TÉCNICO LISBOA

Projecto de Programação com Objectos

27 de Setembro de 2019

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 (ver informação completa nas secções **[Projecto]** e **[Método de Avaliação]** no Fénix):

- Datas: **2019/10/18 23:59** (UML); **2019/11/18 23:59** (intercalar); **2018/12/09 23:59** (final); **2018/12/16-2018/12/20** (teste prático).
- Os diagramas de classe UML são entregues exclusivamente em papel (impressos ou manuscritos) na portaria do Tagus. Diagramas ilegíveis ou sem identificação completa serão sumariamente ignorados. É obrigatório a identificação do grupo e turno de laboratório (dia, hora e sala).
- **Apenas se consideram para avaliação os projetos submetidos no Fénix.** As classes criadas de acordo com as especificações fornecidas devem ser empacotadas num arquivo de nome `proj.jar` (que deverá apenas conter os ficheiros `.java` do código realizado guardados nos *packages* correctos). O ficheiro `proj.jar` deve ser entregue para avaliação através da ligação presente no Fénix. É possível realizar múltiplas entregas do projecto até à data limite, mas apenas será avaliada a última entrega.
- 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 execução do projecto terão como resultado a exclusão dos alunos implicados do processo de avaliação**.

O objectivo do projecto é desenvolver um sistema para gerir o acervo de uma mediateca. O sistema deverá permitir, entre outras operações, (i) fazer pesquisas de obras; (ii) registar dados de utentes; (iii) registar dados de obras; e (iv) registar requisições de obras para consulta domiciliária.

A secção 1 apresenta as entidades do domínio da aplicação a desenvolver. As funcionalidades da aplicação a desenvolver são descritas nas secções 3, 4 e 5. 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: obras e suas categorias, utentes, requisições e tempo.

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 Obras e Categorias

O sistema mantém um registo de obras da mediateca. Cada obra é identificada por um número de obra. O identificador é atribuído automaticamente e incrementalmente (a partir de 0 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 mediateca (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. Deve ser possível criar novas categorias, com um impacto mínimo sobre o sistema desenvolvido.

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: autor (apenas um, por simplicidade), e ISBN (cadeia com dez caracteres);

- DVDs – Para cada DVD, o sistema deverá manter: realizador (apenas um, por simplicidade), e o número de registo na IGAC (Inspecção-Geral das Actividades Culturais) (cadeia com dez 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 Utentes

O sistema mantém um registo de utentes da mediateca. Cada utente é identificado por um número de utente. O identificador é atribuído automaticamente e incrementalmente (a partir de 0 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 mediateca: (i) activo, i.e., o utente pode fazer requisições; (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 até devolver a obra e pagar a multa referente ao atraso na entrega.

A mediateca 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 do utente.

O comportamento (no que diz respeito a entrega, dentro ou fora do prazo, de obras requisitadas antes da suspensão) de um utente suspenso influencia a sua classificação após o pagamento da multa. Por exemplo, se, antes de pagar a multa, o utente entregar 3 obras dentro do prazo e regularizar a sua situação, então passará a ser um utente que pode efectuar requisições com a classificação de normal. Deve ser possível discriminar os utentes quanto à sua conduta, considerando outros critérios ou novas classificações, com um impacto mínimo na implementação desenvolvida.

1.3 Requisições

O sistema garante o cumprimento de regras para a requisição de obras. As regras dependem das características da obra que se pretende requisitar e da conduta passada do utente. Utentes cumpridores estão sujeitos a regras permissivas enquanto que a utentes faltosos são aplicadas regras restritivas. Os restantes utentes seguem um conjunto de regras de base. As regras gerais (a verificar pela ordem indicada) a respeitar pelos utentes são:

1. Não pode requisitar duas vezes a mesma obra (i.e., em duas requisições diferentes e simultaneamente abertas);
2. Não pode requisitar obras um utente que esteja suspenso;
3. Não pode requisitar obras cujos exemplares tenham sido já todos requisitados;
4. Não pode ter mais que n obras requisitadas em cada momento (valor base: 3; utentes cumpridores: 5; utentes faltosos: 1);
5. Não pode requisitar obras de referência;
6. 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 algum exemplar seja devolvido. 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 mediateca 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.

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 implementação desenvolvida.

1.4 Gestão de Tempo

A unidade de tempo do sistema é o dia. A data do sistema começa no dia 0 (zero) e faz parte do estado persistente. Sempre que a data é alterada, deve ser verificada a situação dos utentes.

1.5 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 todos as entidades que demonstraram interesse nessa operação.
- Quando uma obra é devolvida, quer-se enviar uma notificação a todos as entidades que demonstraram interesse nessa operação.

A apresentação das notificações enviadas para um dado utilizador deve ser feita quando se visualiza o utilizador (ver abaixo). Quando se faz a visualização, são anexadas à descrição do utilizador (no final) todas as notificações recebidas. 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 diferença relativamente aos vários pedidos de notificações. No caso de notificações relativamente a obras emprestadas, quer-se receber uma notificação sempre que a obra indicada é requisitada. No caso dos pedidos de notificação relativos à devolução de obras, apenas se quer receber uma notificação quando a obra indicada for devolvida.

2 Requisitos de Desenho

Devem ser possíveis extensões ou alterações de funcionalidade com impacto mínimo no código já produzido para a aplicação. O objectivo é aumentar a flexibilidade da aplicação relativamente ao suporte de novas funções. Assim, deve ser possível:

- Adicionar novos tipos (e.g., CDs ou VHS) e novas categorias de obras;
- Definir novas entidades que desejem ser notificadas da requisição ou devolução de obras;
- Introduzir alterações nas regras para requisição de obras ou nos prazos de requisição permitidos;
- Definir novas classificações para os utentes, para além de faltoso, cumpridor ou normal.

Embora na especificação actual não seja possível a remoção de obras ou de utilizadores, a inclusão destas funcionalidades deve ser prevista, por forma a minimizar o impacto da sua futura inclusão.

3 Funcionalidade da Aplicação

A aplicação permite manter informação sobre as entidades do modelo, permitindo, em particular, gerir utentes, obras e empréstimos. Possui ainda a capacidade de preservar o seu estado (não é possível manter várias versões do estado da aplicação em simultâneo).

A base de dados com os conceitos pré-definidos é carregada no início da aplicação. Não é possível remover utentes ou obras durante a execução da aplicação.

Note-se que não é necessário concretizar de raiz a aplicação. Já é fornecido algum código, nomeadamente, o esqueleto das classes necessárias para concretizar os menus e respectivos comandos a utilizar na aplicação a desenvolver, a classe `m19.app.App`, que representa o ponto de entrada da aplicação a desenvolver, algumas classes do domínio da aplicação e um conjunto de excepções a utilizar durante o desenvolvimento da aplicação. A interface geral do core já está parcialmente concretizada na classe `m19.core.LibraryManager` e outras fornecidas (cujos nomes devem ser mantidos), devendo ser adaptadas onde necessário. É ainda necessário concretizar as restantes classes que suportam a operação da aplicação. A classe `m19.core.LibraryManager` deverá ser finalizada por cada grupo e deverá representar a interface geral do domínio da aplicação desenvolvida. Por esta razão, os comandos a desenvolver na camada de interface com o utilizador devem utilizar exclusivamente esta classe para aceder às funcionalidades suportadas pelas classes do domínio da aplicação (a serem desenvolvidas por cada grupo na package `m19.core`). Desta forma será possível concretizar toda a interacção com o utilizador completamente independente da concretização das entidades do domínio. As excepções a criar na camada de serviços (ou seja nos comandos) já estão todas definidas no package `m19.app.exception`. O package `m19.core.exception` contém algumas excepções a utilizar na camada do domínio. Caso seja necessário podem ser definidas novas excepções neste package para representar situações anómalas que possam ocorrer nas entidades do domínio.

3.1 Serialização

É possível guardar e recuperar o estado actual da aplicação, preservando toda a informação relacionada com a mediateca e que foi descrita na secção 1.

4 Interação com o Utilizador

Descreve-se nesta secção 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). Todos os menus têm 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. As mensagens são produzidas pelos métodos das bibliotecas de suporte (**po-uuilib** e **m19-app**). **Não** podem ser definidas novas mensagens. Potenciais omissões devem ser esclarecidas com o corpo docente antes de qualquer concretização.

Não deve haver código de interacção com o utilizador no núcleo da aplicação (*m19.core*). Desta forma, será possível reutilizar o código do núcleo da aplicação (onde é concretizado o domínio da aplicação) com outras concretizações da interface com o utilizador.

As excepções usadas no código de interacção com o utilizador para descrever situações de erro, excepto se indicado, são subclasses de `pt.tecnico.po.ui.DialogException` e devem ser lançadas pelos comandos (sendo depois tratadas automaticamente pela classe já existente `pt.tecnico.po.ui.Menu`). Estas excepções já estão definidas no package (fornecido) `m19.app.exception`. Outras excepções não devem substituir as fornecidas nos casos descritos.

Note-se que o programa principal e os comandos e menus, a seguir descritos, já estão parcialmente concretizados no package `m19.app` e nos seus sub-packages (por exemplo, `m19.app.main`). Estas classes são de uso obrigatório e estão disponíveis na secção *Projecto* da página da cadeira.

Sempre que existe uma interacção com o utilizador, seja para pedir dados seja para apresentar dados, é indicado (caso seja necessário) qual é o método da classe `Message` do package em causa que é responsável por devolver a mensagem a apresentar ao utilizador.

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**. Inicialmente, a aplicação apenas tem informação sobre as entidades que foram carregados no arranque. As etiquetas das opções deste menu estão definidas na classe `m19.app.main.Label`. Todos os métodos correspondentes às mensagens de diálogo para este menu estão definidos na classe `m19.app.main.Message`.

Os comandos que vão concretizar as funcionalidades deste menu já estão parcialmente concretizados nas várias classes do package `m19.app.main`: `DoOpen`, `DoSave`, `DoDisplayDate`, `DoAdvanceDate`, `DoOpenUsersMenu`, `DoOpenWorksMenu` e `DoOpenWorksMenu`.

4.1.1 Salvaguarda do estado actual

O conteúdo da aplicação (toda a informação detida pela mediateca actualmente em memória) 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 `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 (inclui todas as entidades do domínio da aplicação) no ficheiro associado. Se não existir associação, pede-se o nome do ficheiro a utilizar (utilizando a cadeia de caracteres devolvida por `newSaveAs()`), ficando a ele associado.

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 `currentDate()`.

4.1.3 Avançar data actual

O número de dias a avançar é pedido utilizando a mensagem devolvida por `requestDaysToAdvance()`. 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:

Portal 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 a base de dados de utentes damediateca. A lista completa é a seguinte: **Registar utente**, **Mostrar utente**, **Mostrar utentes**, **Mostrar notificações do utente** e **Pagar multa**.

As etiquetas das opções deste menu estão definidas na entidade `m19.app.users.Label`. Todos os métodos correspondentes às mensagens de diálogo para este menu estão definidos em `m19.app.users.Message`.

Sempre que for pedido o identificador de um utente (utilizando a mensagem devolvida por `requestUserId()`) e o utente não existir deve ser lançada a excepção `NoSuchUserException`.

Os comandos deste menu já estão concretizados nas classes do package `m19.app.users`: `DoRegisterUser`, `DoShowUser`, `DoShowUserNotifications`, `DoShowUsers` e `DoPayFine`.

4.2.1 Registar utente

Pede o nome (`requestUserName()`) e o endereço de correio electrónico (`requestUserEMail()`). O registo bem sucedido é assinalado através da mensagem devolvida por `userRegistrationSuccessful()`; 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). A multa a apresentar, para utentes suspensos, é um valor inteiro. Os formatos de apresentação de um utente activo e suspenso são os seguinte:

```
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 utentes

Apresenta informações sobre todas os utentes registados na mediateca, 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.

4.2.4 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 §1.5):

```
ENTREGA: descrição de obra entregue
REQUISIÇÃO: descrição de obra requisitada
```

Exemplos de notificações

```
ENTREGA: 4 - 2 - DVD - Casamento Real - 8 - Ficção - António Fonseca - 200400500
REQUISIÇÃO: 5 - 4 - 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).

4.2.5 Pagar multa

Pede o identificador do utente cuja multa deve ser paga. Se o utente estiver suspenso, a multa é saldada e o utente passa a poder requisitar obras, de acordo com as regras gerais. Se o utente não estiver suspenso, i.e., não tem multas por saldar, 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` e `Efectuar pesquisa`. Os comandos que concretizam estas operações já estão parcialmente concretizados nas classes do package `m19.app.works`: `DoShowWork`, `DoShowWorks` e `DoPerformSearch`.

As etiquetas das opções deste menu estão definidas na classe `m19.app.works.Label`. Todos os métodos correspondentes às mensagens de diálogo para este menu estão definidos na classe `m19.app.works.Message`.

Sempre que seja pedido o identificador da obra (`requestWorkId()`), deve ser lançada a excepção `NoSuchWorkException` caso a obra indicada não existir.

4.3.1 Mostrar obra

É pedido o identificador da obra (`requestWorkId()`). 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 ao autor 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 - 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 Efectuar pesquisa

Esta opção realiza uma procura por termo (cadeia de caracteres), pedido através de `requestSearchTerm()`. 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, o autor 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␣-␣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 `m19.app.requests`, respectivamente: `DoRequestWork`, `DoReturnWork`.

As etiquetas das opções deste menu estão definidas na classe `m19.app.requests.Label`. Todos os métodos correspondentes às mensagens de diálogo para este menu estão definidos na classe `m19.app.requests.Message`.

Sempre que é pedido o identificador do utente (`requestUserId()`), é lançada a excepção `NoSuchUserException`, se o utente indicado não existir. Sempre que é pedido o identificador da obra (`requestWorkId()`), é lançada a excepção `NoSuchWorkException`, se a obra indicada não existir.

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 `RuleFailedException` (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 `requestReturnNotificationPreference()`, se deseja ser notificado acerca da devolução. Utiliza-se a mensagem devolvida por `workReturnDay()` para comunicar o prazo de devolução, em caso de requisição bem sucedida.

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 `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 `requestFinePaymentChoice()`. 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`. Este ficheiro contém a descrição das obras e utentes a carregar no estado inicial da aplicação. 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`.

As obras da mediateca têm o formato descrito abaixo, respectivamente, para DVDs e livros. Assume-se que os títulos das obras não podem conter o carácter **:** e que o preço é um número inteiro (sugere-se a utilização do método `String.split` para o processamento preliminar destas linhas).

Cada obra é descrita num ficheiro textual utilizando o seguinte formato genérico:

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

O campo *exemplares* indica o número de exemplares da obra disponíveis na mediateca. 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 mediateca:

```
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
```

6 Execução dos Programas e Testes Automáticos

Usando os 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 (`m19.app.App.main`. As propriedades são tratadas automaticamente pelo código de apoio.

```
java -Dimport=test.import -Din=test.in -Dout=test.outhyp m19.app.App
```

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 das saídas 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 este teste não garante o correcto funcionamento do código desenvolvido, apenas verifica alguns aspectos da sua funcionalidade.

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.

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).