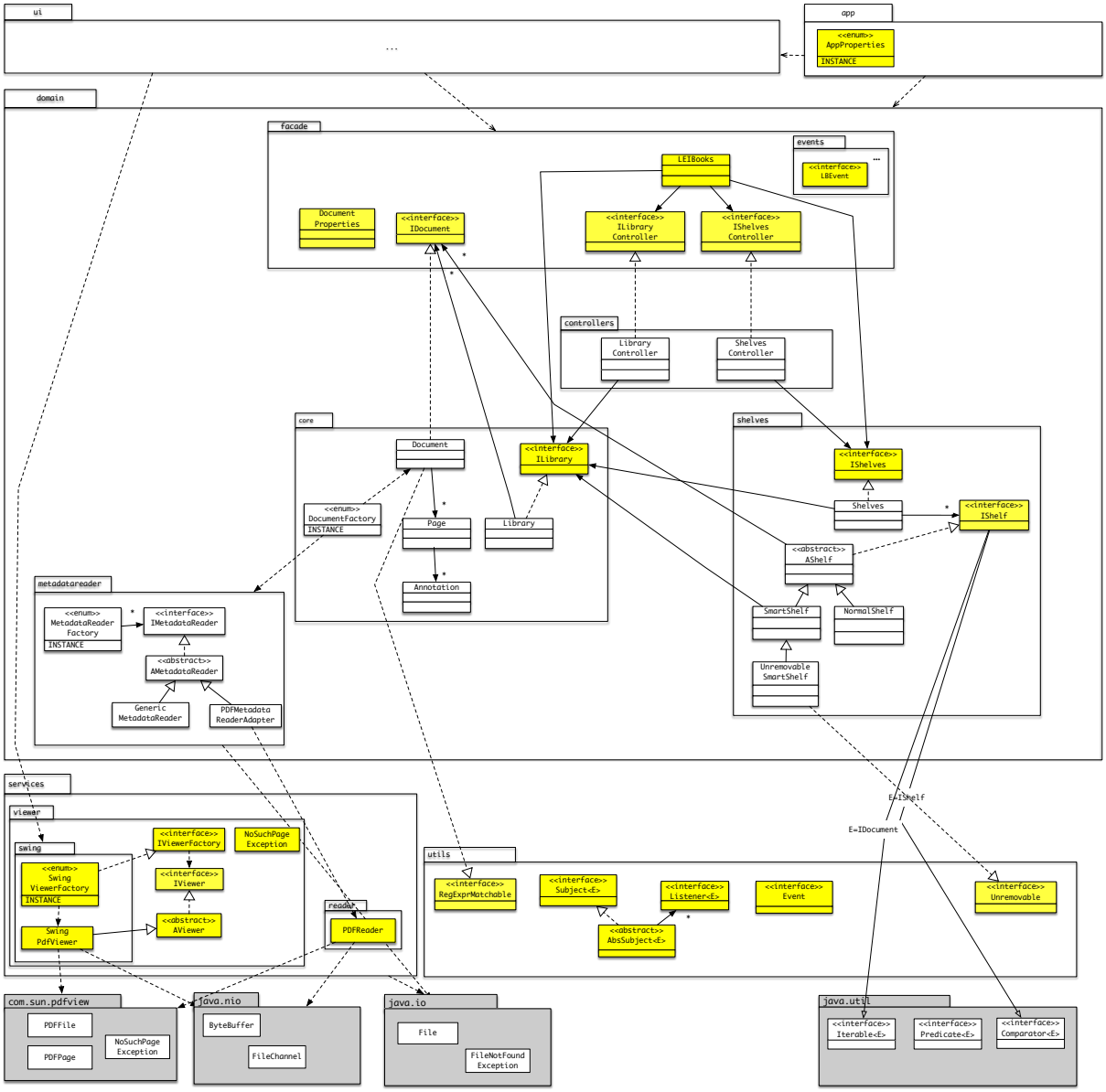
Nesta seção apresentam-se os detalhes mais importantes sobre a solução de desenho que foi desenvolvida para a primeira iteração. O código da aplicação está organizado em cinco pacotes:

- **ui**: com classes responsáveis exclusivamente pela interface da aplicação
- **domain**: com classes cuja responsabilidade se prende apenas com o domínio da aplicação
- **app**: com classes responsáveis exclusivamente pela configuração da aplicação e de estabelecimento da ligação da UI ao domínio

- **services**: com classes cuja responsabilidade se prende com a utilização de serviços externos, por exemplo os serviços que permitem obter os metadados dos documentos e visualizadores de páginas
- **utils**: com classes que foram concebidas especificamente para a aplicação, mas potencialmente reutilizáveis noutros contextos

Porque o foco de DCO não é o desenho da UI, é fornecida uma implementação rudimentar de uma interface gráfica que suporta a maior parte das funcionalidades cobertas nesta iteração. A implementação desta interface foi realizada com o *Swing* e encontra-se no pacote ui. Não é necessário analisar nem fazer qualquer alteração a esse código, mas convém saber que:

- O funcionamento da interface gráfica é baseado não só em chamadas aos métodos dos dois tipos controladores disponibilizados pelo domínio, mas também na observação dos eventos emitidos por estes controladores.
- A aplicação tem vários elementos que são configuráveis, muitos deles respeitando a interface gráfica. A solução baseia-se na utilização de um ficheiro de propriedades com as configurações a usar, que é lido no arranque da aplicação.



O diagrama acima dá uma visão geral da estrutura de classes do resto da solução. Os nomes de atributos e algumas relações foram omitidos de forma a promover a legibilidade do diagrama. Os elementos a amarelo já foram implementados e a sua implementação é fornecida. Detalhes adicionais sobre a solução de desenho são descritos no que se segue.

Pacote `utils`

Este pacote atualmente tem essencialmente elementos que são úteis para aplicar o padrão observador[1].

- Interface **`Event`** representando objetos que são eventos
- Interface **`Listener<E>`** representando objetos que se registam para receber certos eventos de objetos **`Subject`**
- Interface **`Subject<E>`** representando objetos que emitem eventos para os seus *listeners*
- Classe abstrata **`AbsSubject<E>`** com uma implementação esqueleto de **`Subject<E>`**
- Interface **`RegExpMatchable`** representando objetos que suportam o emparelhamento (*match*) com uma expressão regular
- Interface **`Unremovable`** que serve para marcar tipos de objetos que não podem ser removidos das coleções onde são colocados

Pacote `services`

Este pacote contém as classes que fornecem serviços de baixo nível à aplicação. Atualmente tem dois pacotes:

- **`viewer`** com as classes que prestam serviços de obtenção das páginas dos documentos para visualização na UI. Porque o *toolkit* usado na construção do UI condiciona a forma como estes serviços são fornecidos, existe uma implementação desses serviços compatível com o *Swing*. Esta implementação consiste num visualizador para documentos PDF (implementada recorrendo à biblioteca *pdfview* da IBM) e no carregamento dinâmico de outros visualizadores que venham a estar disponíveis aquando da execução da aplicação.
- **`reader`** com as classes que prestam serviços de leitura de metadados para documentos de diferentes naturezas e em diferentes formatos. Atualmente apenas contém uma classe para documentos em pdf (também implementada recorrendo ao *pdfview*).

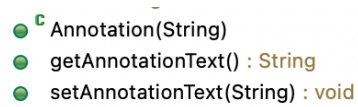
Pacote `domain`

Este pacote subdivide-se em cinco pacotes:

- **`domain.core`** –– tem os elementos base do domínio
- **`domain.shelves`** –– tem os elementos do domínio relacionados com prateleiras
- **`domain.metadatareader`** –– tem os elementos do domínio relacionados com a leitura de metadados
- **`domain.controllers`** –– tem os elementos que controlam as várias formas de interagir com a aplicação
- **`domain.facade`** –– tem os elementos que constituem a interface que o **`domain`** expõe aos pacotes clientes

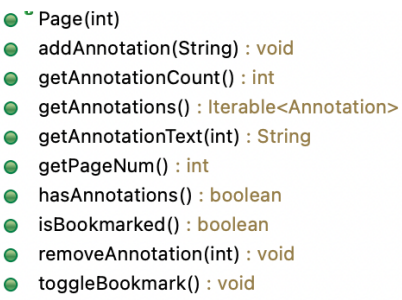
`domain.core`

- Classe **`Annotation`** cujos objetos, mutáveis, representam uma anotação da página de um documento. A classe deve ter os membros mostrados ao lado.



[1] Semelhante ao que existe no *java.util* mas que está *deprecated*. Apesar das limitações que esta solução apresenta, considera-se que, para efeitos pedagógicos neste trabalho, é mais apropriado que as alternativas existentes.

- Classe **`Page`** cujos objetos, mutáveis, representam uma página de um documento. Cada página regista o seu número, as suas anotações e se está ou não marcada. As anotações são numeradas sequencialmente e identificadas pelo seu número. A classe deve ter os membros mostrados ao lado.



- Classe **`Document`** cujos objetos, mutáveis, representam documentos. A classe deve implementar **`IDocument`** e **`RegExpMatchable`**, estender **`AbsSubject<DocumentEvent>`** e oferecer o seguinte construtor:

```
Document (String title, String author, LocalDateTime
dateModified, String mimeType, String pathToFile,
Optional<Integer> numberOfPages)
```

  A igualdade de documentos e o *compareTo* deve ser unicamente baseada no ficheiro do documento em disco. Para o *match* deve ser usado o **título e autor**.

- Enumerado **`DocumentFactory`** cuja única instância oferece um criador de documentos com assinatura

```
IDocument createDocument(String title, String pathToPhotoFile) throws
java.io.FileNotFoundException
```

  recorrendo à utilização de um objeto **`IMetadataReader`** para obter os metadados do ficheiro que deve ser obtido da única instância de **`MetadataReaderFactory`**

- Interface **`ILibrary`** cujos objetos representam bibliotecas de documentos, iteráveis. Estende **`Subject<DocumentEvent>`** e notifica os seus *listeners* quando um documento é adicionado ou removido através da emissão de eventos apropriados.
- Classe **`Library`** que implementa **`ILibrary`** estendendo **`AbsSubject<DocumentEvent>`**.

`domain.shelves`

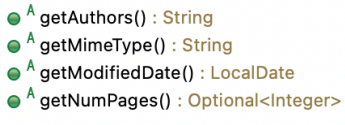
- Interface **`IShelf`** que representa prateleiras, onde podem ser colocados documentos da biblioteca. Uma prateleira é iterável, tem um nome único e não tem documentos repetidos. Estende **`Listener<DocumentEvent>`** de forma a poder reagir a mudanças na biblioteca.
- Classe abstrata **`AShelf`** com uma implementação esqueleto de **`IShelf`** baseada num atributo com o nome da prateleira, com construtor **`AShelf (String name)`**.
- Classe **`NormalShelf`** que estende **`AShelf`** e cujos objetos representam prateleiras normais, guardando os documentos que contêm. Documentos apagados da biblioteca que estavam na prateleira, deixam de lá estar.
- Classe **`SmartShelf`** que estende **`AShelf`** e cujos objetos representam prateleiras inteligentes, criadas com um nome, uma biblioteca e um critério **`Predicate<IDocument>`** com construtor

```
SmartShelf (String name, ILibrary lib, Predicate<IDocument> criteria)
```

  É o critério fornecido na construção que determina que documentos da biblioteca a prateleira tem. Estas prateleiras não suportam as operações de adição e remoção de documentos.

- Classe **`UmremovableSmartShelf`** que estende **`SmartShelf`** e implementa **`UnRemovable.`** Representa prateleiras inteligentes que não podem ser removidas.
- Interface **`IShelves`** que representa um catálogo de prateleiras, iterável. Estende **`Subject<ShelfEvent>`** e notifica os seus *listeners* quando uma prateleira é adicionada ou removida, ou um documento é removido de uma prateleira, através da emissão de eventos apropriados.
- Classe **`Shelves`** que implementa **`IShelves`** estendendo **`AbsSubject<ShelfEvent>`** e tem a biblioteca a que as prateleiras estão associadas e usa um mapa para guardar as prateleiras existentes.

`domain.metadatareader`



- interface **`IMetadataReader`** que representa leitores de metadados de documentos guardados em disco num ficheiro, oferecendo os métodos listados ao lado. Estes métodos nunca retornam *null*.

  É um objeto deste tipo, obtido através de **`MetadataReaderFactory`**, que é usado para ler os metadados.

- classe abstrata **`AMetadataReader`** que dá uma implementação esqueleto de **`IMetadataReader`** e que 1) tem um construtor **`AMetadataReader(String pathToDocFile) throws FileNotFoundException`** que obtém

a data de modificação e o *mimeType* diretamente do ficheiro lançando a exceção quando há problemas com o acesso ao mesmo e 2) declara atributos `Optional<Integer> numPages` e `String authors` como `protected` e define os seus valores *default* como sendo `Optional.empty()` e "n/a".

- classe `GenericMetadataReader` que estende `AMetadataReader` e que usa os valores *default* para o número de páginas e os autores.
- classe `PDFMetadataReaderAdapter` cujos objetos são leitores de documentos em ficheiros PDF, estendendo `AMetadataReader` e usando os serviços em `services.reader.PDFReader`.
- classe `MetadataReaderFactory` que é um *singleton* cuja instância tem a responsabilidade de dar um leitor `MetadataReader` apropriado para ler os metadados do documento guardado num ficheiro em disco através do método

```
IMetadataReader createMetadataReader (String pathToDocFile) throws
FileNotFoundException
```

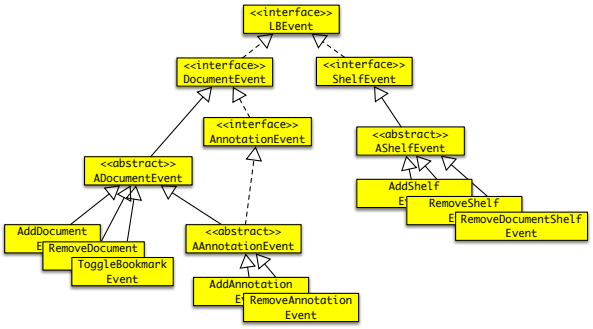
Esta instância obtém do ficheiro o seu *mimetype* e cria um objeto de um leitor apropriado para esse *mimetype*. Os tipos apropriados (classes subtipos de `IMetadataReader`) são carregados num mapa na construção da instância. Este mapa deve ter `PDFMetadataReaderAdapter.class` associado a *"application/pdf"*. Outras classes encontradas numa determinada diretoria devem ser carregadas dinamicamente, considerando a convenção de que uma classe `AbcXyzMetadataReader` é um leitor para *"abc/xyz"*. A diretoria onde os leitores extra são procurados é configurável (ver `AppProperties`)

`domain.controllers`

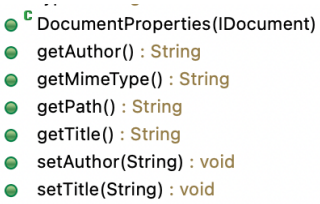
- Classe `LibraryController` que implementa `ILibraryController` estendendo `AbsSubject<LBEvent>` e com construtor `LibraryController (Library library)`
- Classe `ShelvesController` que implementa `IShelvesController` estendendo `AbsSubject<LBEvent>` e com construtor `ShelvesController (IShelves shelves)`

`domain.facade`

- `domain.core.events` pacote com uma hierarquia de subtipos de `leibboks.utils.Event` representando os eventos emitidos pelos objetos do domínio. A raiz é `LBEvent` que é um evento genérico, admitindo especializações que representam eventos sobre a biblioteca e seus documentos e sobre as prateleiras



- Interface `IDocument` que representa os documentos. Estende `Subject<DocumentEvent>` e notifica os seus observadores quando são adicionadas ou removidas anotações às suas páginas ou marcas, através da emissão de eventos apropriados.
- Classe `DocumentProperties` cujos objetos são *data transfer objects* para as propriedades textuais dos documentos. Oferece construtor e métodos mostrados ao lado.



- Interface `ILibraryController` que representa controladores das interações com a biblioteca. Estende `Subject<LBEvent>` e `Listener<DocumentEvent>`, propagando para o exterior do domínio os eventos relativos aos documentos na biblioteca.
- Interface `IShelvesController` que representa controladores das interações com as prateleiras Estende `Subject<LBEvent>` e `Listener<DocumentEvent>`, propagando para o exterior do domínio os eventos relativos às prateleiras.
- Classe `LEIBooks` que fornece o objeto inicial do sistema

3 O que há a fazer?

O trabalho tem duas partes. A primeira parte consiste em implementar a solução fornecida baseada no código dado de forma a obter uma versão funcional da aplicação. A segunda parte consiste em adaptar a solução de desenho fornecida de forma a cobrir mais alguns requisitos funcionais.

3.1 Implementação da Solução de Desenho Fornecida

É preciso tomar em atenção que a solução de desenho descrita não é suficientemente detalhada para que não tenham de tomar ainda decisões ao nível da implementação (por exemplo, a acessibilidade das classes, a factorização de código, atributos necessários e seus tipos, a redefinição apropriada dos métodos herdados de Object, etc). Na implementação dos métodos *toString* devem usar a representação textual mostrada no exemplo em anexo. Relativamente a tudo o que não estiver definido, devem tomar as decisões que considerarem mais adequadas. O código entregue deve estar convenientemente documentado com **JavaDoc**. É boa ideia pedir a ajuda do *copilot* nesta tarefa, mas revejam o resultado.

Em relação ao processo de implementação:

- Devem usar o plugin ***SonarLint***[2] para vos ajudar a controlar a qualidade do vosso código (o plugin existe para vários IDEs como o Eclipse, VSCode e IntelliJ).
- Tendo em vista facilitar o teste do código que vão desenvolver, é fornecido um programa cliente que exercita algumas funcionalidades da aplicação e a interface gráfica. Isto, porém, não dispensa a escrita de testes. São vos fornecidas já duas classes com alguns testes unitários ***JUnit*** para as classes `domain.core.Document` e `domain.core.Library` assim como umas classes auxiliares. Devem completar estas classes com mais testes e fazer testes unitários para mais uma classe à vossa escolha.
- Devem usar o sistema de controlo de versões ***Git***, para vos ajudar no desenvolvimento cooperativo. Dividam as classes a programar pelos dois elementos do grupo. Cada um deve ser responsável por fazer o *code review* das classes programadas pelo outro. Cada um dos elementos do grupo deve tornar a sua atividade no trabalho visível no repositório pois esta informação será também usada para aferir a contribuição individual de cada elemento do grupo.

3.2 Adaptação da Solução de Desenho Fornecida

Porque a solução de desenho fornecida cobre apenas um pequeno conjunto de requisitos funcionais, pretende-se que façam evoluir a solução de desenho de forma a cobrir um conjunto adicional de requisitos:

– Pretende-se suportar *Folder Shelves*, ou seja, prateleiras que possam conter outras prateleiras, além de documentos.
– Pretende-se suportar a ligação a uma *Book Store* eletrónica. Ainda não se sabe qual será escolhida, mas será uma livraria que disponibiliza livros gratuitos. Deve ser possível inspecionar os livros que a loja disponibiliza gratuitamente e importar livros destes para a biblioteca local.

Devem desenhar as alterações que seriam necessárias fazer à solução fornecida. Sugere-se que usem excertos do diagrama de classes para comunicar quais são as mudanças, acompanhados das explicações que acharem necessárias.

Implementar estas alterações **não** faz parte do trabalho.

4 Por onde começar?

Devem importar o projeto fornecido e analisá-lo. Podem fazer a importação do projeto tanto no Eclipse como no VSCode; no VSCode importem de forma que a pasta do projeto seja, para efeitos do IDE, a raiz. Comecem por ler o README do projeto.

Um dos elementos do grupo de trabalho deverá criar um repositório git com o nome *leibooks_XXXXX_YYYYY*, onde *XXXXX* e *YYYYY* são os números dos membros do grupo, e definir a visibilidade do projeto como **privada**. Além de dar acesso ao colega de grupo com o nível *Maintainer*, deve adicionar à lista de membros do projeto o utilizador **dco000** com o nível de *Reporter*.

A relação de dependência entre as classes define naturalmente uma ordem de implementação: primeiro desenvolvem-se as classes sem dependências, depois as que dependem destas, e assim sucessivamente.

[2] https://www.sonarsource.com/products/sonarlint/

5 Como e quando entregamos?

Identifiquem o *commit* de entrega com **`git tag entrega`** e coloquem essa identificação no servidor (**`git push origin entrega`**). O *deadline* para entrega é **10 de Abril**.

O repositório deve conter:

1. o código fonte do vosso projeto (1ªparte do trabalho);
2. um documento pdf com a descrição da adaptação da solução de desenho (2ªparte do trabalho)

Na aula teórica de dia **11 de Abril** será realizado o teste de aferição individual, que é obrigatório.

6 Critérios de Correção

Para ajudar a guiar o vosso esforço indicam-se alguns critérios de correção que vamos usar:

- Se o código compila e o programa cliente fornecido executa até ao fim.
- Em que medida o comportamento do sistema, em cada cenário de teste, é o esperado (isto é feito através da execução do programa cliente fornecido e da execução de testes automáticos).
- Em que medida a organização e legibilidade do vosso código segue as boas práticas e está de acordo com o que vos foi fornecido.
- Em que medida as classes e comportamentos definidos estão de acordo com o que vos é pedido: cada classe deve suportar os métodos descritos e com os parâmetros adequados, o comportamento dos métodos dever ser o indicado.
- Em que medida as decisões sobre acessibilidade das classes e fatorização do código são adequadas e estão de acordo com o princípio do *Information Hiding*.
- Em que medida os métodos herdados de *Object* foram redefinidos de forma apropriada.
- Em que medida a documentação dos tipos é adequada, nomeadamente se há contratos a descrever o comportamento dos construtores e métodos acessíveis.
- Em que medida os testes unitários *JUnit* pedidos foram realizados, seguem as melhores práticas e têm uma cobertura adequada das funcionalidades esperadas.
- Em que medida o documento que apresentaram permite compreender como adaptariam a solução de desenho para os requisitos adicionais e a correção e bondade dessas decisões.

ANEXO

O output indicativo do *SimpleClient* fornecido abaixo foi obtido usando a configuração fornecida. É importante notar que algumas datas são relativas à data de execução do programa.

```
------------------------------------------------
Swing Viewers available
------------------------------------------------
{image/jpegswing=class SwingImageViewer, image/jpgswing=class SwingImageViewer, image/pngswing=class SwingImageViewer,
application/pdfswing=class leibooks.services.viewer.swing.SwingPDFViewer, image/gifswing=class SwingImageViewer}
------------------------------------------------
Metadata readers available
------------------------------------------------
{application/pdf=class leibooks.domain.metadatareader.PDFMetadataReaderAdapter, image/jpeg=class ImageJpegMetadataReader}
------------------------------------------------
Load photos and add them to library
------------------------------------------------
-------->> Library: AddDocumentEvent [document=doc_files/Apresentacao.pdf]<<--------
-------->> Library: AddDocumentEvent [document=doc_files/Aula2.pdf]<<--------
-------->> Library: AddDocumentEvent [document=doc_files/TLXScale.pdf]<<--------
-------->> Library: AddDocumentEvent [document=doc_files/tulips2.jpg]<<--------
-------->> Library: AddDocumentEvent [document=doc_files/SummerFades.jpg]<<--------
-------->> Library: AddDocumentEvent [document=doc_files/ScarletTown.txt]<<--------
File doc_files/MissingFile.jpg not found or could not be open
Library =
Document{title=Aula01, author=Antonia Lopes, file=doc_files/Apresentacao.pdf, date=2025-02-18, mimeType=application/pdf,
numPages=Optional[12], lastPageVisited=0, pages={}}
Document{title=Aula02, author=Antonia Lopes, file=doc_files/Aula2.pdf, date=2025-02-19, mimeType=application/pdf, numPages=Optional[13],
lastPageVisited=0, pages={}}
Document{title=Lyrics Scarlet Town, author=n/a, file=doc_files/ScarletTown.txt, date=2025-02-20, mimeType=text/plain,
numPages=Optional.empty, lastPageVisited=0, pages={}}
Document{title=Flor, author=, file=doc_files/SummerFades.jpg, date=2011-06-09, mimeType=image/jpeg, numPages=Optional[1],
lastPageVisited=0, pages={}}
Document{title=Scale, author=n/a, file=doc_files/TLXScale.pdf, date=2025-02-18, mimeType=application/pdf, numPages=Optional[1],
lastPageVisited=0, pages={}}
Document{title=Tulipas, author=, file=doc_files/tulips2.jpg, date=2011-06-09, mimeType=image/jpeg, numPages=Optional[1],
lastPageVisited=0, pages={}}
Shelves=
Recent= [doc_files/Apresentacao.pdf, doc_files/Aula2.pdf, doc_files/ScarletTown.txt, doc_files/TLXScale.pdf]
Bookmarked= []
------------------------------------------------
Adding bookmarks to
doc_files/Apresentacao.pdf
doc_files/Aula2.pdf
------------------------------------------------
-------->> Document: ToggleBookmarkEvent [document=doc_files/Apresentacao.pdf pageNum=1 isBookmarked=true]<<--------
-------->> Document: ToggleBookmarkEvent [document=doc_files/Aula2.pdf pageNum=1 isBookmarked=true]<<--------
Library =
Document{title=Aula01, author=Antonia Lopes, file=doc_files/Apresentacao.pdf, date=2025-03-10, mimeType=application/pdf,
numPages=Optional[12], lastPageVisited=0, pages={1=Page{bookmark=true, annotations=[], pageNum=1}}}
Document{title=Aula02, author=Antonia Lopes, file=doc_files/Aula2.pdf, date=2025-03-10, mimeType=application/pdf, numPages=Optional[13],
lastPageVisited=0, pages={1=Page{bookmark=true, annotations=[], pageNum=1}}}
Document{title=Lyrics Scarlet Town, author=n/a, file=doc_files/ScarletTown.txt, date=2025-02-20, mimeType=text/plain,
numPages=Optional.empty, lastPageVisited=0, pages={}}
Document{title=Flor, author=, file=doc_files/SummerFades.jpg, date=2011-06-09, mimeType=image/jpeg, numPages=Optional[1],
lastPageVisited=0, pages={}}
Document{title=Scale, author=n/a, file=doc_files/TLXScale.pdf, date=2025-02-18, mimeType=application/pdf, numPages=Optional[1],
lastPageVisited=0, pages={}}
Document{title=Tulipas, author=, file=doc_files/tulips2.jpg, date=2011-06-09, mimeType=image/jpeg, numPages=Optional[1],
lastPageVisited=0, pages={}}
Shelves=
Recent= [doc_files/Apresentacao.pdf, doc_files/Aula2.pdf, doc_files/ScarletTown.txt, doc_files/TLXScale.pdf]
Bookmarked= [doc_files/Apresentacao.pdf, doc_files/Aula2.pdf]
------------------------------------------------
Adding annotations to
doc_files/Apresentacao.pdf
------------------------------------------------
-------->> Document: AddAnnotationEvent [document=doc_files/Apresentacao.pdf pageNum=8 annNum=0 annotationText=CHECK MOODLE ALSO]<<------
--
Library =
Document{title=Aula01, author=Antonia Lopes, file=doc_files/Apresentacao.pdf, date=2025-03-10, mimeType=application/pdf,
numPages=Optional[12], lastPageVisited=0, pages={1=Page{bookmark=true, annotations=[], pageNum=1}, 8=Page{bookmark=false,
annotations=[Annotation [text=CHECK MOODLE ALSO]], pageNum=8}}}
Document{title=Aula02, author=Antonia Lopes, file=doc_files/Aula2.pdf, date=2025-03-10, mimeType=application/pdf, numPages=Optional[13],
lastPageVisited=0, pages={1=Page{bookmark=true, annotations=[], pageNum=1}}}
Document{title=Lyrics Scarlet Town, author=n/a, file=doc_files/ScarletTown.txt, date=2025-02-20, mimeType=text/plain,
numPages=Optional.empty, lastPageVisited=0, pages={}}
Document{title=Flor, author=, file=doc_files/SummerFades.jpg, date=2011-06-09, mimeType=image/jpeg, numPages=Optional[1],
lastPageVisited=0, pages={}}
Document{title=Scale, author=n/a, file=doc_files/TLXScale.pdf, date=2025-02-18, mimeType=application/pdf, numPages=Optional[1],
lastPageVisited=0, pages={}}
Document{title=Tulipas, author=, file=doc_files/tulips2.jpg, date=2011-06-09, mimeType=image/jpeg, numPages=Optional[1],
lastPageVisited=0, pages={}}
Shelves=
Recent= [doc_files/Apresentacao.pdf, doc_files/Aula2.pdf, doc_files/ScarletTown.txt, doc_files/TLXScale.pdf]
Bookmarked= [doc_files/Apresentacao.pdf, doc_files/Aula2.pdf]
------------------------------------------------
Delete
doc_files/TLXScale.pdf
------------------------------------------------
-------->> Library: RemoveDocumentEvent [document=doc_files/TLXScale.pdf]<<--------
```

```
Library =
Document{title=Aula01, author=Antonia Lopes, file=doc_files/Apresentacao.pdf, date=2025-03-10, mimeType=application/pdf,
numPages=Optional[12], lastPageVisited=0, pages={1=Page{bookmark=true, annotations=[], pageNum=1}, 8=Page{bookmark=false,
annotations=[Annotation [text=CHECK MOODLE ALSO]], pageNum=8}}}
Document{title=Aula02, author=Antonia Lopes, file=doc_files/Aula2.pdf, date=2025-03-10, mimeType=application/pdf, numPages=Optional[13],
lastPageVisited=0, pages={1=Page{bookmark=true, annotations=[], pageNum=1}}}
Document{title=Lyrics Scarlet Town, author=n/a, file=doc_files/ScarletTown.txt, date=2025-02-20, mimeType=text/plain,
numPages=Optional.empty, lastPageVisited=0, pages={}}
Document{title=Flor, author=, file=doc_files/SummerFades.jpg, date=2011-06-09, mimeType=image/jpeg, numPages=Optional[1],
lastPageVisited=0, pages={}}
Document{title=Tulipas, author=, file=doc_files/tulips2.jpg, date=2011-06-09, mimeType=image/jpeg, numPages=Optional[1],
lastPageVisited=0, pages={}}
Shelves=
Recent= [doc_files/Apresentacao.pdf, doc_files/Aula2.pdf, doc_files/ScarletTown.txt]
Bookmarked= [doc_files/Apresentacao.pdf, doc_files/Aula2.pdf]
-------------------------------------------------
Search docs in library matching .*Lopes.*
-------------------------------------------------
doc_files/Apresentacao.pdf
doc_files/Aula2.pdf
-------------------------------------------------
Create two normal shelves and populate them
-------------------------------------------------
-------->> Shelves: AddShelfEvent [shelfName=Dco]<<--------
-------->> Shelves: AddShelfEvent [shelfName=Photos]<<--------
Library =
Document{title=Aula01, author=Antonia Lopes, file=doc_files/Apresentacao.pdf, date=2025-03-10, mimeType=application/pdf,
numPages=Optional[12], lastPageVisited=0, pages={1=Page{bookmark=true, annotations=[], pageNum=1}, 8=Page{bookmark=false,
annotations=[Annotation [text=CHECK MOODLE ALSO]], pageNum=8}}}
Document{title=Aula02, author=Antonia Lopes, file=doc_files/Aula2.pdf, date=2025-03-10, mimeType=application/pdf, numPages=Optional[13],
lastPageVisited=0, pages={1=Page{bookmark=true, annotations=[], pageNum=1}}}
Document{title=Lyrics Scarlet Town, author=n/a, file=doc_files/ScarletTown.txt, date=2025-02-20, mimeType=text/plain,
numPages=Optional.empty, lastPageVisited=0, pages={}}
Document{title=Flor, author=, file=doc_files/SummerFades.jpg, date=2011-06-09, mimeType=image/jpeg, numPages=Optional[1],
lastPageVisited=0, pages={}}
Document{title=Tulipas, author=, file=doc_files/tulips2.jpg, date=2011-06-09, mimeType=image/jpeg, numPages=Optional[1],
lastPageVisited=0, pages={}}
Shelves=
Recent= [doc_files/Apresentacao.pdf, doc_files/Aula2.pdf, doc_files/ScarletTown.txt]
Bookmarked= [doc_files/Apresentacao.pdf, doc_files/Aula2.pdf]
Photos= [doc_files/SummerFades.jpg, doc_files/tulips2.jpg]
Dco= [doc_files/Apresentacao.pdf, doc_files/Aula2.pdf]
-------------------------------------------------
Create a smart shelf with docs modified today
-------------------------------------------------
-------->> Shelves: AddShelfEvent [shelfName=Today]<<--------
Library =
Document{title=Aula01, author=Antonia Lopes, file=doc_files/Apresentacao.pdf, date=2025-03-10, mimeType=application/pdf,
numPages=Optional[12], lastPageVisited=0, pages={1=Page{bookmark=true, annotations=[], pageNum=1}, 8=Page{bookmark=false,
annotations=[Annotation [text=CHECK MOODLE ALSO]], pageNum=8}}}
Document{title=Aula02, author=Antonia Lopes, file=doc_files/Aula2.pdf, date=2025-03-10, mimeType=application/pdf, numPages=Optional[13],
lastPageVisited=0, pages={1=Page{bookmark=true, annotations=[], pageNum=1}}}
Document{title=Lyrics Scarlet Town, author=n/a, file=doc_files/ScarletTown.txt, date=2025-02-20, mimeType=text/plain,
numPages=Optional.empty, lastPageVisited=0, pages={}}
Document{title=Flor, author=, file=doc_files/SummerFades.jpg, date=2011-06-09, mimeType=image/jpeg, numPages=Optional[1],
lastPageVisited=0, pages={}}
Document{title=Tulipas, author=, file=doc_files/tulips2.jpg, date=2011-06-09, mimeType=image/jpeg, numPages=Optional[1],
lastPageVisited=0, pages={}}
Shelves=
Recent= [doc_files/Apresentacao.pdf, doc_files/Aula2.pdf, doc_files/ScarletTown.txt]
Bookmarked= [doc_files/Apresentacao.pdf, doc_files/Aula2.pdf]
Photos= [doc_files/SummerFades.jpg, doc_files/tulips2.jpg]
Dco= [doc_files/Apresentacao.pdf, doc_files/Aula2.pdf]
Today= [doc_files/Apresentacao.pdf, doc_files/Aula2.pdf]
-------------------------------------------------
Adding bookmarks to
doc_files/ScarletTown.txt
-------------------------------------------------
-------->> Document: ToggleBookmarkEvent [document=doc_files/ScarletTown.txt pageNum=1 isBookmarked=true]<<--------
Library =
Document{title=Aula01, author=Antonia Lopes, file=doc_files/Apresentacao.pdf, date=2025-03-10, mimeType=application/pdf,
numPages=Optional[12], lastPageVisited=0, pages={1=Page{bookmark=true, annotations=[], pageNum=1}, 8=Page{bookmark=false,
annotations=[Annotation [text=CHECK MOODLE ALSO]], pageNum=8}}}
Document{title=Aula02, author=Antonia Lopes, file=doc_files/Aula2.pdf, date=2025-03-10, mimeType=application/pdf, numPages=Optional[13],
lastPageVisited=0, pages={1=Page{bookmark=true, annotations=[], pageNum=1}}}
Document{title=Lyrics Scarlet Town, author=n/a, file=doc_files/ScarletTown.txt, date=2025-03-10, mimeType=text/plain,
numPages=Optional.empty, lastPageVisited=0, pages={1=Page{bookmark=true, annotations=[], pageNum=1}}}
Document{title=Flor, author=, file=doc_files/SummerFades.jpg, date=2011-06-09, mimeType=image/jpeg, numPages=Optional[1],
lastPageVisited=0, pages={}}
Document{title=Tulipas, author=, file=doc_files/tulips2.jpg, date=2011-06-09, mimeType=image/jpeg, numPages=Optional[1],
lastPageVisited=0, pages={}}
Shelves=
Recent= [doc_files/Apresentacao.pdf, doc_files/Aula2.pdf, doc_files/ScarletTown.txt]
Bookmarked= [doc_files/Apresentacao.pdf, doc_files/Aula2.pdf, doc_files/ScarletTown.txt]
Photos= [doc_files/SummerFades.jpg, doc_files/tulips2.jpg]
Dco= [doc_files/Apresentacao.pdf, doc_files/Aula2.pdf]
Today= [doc_files/Apresentacao.pdf, doc_files/Aula2.pdf, doc_files/ScarletTown.txt]
-------------------------------------------------
Delete
doc_files/Apresentacao.pdf
-------------------------------------------------
-------->> Library: RemoveDocumentEvent [document=doc_files/Apresentacao.pdf]<<--------
Library =
Document{title=Aula02, author=Antonia Lopes, file=doc_files/Aula2.pdf, date=2025-03-10, mimeType=application/pdf, numPages=Optional[13],
lastPageVisited=0, pages={1=Page{bookmark=true, annotations=[], pageNum=1}}}
```

```
Document{title=Lyrics Scarlet Town, author=n/a, file=doc_files/ScarletTown.txt, date=2025-03-10, mimeType=text/plain,
numPages=Optional.empty, lastPageVisited=0, pages={1=Page{bookmark=true, annotations=[], pageNum=1}}}
Document{title=Flor, author=, file=doc_files/SummerFades.jpg, date=2011-06-09, mimeType=image/jpeg, numPages=Optional[1],
lastPageVisited=0, pages={}}
Document{title=Tulipas, author=, file=doc_files/tulips2.jpg, date=2011-06-09, mimeType=image/jpeg, numPages=Optional[1],
lastPageVisited=0, pages={}}
Shelves=
Recent= [doc_files/Aula2.pdf, doc_files/ScarletTown.txt]
Bookmarked= [doc_files/Aula2.pdf, doc_files/ScarletTown.txt]
Photos= [doc_files/SummerFades.jpg, doc_files/tulips2.jpg]
Dco= [doc_files/Aula2.pdf]
Today= [doc_files/Aula2.pdf, doc_files/ScarletTown.txt]
-------------------------------------------------
Delete doc_files/tulips2.jpg
from shelf Photos
-------------------------------------------------
-------->> Shelves: RemoveDocumentShelfEvent [document=doc_files/tulips2.jpg shelf=Photos]<<--------
Library =
Document{title=Aula02, author=Antonia Lopes, file=doc_files/Aula2.pdf, date=2025-03-10, mimeType=application/pdf, numPages=Optional[13],
lastPageVisited=0, pages={1=Page{bookmark=true, annotations=[], pageNum=1}}}
Document{title=Lyrics Scarlet Town, author=n/a, file=doc_files/ScarletTown.txt, date=2025-03-10, mimeType=text/plain,
numPages=Optional.empty, lastPageVisited=0, pages={1=Page{bookmark=true, annotations=[], pageNum=1}}}
Document{title=Flor, author=, file=doc_files/SummerFades.jpg, date=2011-06-09, mimeType=image/jpeg, numPages=Optional[1],
lastPageVisited=0, pages={}}
Document{title=Tulipas, author=, file=doc_files/tulips2.jpg, date=2011-06-09, mimeType=image/jpeg, numPages=Optional[1],
lastPageVisited=0, pages={}}
Shelves=
Recent= [doc_files/Aula2.pdf, doc_files/ScarletTown.txt]
Bookmarked= [doc_files/Aula2.pdf, doc_files/ScarletTown.txt]
Photos= [doc_files/SummerFades.jpg]
Dco= [doc_files/Aula2.pdf]
Today= [doc_files/Aula2.pdf, doc_files/ScarletTown.txt]
-------------------------------------------------
Delete doc_files/Aula2.pdf
from shelf Recent
-------------------------------------------------
Raised OperationNotSupportedException
Library =
Document{title=Aula02, author=Antonia Lopes, file=doc_files/Aula2.pdf, date=2025-03-10, mimeType=application/pdf, numPages=Optional[13],
lastPageVisited=0, pages={1=Page{bookmark=true, annotations=[], pageNum=1}}}
Document{title=Lyrics Scarlet Town, author=n/a, file=doc_files/ScarletTown.txt, date=2025-03-10, mimeType=text/plain,
numPages=Optional.empty, lastPageVisited=0, pages={1=Page{bookmark=true, annotations=[], pageNum=1}}}
Document{title=Flor, author=, file=doc_files/SummerFades.jpg, date=2011-06-09, mimeType=image/jpeg, numPages=Optional[1],
lastPageVisited=0, pages={}}
Document{title=Tulipas, author=, file=doc_files/tulips2.jpg, date=2011-06-09, mimeType=image/jpeg, numPages=Optional[1],
lastPageVisited=0, pages={}}
Shelves=
Recent= [doc_files/Aula2.pdf, doc_files/ScarletTown.txt]
Bookmarked= [doc_files/Aula2.pdf, doc_files/ScarletTown.txt]
Photos= [doc_files/SummerFades.jpg]
Dco= [doc_files/Aula2.pdf]
Today= [doc_files/Aula2.pdf, doc_files/ScarletTown.txt]
-------------------------------------------------
Delete shelf with name Photos
-------------------------------------------------
-------->> Shelves: RemoveShelfEvent [shelfName=Photos]<<--------
Library =
Document{title=Aula02, author=Antonia Lopes, file=doc_files/Aula2.pdf, date=2025-03-10, mimeType=application/pdf, numPages=Optional[13],
lastPageVisited=0, pages={1=Page{bookmark=true, annotations=[], pageNum=1}}}
Document{title=Lyrics Scarlet Town, author=n/a, file=doc_files/ScarletTown.txt, date=2025-03-10, mimeType=text/plain,
numPages=Optional.empty, lastPageVisited=0, pages={1=Page{bookmark=true, annotations=[], pageNum=1}}}
Document{title=Flor, author=, file=doc_files/SummerFades.jpg, date=2011-06-09, mimeType=image/jpeg, numPages=Optional[1],
lastPageVisited=0, pages={}}
Document{title=Tulipas, author=, file=doc_files/tulips2.jpg, date=2011-06-09, mimeType=image/jpeg, numPages=Optional[1],
lastPageVisited=0, pages={}}
Shelves=
Recent= [doc_files/Aula2.pdf, doc_files/ScarletTown.txt]
Bookmarked= [doc_files/Aula2.pdf, doc_files/ScarletTown.txt]
Dco= [doc_files/Aula2.pdf]
Today= [doc_files/Aula2.pdf, doc_files/ScarletTown.txt]
-------------------------------------------------
Delete shelf with name Recent
-------------------------------------------------
OperationNotSupportedException
Library =
Document{title=Aula02, author=Antonia Lopes, file=doc_files/Aula2.pdf, date=2025-03-10, mimeType=application/pdf, numPages=Optional[13],
lastPageVisited=0, pages={1=Page{bookmark=true, annotations=[], pageNum=1}}}
Document{title=Lyrics Scarlet Town, author=n/a, file=doc_files/ScarletTown.txt, date=2025-03-10, mimeType=text/plain,
numPages=Optional.empty, lastPageVisited=0, pages={1=Page{bookmark=true, annotations=[], pageNum=1}}}
Document{title=Flor, author=, file=doc_files/SummerFades.jpg, date=2011-06-09, mimeType=image/jpeg, numPages=Optional[1],
lastPageVisited=0, pages={}}
Document{title=Tulipas, author=, file=doc_files/tulips2.jpg, date=2011-06-09, mimeType=image/jpeg, numPages=Optional[1],
lastPageVisited=0, pages={}}
Shelves=
Recent= [doc_files/Aula2.pdf, doc_files/ScarletTown.txt]
Bookmarked= [doc_files/Aula2.pdf, doc_files/ScarletTown.txt]
Dco= [doc_files/Aula2.pdf]
Today= [doc_files/Aula2.pdf, doc_files/ScarletTown.txt]
```