TÉCNICO LISBOA

Projecto de Programação com Objectos

8 de Setembro de 2025

Esclarecimento de dúvidas:

- Consultar sempre o corpo docente atempadamente: presencialmente ou através do endereço **po-tagus@disciplinas.tecnico.ulisboa.pt**.
- Não utilizar fontes de informação não oficialmente associadas ao corpo docente (podem colocar em causa a aprovação à disciplina).
- Não são aceites justificações para violações destes conselhos: quaisquer consequências nefastas são da responsabilidade do aluno.

Requisitos para desenvolvimento, material de apoio e actualizações do enunciado (ver informação completa na secção **[Projecto]** no Fénix):

- O material de apoio é de uso obrigatório e não pode ser alterado.
- Verificar atempadamente (mínimo de 48 horas antes do final de cada prazo) os requisitos exigidos pelo processo de avaliação.

Processo de avaliação:

- O projecto é constituído por três entregas: (i) A primeira corresponde à modelação do domínio da solução; (ii) a segunda corresponde à concretização de uma parte do domínio da solução e de algumas funcionalidades da aplicação; e (iii) a terceira parte corresponde à concretização do domínio total da solução e das restantes funcionalidades da aplicação.
- Após a entrega 3ª parte do projecto, haverá a realização de um teste prático **individual**.
- Ver informação completa nas secções **[Projecto]** e **[Método de Avaliação]** da página Web da disciplina no Fénix) (incluindo as datas para cada uma das entregas do projecto e do teste prático).
- O material de apoio é de uso obrigatório e não pode ser alterado. Não serão consideradas quaisquer alterações aos ficheiros de apoio disponibilizados: eventuais entregas dessas alterações serão automaticamente substituídas durante a avaliação da funcionalidade do código entregue.
- Trabalhos não presentes no Fénix no final do prazo têm classificação 0 (zero) (não são aceites outras formas de entrega).
- A avaliação do projecto pressupõe o compromisso de honra de que o trabalho foi realizado pelos alunos correspondentes ao grupo de avaliação.
- **Fraudes na realização do projecto terão como resultado a exclusão dos alunos implicados no processo de avaliação**.

O objectivo do projecto é desenvolver uma aplicação de gestão do acervo de uma biblioteca. Este documento está organizado da seguine forma. A secção 1 apresenta as entidades do domínio da aplicação a desenvolver. As funcionalidades da aplicação a desenvolver são descritas nas secções 4 e 5. A secção 2 descreve os requisitos de desenho que a aplicação desenvolvida deve oferecer.

A secção 2 descreve as qualidades que a solução deve oferecer.

Neste texto, o tipo **negrito** indica um literal (i.e., é exactamente como apresentado); o símbolo ␣ indica um espaço; e o tipo *itálico* indica uma parte variável (i.e., uma descrição).

1 Entidades do Domínio

Nesta secção descrevem-se as várias entidades que vão ser manipuladas no contexto da aplicação a desenvolver. Existem vários conceitos importantes neste contexto, como por exemplo, obras, criadores, utentes e requisições. Os conceitos listados não são os únicos possíveis no modelo a realizar por cada grupo e as suas relações (assim como relações com outros conceitos não mencionados) podem depender das escolhas do projecto.

1.1 Criadores

Um criador representa alguém que criou uma obra, por exemplo, escreveu um livro ou realizou um filme. Cada criador tem um nome (que é um identificador único no contexto desta aplicação) e deve saber as obras que criou. O sistema é responsável por manter actualizado a lista de obras para cada criador. Só devem ser mantidos os criadores que têm pelo menos uma obra. Se um criador deixar de ter obras, então deve ser removido do sistema.

Note-se que os criadores, autores dos livros e realizadores dos filmes, começam por ter apenas um nome. Deve ser possível adicionar outras propriedades sem impacto para o código já desenvolvido.

1.2 Obras

O sistema mantém um registo de obras da biblioteca. Cada obra é identificada por um número de obra. O identificador é atribuído automaticamente e incrementalmente (a partir de 1 ou do último valor atribuído, caso o estado do sistema tenha sido recuperado).

As obras registam ainda o número de exemplares existentes no acervo da biblioteca (várias cópias da mesma obra). Todas as obras têm um título (cadeia de caracteres) e um preço (número inteiro, por simplicidade).

Cada obra tem uma categoria, de acordo com o assunto nela tratado. Inicialmente, consideram-se as seguintes categorias: (i) obras de referência: onde se incluem dicionários, gramáticas, enciclopédias e documentários; (ii) obras de ficção; e (iii) obras técnicas e científicas.

As obras a considerar inicialmente são livros e DVDs. As propriedades específicas de cada um (além das gerais) são as seguintes:

- Livros – O sistema deverá manter, para cada livro, a seguinte informação: autores (um ou mais criadores), e ISBN (cadeia com dez ou 13 caracteres);
- DVDs – Para cada DVD, o sistema deverá manter: realizador (apenas um criador), e o número de registo na IGAC (Inspecção-Geral das Actividades Culturais) (cadeia de caracteres).

Deve ser possível criar novos tipos de obras. O impacto da introdução dos novos tipos na implementação desenvolvida deve ser mínimo. Deve ainda ser possível adicionar novas categorias com impacto mínimo na aplicação já desenvolvida.

1.2.1 Alterações de Inventário de Obras

O inventário de uma obra pode ser alterado através da especificação de um valor a somar/subtrair ao número de exemplares: se a quantidade indicada for um valor positivo, aumenta-se o número de exemplares disponíveis; se for um valor negativo, decrementa-se o número de exemplares disponíveis desde que o número de exemplares disponíveis não fique negativo. Caso o número de exemplares de uma obra chegue a 0, então a obra deve ser removida do sistema. Note-se que a remoção de uma obra tem impacto na gestão de criadores.

1.3 Utentes

O sistema mantém um registo de utentes da biblioteca. Cada utente é identificado por um número de utente. O identificador é atribuído automaticamente e incrementalmente (a partir de 1 ou do último valor atribuído, caso o estado do sistema tenha sido recuperado).

O sistema mantém ainda, para cada utente, o seu nome e endereço de correio electrónico. Estes campos não podem ser vazios. É ainda mantida informação sobre a situação do utente perante a biblioteca: (i) activo, i.e., o utente pode fazer requisições; ou (ii) suspenso, i.e., o utente não pode fazer novas requisições. Um utente é suspenso se não devolver uma obra requisitada dentro do prazo estipulado. Permanece suspenso enquanto tiver uma ou mais obras requisitadas por entregar fora do prazo de entrega ou não tiver pago a multa referente ao atraso na entrega.

A biblioteca distingue entre utentes que cumprem as regras de funcionamento e utentes que violam sistematicamente os compromissos. Existe ainda uma classificação intermédia para novos utentes ou para utentes com comportamento misto. Um utente que nas últimas 3 requisições não tenha cumprido os prazos de devolução, é classificado como faltoso. Um utente faltoso que proceda a 3 devoluções consecutivas dentro do prazo é considerado normal. Um cliente que tenha cumprido rigorosamente os prazos de entrega nas últimas 5 requisições é classificado como cumpridor. Em todos os outros casos, o utente não tem classificação especial e é considerado normal. Como se verá adiante, a classificação influencia a capacidade de requisição de um utente activo.

O comportamento (no que diz respeito a entrega, dentro ou fora do prazo, de obras requisitadas antes da suspensão) de um utente suspenso também influencia a sua classificação. Deve ser possível discriminar os utentes quanto à sua conduta, considerando outros critérios ou novas classificações, com um impacto mínimo na implementação desenvolvida.

1.4 Requisições

O sistema garante o cumprimento de regras para a requisição de obras. As regras têm um identificador numérico e dependem das características da obra que se pretende requisitar e da conduta passada do utente. As regras devem ser verificadas por ordem crescente do seu identificador e representam uma condição que o utentes e/ou a obra em questão devem verificar. As regras a considerar são as seguintes (os números de ordem são os identificadores das regras correspondentes):

1. O utente não pode requisitar duas vezes a mesma obra (i.e., em duas requisições diferentes e simultaneamente abertas);
2. O utente está activo;
3. A obra tem que ter pelo menos um exemplar disponível;
4. O utente não pode ter mais que n obras requisitadas em cada momento (valor base: 3; utentes cumpridores: 5; utentes faltosos: 1);

5. A categoria da obra não é a categoria *referência*;
6. O utente não pode requisitar obras com um preço superior a €25,00 (não aplicável a utentes cumpridores);

No caso de violação da regra 3, o utente pode pedir para ser notificado assim que a obra tenha pelo menos um exemplar disponível. A notificação consiste na indicação de que a obra já se encontra disponível.

Ao requisitar uma obra, o utente deve ser informado da data limite para a devolução. O tempo de requisição permitido para cada obra depende do número total de exemplares que constem do acervo da biblioteca e da conduta do utente. Os prazos, em dias, são os seguintes:

- Obras com apenas um exemplar: valor de base: 3; utentes cumpridores: 8; utentes faltosos: 2;
- Obras com 5 exemplares ou menos: valor de base 8; utentes cumpridores: 15; utentes faltosos: 2;
- Obras com mais de 5 exemplares: valor de base 15; utentes cumpridores: 30; utentes faltosos: 2.

A alteração do número de exemplares de uma obra só tem impacto no prazo de de entraga das requisições que venham a acontecer após a alteração do número de exemplares.

Se o utente não entregar as obras requisitadas no prazo devido, fica imediatamente suspenso, não podendo requisitar mais obras até regularizar a situação. Por cada dia de atraso, o utente fica sujeito ao pagamento de uma multa de €5,00 (cinco euros). A situação só se considera regularizada após a devolução das obras em atraso e o pagamento da multa. Para efeitos de pagamento de multas, fracções de dia contam como um dia (a unidade de tempo do sistema é o dia).

Deve ser possível alterar ou acrescentar regras para a requisição de obras, bem como fazer alterações aos tempos de requisição permitidos. As alterações devem ter impacto mínimo na concretização desenvolvida.

1.5 Gestão de Tempo

A unidade de tempo do sistema é o dia. A data do sistema começa no dia 1 (um) e faz parte do estado persistente do sistema. Sempre que a data é alterada, deve ser verificada a situação dos utentes.

1.6 Notificações

Deve existir um mecanismo de notificações que permita avisar eventuais interessados quando as obras ficam em determinadas situações:

- Quando uma obra é emprestada, quer-se enviar uma notificação a todas as entidades que demonstraram interesse nessa operação.
- Quando uma obra sem nenhum exemplar disponível passa a ter pelo menos um exemplar disponível, quer-se enviar uma notificação a todas as entidades que demonstraram interesse nesta situação.

As notificações de um utilizador apenas devem ser apresentadas uma única vez. Após esta visualização, considera-se que o utilizador fica sem notificações. As notificações devem ser apresentadas pela mesma ordem em que foram enviadas pelo sistema.

Note-se que existem diferenças relativamente à manutenção do interesse da entidade interessada:

- No caso de uma entidade interessada em ser avisada sempre que uma dada obra é emprestada, isto significa que enquanto o interesse for válido, então a entidade deverá receber uma notificação sempre que um utilizador pede a obra emprestada;
- No caso de uma entidade interessada em ser avisada da disponibilidade de uma obra, isto significa que sempre que a obra em causa passar da situação de indisponível (todos os exemplares emprestados) para disponível (pelo menos um exemplar disponível), a entidade deve receber uma notificação. Neste caso, o interesse da entidade é cancelado de forma automática quando a entidade em questão conseguir requisitar a obra em causa.

1.7 Serialização

É possível guardar e recuperar o estado actual da aplicação, preservando toda a informação relevante do domínio da aplicação.A serialização Java usa as classes da package `java.io`, em particular, a interface`java.io.Serializable` e as classes de leitura `java.io.ObjectInputStream` e escrita `java.io.ObjectOutputStream` (entre outras).

2 Requisitos de Desenho

O sistema a desenvolver deve seguir o princípio de desenho *aberto-fechado* por forma a aumentar a extensibilidade do seu código. Por exemplo, deve ser possível adicionar novos tipos de obra com um impacto mínimo no código já existente do domínio da aplicação.

Deve ser possível introduzir alterações nas regras para requisição de obras de forma simples e permitir a adição de novas regras ou remoção de regras existentes com um impacto mínimo no código existente.

Aplicação de um padrão de desenho para melhorar a legibilidade do código relacionado com o conceito Utente e permitir ao mesmo tempo que se possam adicionar novas classificações de utentes com um impacto reduzido no código já realizado. O padrão deve ainda poder permitir a fácil alteração da funcionalidade associada a cada comportamento.

Aplicação de um padrão de desenho para que se possam definir novas entidades que desejem ser notificadas da requisição ou disponibilidade de obras sem que isso implique alterações no código já realizado.

A aplicação deve estar preparada para que se possa remover um utente por forma a minimizar o impacto da futura inclusão deste requisito.

3 Arquitectura da Aplicação

A aplicação a desenvolver deve seguir uma arquitectura em camadas: as entidades relacionadas com interacção com o utilizador estão concretizadas na camada de apresentação (package `bci.app`) e as entidades relacionadas com as entidades do domínio estão concretizadas na camada de domínio (package `bci.core`). Por forma a garantir uma independência entre as várias camadas da aplicação não deve haver código de interacção com o utilizador no código respeitante ao domínio da aplicação (concretizada na camada de domínio). Desta forma, será possível reutilizar o código do domínio da aplicação com outras concretizações da interface com o utilizador sem que isso implique alterar o código da camada de domínio. Para garantir um melhor desenho da aplicação toda a lógica de negócio deve estar concretizada no código respeitante ao domínio da aplicação (camada *core*) e não na camada de apresentação. Assim potencia-se a reutilização de código e, além disso, o código fica concretizado nas entidades a que diz respeito, o que torna a aplicação mais legível e mais fácil de manter.

A interacção com o utilizador deve ser realizada utilizando a *framework* de interacção com o utilizador **po-uilib** (disponível na secção **Projecto** da página da disciplina. A interacção com o utilizador suportada por esta framework é realizada através de menus, em que cada menu tem um determinado conjunto de opções. Cada menu é concretizado por uma subclasse da classe `pt.tecnico.po.ui.Menu`) (fornecida pela framework), onde se deve indicar qual o conjunto de opções do menu. Cada opção de um menu (designado como *comando*) é concretizada por uma subclasse de `pt.tecnico.uilib.menus.Command` (também fornecida pela framework) que tem a responsabilidade de suportar a funcionalidade correspondente à opção em causa. Todos os menus suportam automaticamente a opção **Sair** (fecha o menu).

As operações de pedido e apresentação de informação ao utilizador **devem** realizar-se através dos objectos *form* e *display*, respectivamente, presentes em cada comando. No caso de um comando utilizar mais do que um formulário para interagir com o utilizador então será necessário criar pelo menos mais uma instância de `Form` durante a execução do comando em causa. As mensagens a apresentar ao utilizador normalmente são produzidas pelos métodos de interfaces já definidas na camada de apresentação do esqueleto da aplicação fornecido (ver secção 3.1).

Podem acontecer situações de erro durante a execução de um comando. Quando isto ocorre, o comando não deve ter qualquer efeito no estado da aplicação (excepto se indicado em contrário na operação em causa) e o comando deve instanciar uma excepção (pertencente a uma subclasse de `pt.tecnico.uilib.menus.CommandException` correspondente ao erro que aconteceu) e lançá-la de seguida. Estas excepções serão depois tratadas automaticamente pelo object `Menu` no contexto do qual o comando que lançou a excepção está a ser executado.

3.1 Esqueleto da Aplicação

Já é fornecido um esqueleto da aplicação a desenvolver, não sendo por isso necessário concretizar a aplicação de raiz. Este esqueleto está disponível no arquivo `bci-skeleton.jar` presente na secção *Projecto* da página da cadeira. O esqueleto inclui a classe `bci.app.App`, que representa o ponto de entrada da aplicação a desenvolver e os vários comandos e menus da aplicação a desenvolver (descritos na secção 4). Alguns dos comandos já estão completamente finalizados, outros (a grande maioria) estão parcialmente concretizados e têm de ser alterados por forma a suportarem a funcionalidade descrita neste enunciado. Cada menu e respectivos comandos estão definidos num subpackage de `bci.app`. Por exemplo, o menu principal e respectivos comandos estão definidos em `bci.app.main`. Os menus já estão concretizados.

Cada subpackage relacionado com um menu contém ainda as interfaces `Message`, `Prompt` (definem as mensagens a utilizar na interacção com o utilizador) e `Label` (define as etiquetas do menu e dos vários comandos). **Não** podem ser definidas novas mensagens. Potenciais omissões devem ser esclarecidas com o corpo docente antes de qualquer concretização.

O esqueleto inclui ainda algumas classes do domínio da aplicação (presentes no package `bci.core`) parcialmente concretizadas. Será ainda necessário concretizar novas entidades do domínio da aplicação e finalizar as que já estão parcialmente fornecidas por forma a ter um conjunto de entidades que suportem a totalidade da funcionalidade do domínio da aplicação a desenvolver.

Finalmente, as excepções a lançar nos vários comandos já estão definidas no package `bci.app.exception` do esqueleto da aplicação. Cada uma destas excepções representa uma situação de erro distinta e deve ser lançada por um comando sempre que essa situação acontece. Estas excepções serão depois tratadas de forma automática pela biblioteca de interacção *po-uilib* no contexto da classe `Menu`. Estas excepções **não** podem ser alteradas e apenas devem ser utilizadas no código de interacção com o utilizador pela razão indicada anteriormente. O package `bci.core.exception` contém algumas excepções a utilizar na camada do domínio, mas podem (e **devem**) ser acrescentadas novas excepções para representar outras situações anómalas que possam acontecer no contexto do domínio.

4 Interação com o Utilizador

Esta secção descreve a **funcionalidade máxima** da interface com o utilizador. Em geral, os comandos pedem toda a informação antes de proceder à sua validação (excepto onde indicado).

A aplicação permite gerir a informação sobre as várias entidades do modelo. Possui ainda a capacidade de preservar o seu estado (não é possível manter várias versões do estado da aplicação em simultâneo). No início, a aplicação está vazia, mas pode ser carregada uma base de dados textual com conceitos pré-definidos. Neste caso, a aplicação começará com um estado referente às entidades que foram carregadas no arranque da aplicação. Deve ser possível registar, visualizar, e operar sobre as várias entidades do domínio. Deve ser possível efectuar pesquisas sujeitas a vários critérios e sobre as diferentes entidades geridas pela aplicação.

A descrição da funcionalidade de cada comando indica quais as excepções que podem ser lançadas pelo comando. Para cada interacção com o utilizador que exista no contexto de um comando em que é necessário utilizar uma mensagem específica é indicado qual é o método da interface `Prompt` (caso a mensagem seja para pedir dados) ou `Message` (caso a mensagem seja para apresentar dados) que é responsável por devolver a mensagem em questão.

No contexto de um comando pode ser pedido ao utilizador um ou mais identificadores de objectos do domínio necessários para a realização da funcionalidade do comando em causa. A mensagem a utilizar para pedir o identificador e a excepção a lançar caso o identificador não corresponda a um objecto conhecido (excepto no processo de registo ou caso indicado em contrário) depende do tipo de entidade em questão:

Entidade	Pedido a apresentar	Excepção a lançar se desconhecido
Utente	`bci.app.user.Prompt.userId()`	`bci.app.exception.NoSuchUserException`
Obra	`bci.app.workl.Prompt.workId()`	`bci.app.exception.NoSuchWorkException`
Criador	`bci.app.work.Prompt.creatorId()`	`bci.app.exception.NoSuchCreatorException`

Note-se que estas excepções não devem ser utilizadas no código do domínio da aplicação como explicado em 3. Alguns casos particulares podem usar pedidos específicos não apresentados nesta tabela.

4.1 Menu Principal

As acções deste menu permitem gerir a salvaguarda do estado da aplicação, ver e alterar a data actual e abrir submenus. A lista completa é a seguinte: **Abrir**, **Guardar**, **Mostrar data actual**, **Avançar data actual**, **Menu de Gestão de Utentes**, **Menu de Gestão de Obras** e **Menu de Gestão de Requisições**. Por omissão, as mensagens de diálogo para este menu estão definidos em `bci.app.main.Prompt` e `bci.app.main.Message`.

Os comandos que vão concretizar as funcionalidades deste menu já estão parcialmente concretizados nas várias classes do package `bci.app.main`: `DoOpenFile`, `DoSaveFile`, `DoDisplayDate`, `DoAdvanceDate`, `DoOpenMenuUsers`, `DoOpenMenuWorks` e `DoOpenMenuRequests`.

4.1.1 Salvaguarda do estado actual

O conteúdo da aplicação (que representa o estado relevante actualmente em memória da aplicação) pode ser guardado para posterior recuperação (via serialização Java: `java.io.Serializable`). Na leitura e escrita do estado da aplicação, devem ser tratadas as excepções associadas. A funcionalidade é a seguinte:

Abrir – Carrega os dados de uma sessão anterior a partir de um ficheiro previamente guardado (ficando este ficheiro associado à aplicação, para futuras operações de salvaguarda). Pede-se o nome do ficheiro a abrir (utilizando a cadeia de caracteres devolvida por `Prompt.openFile()`). Caso ocorra um problema na abertura ou processamento do ficheiro, deve ser lançada a excepção `FileOpenFailedException`. A execução bem sucedida desta opção substitui toda a informação da aplicação.

Guardar – Guarda o estado actual da aplicação no ficheiro associado. Se não existir associação, pede-se o nome do ficheiro a utilizar ao utilizador (utilizando a cadeia de caracteres devolvida por `Prompt.newSaveAs()`), ficando a aplicação associada a este ficheiro. Não é executada nenhuma acção se não existirem alterações desde a última salvaguarda.

A aplicação apenas gere uma biblioteca em cada momento. Quando se abandona uma biblioteca com modificações não guardadas (porque se abra outra), deve-se perguntar ao utilizador se quer guardar a informação actual antes carregar a nova biblioteca na aplicação, através de `Prompt.saveBeforeExit()` (a resposta é obtida invocando o método `readBoolean()` ou `Form.confirm()` da *framework* de interacção com o utilizador).

Note-se que a opção `Abrir` não suporta a leitura de ficheiros de texto (estes apenas são utilizados na inicialização da aplicação). A opção `Sair` **nunca** guarda o estado da aplicação, mesmo que existam alterações.

4.1.2 Mostrar data actual

A data actual do sistema é apresentada através da mensagem `Messge.currentDate()`.

4.1.3 Avançar data actual

O número de dias a avançar é pedido utilizando a mensagem devolvida por `Prompt.daysToAdvance()`. O valor indicado deve ser positivo. Caso contrário, a operação não tem efeito.

Além da data, o sistema deve actualizar, caso seja necessário, outros aspectos que dela dependam, designadamente, a situação dos utentes relativa a prazos.

4.1.4 Gestão e consulta de dados da aplicação

A gestão e consulta de dados da aplicação são realizadas através das seguintes opções:

Menu de Gestão de Utentes – Abre o menu de gestão de utentes e operações associadas.

Menu de Gestão de Obras – Abre o menu de gestão de obras e operações associadas.

Menu de Gestão de Requisições – Abre o menu de gestão de requisições e operações associadas.

4.2 Menu de Gestão de Utentes

Este menu permite efectuar operações sobre os utentes da biblioteca. A lista completa é a seguinte: **Registar utente**, **Mostrar utente**, **Mostrar utentes**, **Mostrar notificações do utente** e **Pagar multa**. Os comandos deste menu já estão parcialmente concretizados nas classes do package `bci.app.user`: `DoRegisterUser`, `DoShowUser`, `DoShowUserNotifications`, `DoShowUsers` e `DoPayFine`. Por omissão, as mensagens de diálogo para este menu estão definidos em `bci.app.user.Prompt` e `bci.app.user.Message`.

4.2.1 Registar utente

Pede o nome (`Prompt.userName()`) e o endereço de correio electrónico (`Prompt.userEMail()`). O registo bem sucedido é assinalado através da mensagem devolvida por `Message.registrationSuccessful()`; caso contrário, é lançada a excepção `UserRegistrationFailedException`.

Note-se que a atribuição do identificador do utente é automática e que utentes diferentes são registados em cada operação de registo.

4.2.2 Mostrar utente

É pedido o identificador do utente, sendo apresentadas as informações sobre esse utente, de acordo com o seguinte formato (e variações descritas abaixo). No caso de utentes suspensos, a multa a apresentar é um valor inteiro. Os formatos de apresentação de um utente activo e suspenso são os seguintes:

```
id␣-␣nome␣-␣email␣-␣comportamento␣-␣ACTIVO
id␣-␣nome␣-␣email␣-␣comportamento␣-␣SUSPENSO␣-␣EUR␣multa
```

Exemplo de apresentação de utentes:

```
2␣-␣Alberto␣Meireles␣-␣ameireles@mymail.com␣-␣CUMPRIDOR␣-␣ACTIVO
1␣-␣Fernando␣Meireles␣-␣fmeireles@mymail.com␣-␣FALTOSO␣-␣SUSPENSO␣-␣EUR␣10
3␣-␣Fernando␣Meireles␣-␣ffm@mymail.com␣-␣NORMAL␣-␣ACTIVO
```

4.2.3 Mostrar notificações do utente

É pedido o identificador do utente, sendo apresentadas as notificações para esse utente, de acordo com os seguintes formatos (correspondente aos casos descritos na secção §1.6):

```
DISPONIBILIDADE:␣descrição␣de␣obra␣disponível
REQUISIÇÃO:␣descrição␣de␣obra␣requisitada
```

Exemplos de notificações

```
DISPONIBILIDADE:␣4␣-␣2␣de␣4␣-␣DVD␣-␣Casamento␣Real␣-␣8␣-␣Ficção␣-␣António␣Fonseca␣-␣200400500
REQUISIÇÃO:␣5␣-␣4␣de␣22␣-␣Livro␣-␣Dicionário␣-␣45␣-␣Referência␣-␣Pedro␣Casanova␣-␣1234567893
```

Note-se que a descrição é idêntica à que é realizada para mostrar cada obra (ver secção §4.3.1) . No entanto, a solução deve ser suficientemente flexível para permitir outros formatos de apresentação das notificações (sem impacto no código do domínio da aplicação).

4.2.4 Mostrar utentes

Apresenta informações sobre todas os utentes registados na biblioteca, ordenando-os lexicograficamente pelo nome. Caso existam utentes com o mesmo nome, devem ser ordenados por ordem crescente dos seus identificadores. A informação de cada utente é apresentada de acordo com o formato descrito na secção §4.2.2.

4.2.5 Pagar multa

Pede o identificador do utente cuja multa deve ser paga. Se o utente estiver suspenso, a multa é saldada e o utente deixará de estar suspenso (caso não tenha nenhuma obra por entregar fora de prazo). Se o utente não estiver suspenso, deve lançar-se a excepção `UserIsActiveException`.

4.3 Menu de Gestão de Obras

Este menu apresenta as operações disponíveis sobre obras. A lista completa é a seguinte: `Mostrar obra`, `Mostrar obras`, `Mostrar obras de criador`, `Alterar inventário de uma obra` e `Efectuar pesquisa`. Os comandos que concretizam estas operações já estão parcialmente concretizados nas classes do package `bci.app.work`: `DoShowWork`, `DoShowWorks`, `DoDisplayWorksByCreator`, `DoChangeWorkInventory` e `DoPerformSearch`.

4.3.1 Mostrar obra

É pedido o identificador da obra. Se a obra existir, é apresentada de acordo com o seguinte seguinte formato genérico (para livros e DVDs):

```
id␣-␣disponíveis␣de␣total␣-␣tipo␣-título␣-␣preço␣-␣categoria␣-␣informação␣adicional
```

onde *disponíveis* e *total* representam, respectivamente, o número de exemplares disponíveis e o número total de exemplares da obra em causa. Para livros, a informação adicional corresponde à lista de autores (preservando a ordem dos autores indicada na criação da obra – os autores devem estar separados pelo carácter `;`) e ao ISBN; para DVDs, a informação adicional corresponde ao realizador e ao número de registo no IGAC.

Exemplos de apresentação de obras

```
␣3␣-␣20␣de␣23␣-␣Livro␣-␣Casa␣Azul␣-␣15␣-␣Ficção␣-␣João␣Fonseca;␣Zé␣Fonseca␣-␣1234567891
␣4␣-␣2␣de␣2␣-␣DVD␣-␣Casamento␣Real␣-␣8␣-␣Ficção␣-␣António␣Fonseca␣-␣200400500
␣5␣-␣0␣de␣4␣-␣Livro␣-␣Dicionário␣-␣45␣-␣Referência␣-␣Pedro␣Casanova␣-␣1234567893
␣6␣-␣1␣de␣21␣-␣Livro␣-␣Enciclopédia␣-␣100␣-␣Técnica␣e␣Científica␣-␣Zé␣Fonseca␣-␣1234567894
```

4.3.2 Mostrar obras

Apresenta informações sobre todas as obras, ordenando-as pelos seus identificadores. O formato de apresentação de cada obra segue o mesmo formato descrito na secção 4.3.1.

4.3.3 Mostrar obras de criador

É pedido o identificador de um criador e mostram-se todas as obras desse criador, ordenando-as pelos seus títulos (sem diferenças entre maiúsculas e minúsculas). O formato de apresentação de cada obra segue o mesmo formato descrito na secção 4.3.1.

4.3.4 Alterar inventário de uma obra

É pedido o identificator de uma obra e uma quantidade (`Prompt.amountToUpdate()`) e actualizado o número de exemplares disponíveis da obra indicada somando a quantidade indicada ao número de exemplares disponíveis. No caso de não ser possível actualizar o número de exemplares devido ao facto de a quantidade indicada ser inválida, então a operação não deve ter qualquer efeito e deve ser apresentada a mensagem `Message.notEnoughInventory()`.

4.3.5 Efectuar pesquisa

Esta opção realiza uma procura por termo (cadeia de caracteres), pedido através de `Prompt.searchTerm()`. Como resultado, deve ser apresentada uma lista das obras encontradas pela pesquisa, ordenadas por ordem crescente do seu identificador, utilizando o formato descrito na secção 4.3.1.

O termo de pesquisa deve ser comparado (sem distinção entre letras maiúsculas e minúsculas) com os campos relevantes de cada obra: para DVDs, o realizador e o título; para livros, cada um dos seus autores e o título. Só devem ser apresentadas obras que contenham o termo de pesquisa num dos campos relevantes.

Assim, considerando as quatro obras no exemplo anterior, uma pesquisa pelo termo **casa** apresentaria a obras com os identificadores 3, 4 e 5:

```
3␣-␣20␣de␣23␣-␣Livro␣-␣Casa␣Azul␣-␣15␣-␣Ficção␣-␣João␣Fonseca,␣Zé␣Fonseca␣-␣1234567891
4␣-␣2␣de␣2␣-␣DVD␣-␣Casamento␣Real␣-␣8␣-␣Ficção␣-␣António␣Fonseca␣-␣200400500
5␣-␣0␣de␣4␣-␣Livro␣-␣Dicionário␣-␣45␣-␣Referência␣-␣Pedro␣Casanova␣-␣1234567893
```

Caso não sejam encontradas obras, não deve ser produzido qualquer resultado.

4.4 Menu de Gestão de Requisições

Este menu apresenta as operações relacionadas com requisições de obras. A lista completa é a seguinte: `Requisitar obra` e `Devolver obra`. As operações deste menu já estão parcialmente concretizadas nas classes da package `bci.app.request`, respectivamente: `DoRequestWork`, `DoReturnWork`. Por omissão, as mensagens de diálogo para este menu estão definidos em `bci.app.request.Prompt` e `bci.app.request.Message`.

4.4.1 Requisitar obra

No processo de requisição de uma obra, o sistema pede, primeiro, a identificação do utente e, de seguida, o identificador da obra a requisitar. Se o utente não puder requisitar a obra (considerando-se as regras definidas acima), deve ser lançada a excepção `BorrowingRuleFailedException` (excepto regra 3: ver a seguir).

Se a requisição não for possível por falta de exemplares (violação da regra 3), deve-se perguntar ao utente, utilizando a mensagem devolvida por `Prompt.returnNotificationPreference()`, se deseja ser notificado acerca da devolução. No caso de uma requisição bem sucedida, deve-se utilizar a mensagem devolvida por `Message.workReturnDay()` para comunicar o prazo de devolução.

4.4.2 Devolver obra

No processo de devolução de uma obra, o sistema pede, primeiro, o identificador do utente e, de seguida, o da obra a devolver. Se a obra não tiver sido requisitada pelo utente indicado, deve-se lançar a excepção `WorkNotBorrowedByUserException`. Caso contrário, o sistema processa a entrega e, caso haja lugar ao pagamento de multa, é apresentada a mensagem devolvida por `Message.showFine()`.

O utente pode entregar uma obra sem pagar a multa, continuando suspenso até regularizar a situação, sem prejuízo da obra ser assinalada como entregue. Antes de liquidar a multa, o sistema interroga o utente sobre o desejo de pagamento, através da mensagem devolvida por `Prompt.finePaymentChoice()`. Se a resposta for positiva, a multa é liquidada e o utente fica activo caso não tenha nenhuma obra por entregar fore de prazo.

5 Leitura de Dados a Partir de Ficheiros Textuais

Além das opções de manipulação de ficheiros descritas na secção §4.1.1, é possível iniciar a aplicação com um ficheiro de texto especificado pela propriedade Java com o nome `import`. Quando se especifica esta propriedade, a aplicação é povoada com os objectos correspondentes ao conteúdo do ficheiro indicado. Cada linha deste ficheiro textual descreve uma dada entidade a carregar no estado inicial do sistema. Pode assumir que não existem entradas mal-formadas nestes ficheiros. A codificação dos ficheiros a ler é garantidamente `UTF-8`. Sugere-se a utilização do método `String.split` para o processamento preliminar das linhas do ficheiro de texto.

Cada linha do ficheiro de texto diz respeito a uma entidade distinta (utente ou obra). As obras da biblioteca têm o seguinte formato genérico, respectivamente, para DVDs e livros:

```
DVD:título:realizador:preço:categoria:númeroIGAC:exemplares
BOOK:título:autores:preço:categoria:ISBN:exemplares
```

O campo autores é uma lista (separada por vírgulas – podem ocorrer espaços perto das vírgulas): `autor1, autor2, ..., autorN`. Assume-se que os títulos das obras não podem conter o carácter **:** e que o preço é um número inteiro. O campo *exemplares* indica o número de exemplares da obra disponíveis na biblioteca.

Por seu lado, os utentes são descritos aplicando o seguinte formato genérico: `USER:nome:email`.

De seguida descreve-se o conteúdo de um possível ficheiro textual que representa um determinado estado inicial da biblioteca:

```
DVD:Era␣uma␣vez␣na␣Amadora:Fernando␣Fonseca:20:FICTION:200505550:10
BOOK:A␣arte␣de␣sobreviver␣no␣36:Joao␣Fonseca:20:FICTION:1234567892:22
BOOK:Bairro␣Alto␣e␣o␣Budismo␣Zen:Zun␣Tse␣Fonseca:20:FICTION:1234567891:50
DVD:48␣Horas␣para␣o␣Exame:Orlando␣Fonseca:12:FICTION:200505553:10
BOOK:Analise␣Matematica␣sem␣Mestre:Carlos␣Fonseca:20:SCITECH:1234567890:5
DVD:Lumiar␣Selvagem:Pedro␣Fonseca:20:FICTION:200505551:5
BOOK:Dicionário␣de␣Programação:Odete␣Fonseca:20:REFERENCE:1234567890:50
USER:Obi-Wan␣Kenobi:obiwan@jedi.org
```

Note-se que o programa **nunca** produz ficheiros com este formato, apenas lê ficheiros com este formato.

6 Execução do Programa e Testes Automáticos

É possível verificar automaticamente o resultado correcto do programa. Cada teste é constituído por dois ou três ficheiros de texto:

- um com extensão `.in` e que representa o utilizador, ou seja, vai conter os dados de entrada a serem processados pelo programa;
- um com extensão `.out` e que representa o resultado esperado da execução do programa para o teste em causa;
- um com extensão `.import` e que representa o estado inicial do sistema. Este ficheiro é opcional e pode não fazer parte de um teste.

Dado um teste constituído pelos ficheiros `test.import`, `test.in` e `test.out`, é possível verificar automaticamente o resultado correcto do programa. Note-se que pode ser necessária a definição apropriada da variável de ambiente `CLASSPATH` (ou da opção equivalente `-cp` do comando `java`), para localizar as classes do programa, incluindo a que contém o método correspondente ao ponto de entrada da aplicação (`bci.app.App.main`). As propriedades são tratadas automaticamente pelo código de apoio.

```
java -cp -Dimport=test.import -Din=test.in -Dout=test.outhyp bci.app.App
```

Caso o teste não tenha o ficheiro de *import*, então a forma de executar o teste é semelhante ao anterior, não especificando a propriedade *import*. Assumindo que aqueles ficheiros estão no directório onde é dado o comando de execução, o programa produz o ficheiro de saída `test.outhyp`. Em caso de sucesso, os ficheiros da saída esperada (`test.out`) e obtida (`test.outhyp`) devem ser iguais. A comparação pode ser feita com o comando:

```
diff -b test.out test.outhyp
```

Este comando não deve produzir qualquer resultado quando os ficheiros são iguais. Note-se, contudo, que cada teste apenas verifica um determinado comportamento da aplicação.

7 Notas de Concretização

Tal como indicado neste documento, algumas classes fornecidas como material de apoio, são de uso obrigatório e não podem ser alteradas. Outras dessas classes são de uso obrigatório e têm de ser alteradas.