

# Programação com Objectos/Projecto de Programação com Objectos/Enunciado do Projecto de 2025-2026

From Wiki\*\*3

< [Programação com Objectos](#) | [Projecto de Programação com Objectos](#)

AVISOS - Avaliação em Época Normal	[Expand]
Material de Uso Obrigatório	[Expand]

## ÉPOCA NORMAL

O objectivo do projecto é desenvolver um sistema para gerir o acervo de uma biblioteca. 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.

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).

## Conceitos e Relações do Modelo

Existem vários conceitos importantes neste contexto: criadores, obras, utentes e requisições.

Os conceitos listados não são os únicos possíveis no modelo e as suas relações (assim como relações com outros conceitos não mencionados) podem depender das escolhas do projecto.

## 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 mantém registo das 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, 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.

## 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

# Contents

- 1 Conceitos e Relações do Modelo
  - 1.1 Criadores
  - 1.2 Obras
  - 1.3 Utentes
  - 1.4 Comportamento dos Utentes
  - 1.5 Requisições
  - 1.6 Pesquisas
  - 1.7 Alterações de Inventário de Obras
  - 1.8 Gestão de Tempo
  - 1.9 Notificações
- 2 Requisitos de Desenho
- 3 Funcionalidade da aplicação
  - 3.1 Serialização
- 4 Interacção com o utilizador
  - 4.1 Menu Principal
    - 4.1.1 Salva-guarda do estado actual da aplicação
    - 4.1.2 Mostrar data actual
    - 4.1.3 Avançar data actual
    - 4.1.4 Gestão e consulta de dados da aplicação
  - 4.2 Menu de Gestão de Utentes
    - 4.2.1 Registar utente
    - 4.2.2 Mostrar utente
    - 4.2.3 Mostrar notificações do utente
    - 4.2.4 Mostrar utentes
    - 4.2.5 Pagar multa
  - 4.3 Menu de Gestão de Obras
    - 4.3.1 Mostrar obra
    - 4.3.2 Mostrar obras
    - 4.3.3 Mostrar obras de criador
    - 4.3.4 Alterar inventário de uma obra
    - 4.3.5 Efectuar pesquisa
  - 4.4 Menu de Gestão de Requisições
    - 4.4.1 Requisitar obra
    - 4.4.2 Devolver obra
- 5 Leitura de Dados a Partir de Ficheiros Textuais
- 6 Execução do Programa e Testes Automáticos

## 7 Notas de Implementação

as seguintes categorias: (i) obras de referência (e.g., dicionários, gramáticas, enciclopédias e documentários); (ii) obras de ficção; e (iii) obras técnicas e científicas. Deve ser possível adicionar novas categorias com impacto mínimo na aplicação já desenvolvida.

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

- Livros – Autores (um ou mais criadores), e ISBN (cadeia com 10 ou 13 caracteres).
- DVDs – Realizador (apenas um criador), e número de registo na IGAC (Inspeção-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.

## 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). Cada utente tem ainda informação sobre 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; (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.

## Comportamento dos Utentes

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 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.

## 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 gerais a respeitar pelos utentes são (os números de ordem são os identificadores das regras correspondentes):

1. Não pode requisitar duas vezes a mesma obra (i.e., em duas requisições diferentes e simultaneamente abertas);
2. Não pode estar 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);

As regras devem ser verificadas por ordem. 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.

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.

## Pesquisas

Para permitir que o sistema ajude os utentes a determinar a existência de uma obra, deverá ser possível efectuar pesquisas. As pesquisas consideram os campos relevantes das várias obras. Deve ser possível introduzir novos métodos de pesquisa de uma obra com um impacto mínimo na

implementação desenvolvida.

## 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 (os prazos de entrega só afectam requisições futuras); 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.

## 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. Sempre que a data é alterada, deve ser verificada a situação dos utentes.

## 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 todos 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 quando 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.

## Requisitos de Desenho

A aplicação 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 da aplicação.

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);
- Definir novas entidades que desejem ser notificadas da requisição ou disponibilidade 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;
- 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.

## 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 aplicação deve estar preparada para que se possa remover qualquer entidade (não apenas obras).

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.

💡 Note-se que não é necessário implementar de raiz a aplicação: já existem classes que representam e definem a interface geral da funcionalidade do *core* da aplicação, tal como é visível pelos comandos da aplicação.

🔗 A interface geral do *core* já está parcialmente implementada na classe **bci.LibraryManager** e outras fornecidas (cujos nomes devem ser mantidos), devendo ser adaptadas onde necessário. É ainda necessário criar e implementar as restantes classes que suportam a operação da aplicação.

## Serialização

É possível guardar e recuperar o estado actual da aplicação, preservando toda a informação relacionada com a biblioteca e que foi descrita acima.

## Interacção 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 procederem 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-uilib** e **bci-app**). As mensagens não podem ser usadas no núcleo da aplicação (**hva-core**). Além disso, não podem ser definidas novas. Potenciais omissões devem ser esclarecidas antes de qualquer implementação.


De um modo geral, sempre que no contexto de uma operação com o utilizador aconteça alguma excepção, então a operação não deve ter qualquer efeito no estado da aplicação, excepto em caso de indicação contrária na operação em causa. As excepções estão na package **bci.app.exceptions**, excepto em caso de indicação contrária.

As excepções usadas na interacção (subclasses de **pt.tecnico.uilib.menus.CommandException**), excepto em caso de indicação contrária, são lançadas pelos comandos (subclasses de **pt.tecnico.uilib.menus.Command**) e tratadas pelos menus (instâncias de subclasses de **pt.tecnico.uilib.menus.Menu**). Outras excepções não devem substituir as fornecidas nos casos descritos.

Nos pedidos e usos dos vários identificadores, podem ocorrer as seguintes excepções, caso o identificador indicado não corresponda a um objecto conhecido (excepto no processo de registo ou em caso de indicação contrária). Note-se que estas excepções não são utilizáveis no núcleo da aplicação.


Tipo de dados	Classe	Pedido a apresentar	Excepção a lançar se desconhecido
Criador	<b>Creator</b>	<b>bci.app.work.Prompt.creatorId()</b>	<b>bci.app.exceptions.NoSuchCreatorException</b>
Obra	<b>Work</b>	<b>bci.app.work.Prompt.workId()</b>	<b>bci.app.exceptions.NoSuchWorkException</b>
Utente	<b>User</b>	<b>bci.app.user.Prompt.userId()</b>	<b>bci.app.exceptions.NoSuchUserException</b>


Alguns casos particulares podem usar pedidos específicos não apresentados nesta tabela.

 Note-se que o programa principal e os comandos e menus, a seguir descritos, já estão parcial ou completamente implementados nas *packages* **bci.app**, **bci.app.main**, **bci.app.requests**, **bci.app.users**. Estas classes são de uso obrigatório e estão disponíveis no [GIT](#) (módulo **bci-app**).

## Menu Principal

As acções deste menu permitem gerir a salvaguarda do estado da aplicação 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 **bci.app.main.Label**. Todos os métodos correspondentes às mensagens de diálogo para este menu estão definidos na classe **bci.app.main.Message**.

 Estes comandos já estão implementados nas classes da *package* **bci.app.main** (disponível no CVS), respectivamente: **DoOpen**, **DoSave**, **DoDisplayDate**, **DoAdvanceDate**, **DoOpenUsersMenu**, **DoOpenWorksMenu**, **DoOpenRequestsMenu**.

## Salvaguarda do estado actual da aplicação

O conteúdo da aplicação (toda a informação detida pela biblioteca 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 (**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, ficando a ele associado (para operações de salvaguarda subsequentes). Esta interacção realiza-se através do método **Prompt.newSaveAs()**. Não é executada nenhuma acção se não existirem alterações desde a última salvaguarda.

Quando se abandona uma aplicação com modificações não guardadas (porque se abre outra), deve perguntar-se se se quer guardar a informação actual antes de a abandonar, através de **Prompt.saveBeforeExit()** (a resposta é obtida invocando **readBoolean()** ou de **Form.confirm()**).

💡 Note-se que a opção **Abrir** não permite a leitura de ficheiros de texto (estes apenas podem ser utilizados no início da aplicação).

💡 A opção **Sair** nunca implica a salvaguarda do estado da aplicação, mesmo que existam alterações.

## Mostrar data actual

A data actual do sistema é apresentada através da mensagem **Message.currentDate()**.

## Avançar data actual

O número de dias a avançar é pedido através de **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.

## Gestão e consulta de dados da aplicação

- **Menu de Gestão de Utes** -- 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.

## Menu de Gestão de Utes

Este menu permite efectuar operações sobre a base de dados de utentes da biblioteca. A lista completa é a seguinte: [Registar utente](#), [Mostrar utente](#), [Mostrar utentes](#), [Mostrar notificações do utente](#), [Pagar multa](#).

📄 As etiquetas das opções deste menu estão definidas em **bci.app.users.Label**. Os métodos correspondentes às mensagens de diálogo para este menu estão definidos em **bci.app.users.Prompt** e **bci.app.users.Message**.

📄 Estes comandos já estão implementados nas classes da *package* **bci.app.users** (disponível no CVS), respectivamente: **DoRegisterUser**, **DoShowUser**, **DoShowUserNotifications**, **DoShowUsers**, **DoPayFine**.




## Registar utente

Pede o nome (**Prompt.userName()**) e o endereço de correio electrónico (**Prompt.userEMail()**). O registo bem sucedido é assinalado através da mensagem **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.

## 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.

 **Formatos de apresentação de um utente (activos e suspensos)**


[Expand]

 **Exemplos de apresentação de utentes**

[Expand]

## Mostrar notificações do utente

É pedido o identificador do utente, sendo apresentadas as notificações para esse utente, de acordo com o seguinte formato (correspondente aos casos descritos acima):

 **Formatos de apresentação das notificações de um utente**

[Expand]

 **Exemplos de notificações**

[Expand]

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

## Mostrar utentes

Apresenta informações sobre todos os utentes, ordenando-os lexicograficamente pelo nome. Caso existam utentes com o mesmo nome, devem ser ordenados por ordem crescente dos seus identificadores. O formato é o descrito em [Mostrar utente](#).

## 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, i.e., não tem multas por saldar, deve lançar-se uma excepção **UserIsActiveException**.

## 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](#).

 As etiquetas das opções deste menu estão definidas em **bci.app.work.Label**. Os métodos correspondentes às mensagens de diálogo para este menu estão definidos em **bci.app.work.Prompt** e **bci.app.work.Message**.

 Estes comandos já estão implementados nas classes da *package* **bci.app.work** (disponível no GIT), respectivamente: **DoDisplayWork**, **DoDisplayWorks**, **DoDisplayWorksByCreator**, **DoChangeWorkInventory**, **DoPerformSearch**.

## Mostrar obra

É pedido o identificador da obra. Se a obra existir, é apresentada de acordo com os seguintes formatos (para livros e DVDs).

O formato genérico de apresentação da obras é como se segue:

### Formato de apresentação de obras

[\[Expand\]](#)

Para livros, a informação adicional corresponde à lista dos 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

[\[Expand\]](#)

## Mostrar obras

Apresenta informações sobre todas as obras, ordenando-as pelos seus identificadores. O formato de apresentação é como descrito em [Mostrar obra](#).

## 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 é como descrito em [Mostrar obra](#).

## Alterar inventário de uma obra

É pedido o identificador 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()**.

## 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 para [Mostrar obra](#).

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 acima, uma pesquisa pelo termo **casa** retornaria as obras com os identificadores 3, 4 e 5.

 Resultados de pesquisa pelo termo "casa"


[Expand]

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

## Menu de Gestão de Requisições

Este menu apresenta as operações relacionadas com requisições de obras. A lista completa é a seguinte: [Requisitar obra](#), [Devolver obra](#).

 As etiquetas das opções deste menu estão definidas em **bci.app.request.Label**. Os métodos correspondentes às mensagens de diálogo para este menu estão definidos em **bci.app.request.Prompt** e **bci.app.request.Message**.

 Estes comandos já estão implementados nas classes da *package* **hva.app.animal** (disponível no GIT), respectivamente: **DoRequestWork**, **DoReturnWork**.

### 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 **Prompt.returnNotificationPreference()**, se deseja ser notificado acerca da devolução. Utiliza-se a mensagem **Message.workReturnDay()** para comunicar o prazo de devolução, em caso de requisição bem sucedida.

### 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 uma excepção **WorkNotBorrowedByUserException**. Caso contrário, o sistema processa a entrega e, caso haja lugar ao pagamento de multa, é apresentada a mensagem **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 **Prompt.finePaymentChoice()**. Se a resposta for positiva, a multa é liquidada e o utente fica activo, caso não tenha nenhuma obra por entregar fora de prazo.

## Leitura de Dados a Partir de Ficheiros Textuais

Além das opções de manipulação de ficheiros descritas no [menu principal](#), é possível iniciar a aplicação com um ficheiro de texto especificado pela propriedade Java **import**.

As obras da biblioteca 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. Não existem entradas mal-formadas.

Cada linha tem uma descrição distinta mas que segue o seguinte formato geral. O campo "exemplares" indica o número de exemplares da obra disponíveis na biblioteca.

```
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
```

É ainda possível definir utentes, de acordo com o seguinte formato:

```
USER:nome:email
```

Um exemplo de conteúdo do ficheiro inicial é como se segue:

 **Exemplo de ficheiro de entrada textual**

[Expand]

A codificação dos ficheiros a ler é garantidamente UTF-8.

💡 Usar o método **`String.split()`** para partir cadeias de caracteres e **`String.trim()`** para remover espaços no início e no final.

💡 Note-se que o programa nunca produz ficheiros com este formato (é apenas um formato para ficheiros de entrada).

## Execução do Programa 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 é necessária a definição apropriada da variável **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 -Dimport=test.import -Din=test.in -Dout=test.outhyp bci.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 verificando alguns aspectos da sua funcionalidade.

## Notas de Implementação

Tal como indicado acima, 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).

Categories:

- Ensino
- PO
- Projecto de PO

This page was last edited on 3 September 2025, at 18:28.

[Privacy policy](#)

[About Wiki\\*\\*3](#)

[Disclaimers](#)

**Powered by MediaWiki**