# #FindConcertTicket

*Programação Orientada pelos Objectos*

*Enunciado do 2º Trabalho Prático, versão 1.5 – 2017-05-23*

*Contacto: jmc.cunha@fct.unl.pt*

**Alterações em relação à versão 1.0:** Alterações de texto são apresentadas a **verde**.

**Alterações em relação à versão 1.3:** Alterações de texto são apresentadas a **azul**.

**Alterações em relação à versão 1.4:** Foi eliminada uma frase, apresentada a laranja.

## Notas importantes:

**Prazo de entrega:** até às 23h59 do dia 01 de Junho de 2016.

**Realização em grupos:** grupos de 2 alunos de qualquer turno.

**Material a entregar:**

- **Moodle**: ficheiro zip submetido via Moodle contendo o **projecto eclipse completo** da aplicação desenvolvida, com código fonte devidamente comentado, o respectivo **Javadoc** e o **diagrama de classes e interfaces** (formato *jpg*). O nome do ficheiro compactado tem as mesmas regras que nos trabalhos práticos anteriores.
- **Mooshak**: submissão e aceitação do código fonte pelo sistema de avaliação automática Mooshak.

**Recomendações:** A boa documentação do seu código fonte será valorizada, tal como a utilização do melhor estilo de programação possível, para além, claro, do correcto funcionamento do projecto. Não se esqueça de comentar as declarações das classes e das interfaces com uma explicação do significado das mesmas.

## 1. Desenvolvimento de aplicação #FindConcertTicket

### 1.1. Descrição do problema

O objectivo deste trabalho é o desenvolvimento de uma aplicação – #FindConcertTicket – para a gestão de espectáculos musicais e respetiva venda de bilhetes. A aplicação #FindConcertTicket deve permitir:

- Registar e manter informação sobre espectáculos musicais: concertos e festivais;
- Registar e manter informação sobre bilhetes vendidos e disponíveis;
- Gerir os utilizadores da aplicação.

A aplicação suporta três tipos de utilizadores: *Admin*, *Client* e ainda utilizadores não registados. Os utilizadores *Admin* administram o sistema e podem inserir os espectáculos e listar toda a informação sobre os utilizadores registados. Os utilizadores *Client* representam os utilizadores registados que podem assim adquirir bilhetes para espectáculos. Os utilizadores não registados representam os visitantes da aplicação, que não podem comprar bilhetes, mas podem pesquisar e listar espectáculos (estes utilizadores não precisam de iniciar uma sessão no sistema).

Um utilizador pode registar-se como *Admin* e nesse caso pode inserir os espectáculos. Um utilizador que se registe como *Client* está autorizado a comprar bilhetes para espectáculos. Os utilizadores registados são identificados pelo seu email e uma password gerada pelo sistema.

### Dados sobre os utilizadores

Cada utilizador é caracterizado pelo seu username e password. Considere que a password devolvida para um utilizador *Admin* é a string "admin1" para o primeiro utilizador *Admin* registado, "admin2" para o segundo utilizador *Admin* registado e assim sucessivamente. Para os utilizadores *Client* a regra é similar, mas neste caso é usado o prefixo "client".

### Dados sobre os espectáculos

Os espectáculos musicais incluem concertos individuais de artistas/bandas e festivais (por exemplo festivais de verão).

- **Concerto.** Cada concerto tem os seguintes dados associados: (1) um nome; (2) um artista/banda; (3) uma descrição; (4) a data em que decorrerá; (5) um preço; (6) o número de bilhetes disponíveis. Considere que cada concerto é univocamente identificado pelo nome do artista e data.
- **Festival.** Cada festival tem os seguintes dados: (1) um nome; (2) um alinhamento, por dia, dos artistas; (3) uma descrição; (4) um intervalo de datas em que decorrerá; (5) vários preços, para os diferentes números de dias (por exemplo, preço para 1 dia, para 2 dias, etc.); (6) o número de bilhetes disponíveis por dia. Considere que cada festival é univocamente identificado pelo seu nome e data de início.

Independentemente do tipo, é sempre necessário que cada espectáculo registe os utilizadores que compraram bilhetes para o espectáculo.

Para representar as datas use a classe `java.time.LocalDate`.

### Dados sobre os artistas/bandas

- **Artista.** Para cada artista existem os seguintes dados: (1) nome; (2) discografia, neste caso, apenas a lista dos nomes dos álbuns publicados.
- **Banda.** Para uma banda existem os seguintes dados: (1) nome; (2) discografia, neste caso, apenas a lista de álbuns publicados; (3) nome dos elementos da banda.

### Dados sobre os bilhetes

Cada bilhete tem ~~código de barras (representado por uma string),~~ um espectáculo associado, e o preço que custou.

- **Concerto.** Os bilhetes para concertos têm o número de pessoas que podem usufruir do bilhete, uma vez que são bilhetes multi-espectadores.
- **Festival.** Os bilhetes para festivais têm indicação dos dias (datas) que o espectador pode assistir ao festival.

## 2. Comandos a implementar

A aplicação deve interagir com o utilizador por meio de menus de comandos. Caso seja introduzido um comando desconhecido, a aplicação deve retornar a mensagem (`Unknown command`).

A aplicação e o seu menu de comandos prevêem três níveis de utilizadores: (1) utilizadores não registados, que consultam informação sobre os espectáculos musicais disponíveis; (2) utilizadores registados como *Admin* que podem inserir os espectáculos; (3) utilizadores registados como *Client* que podem comprar bilhetes.

Para simplificar, considera-se que somente um utilizador pode ter uma sessão aberta no sistema em cada instante – *Admin*, *Client* ou utilizador não registado. O sistema disponibiliza opções diferentes a utilizadores diferentes – há, pois, três "modos" de utilização diferentes que oferecem opções de menu diferentes: um para utilizadores registados como *Admin*, outro para utilizadores registados como *Client* e por último um para utilizadores não registados.

Para se executar um comando acessível a um tipo de utilizador diferente do actual, é necessário que um faça *logout* e outro faça *login*.

Sempre que é necessário apresentar informação sobre um espectáculo, esta deve ser a seguinte:

- **Concertos**: nome do concerto, nome do artista, a data, o preço do bilhete e o número de bilhetes disponíveis.
- **Festivais**: nome, os nomes dos artistas seriados pela ordem de inserção e por dia, a data de início e de fim, o preço por número de dias e o número de bilhetes disponíveis por dia.

No caso de a operação ser realizado por um utilizador do tipo *Admin*, são também listados os emails dos espectadores que compraram bilhete para o concerto, bem como o número de espectadores por bilhete.

A não ser que seja dito o contrário, os espectáculos são seriados por data de início, depois por nome.

## 2.1. Comandos comuns a todos utilizadores, inclusive utilizadores não registados

1. **Listar todos os espectáculos** (comando `SHOWS`).
   A operação apresenta todos os espectáculos seriados pela ordem de entrada no sistema. O cabeçalho da listagem a produzir é a mensagem (`All shows:`), mesmo que não existam espectáculos a listar. O mesmo acontece com as várias listagens. A operação tem sempre sucesso.
2. **Listar espectáculos por número de espectadores** (comando `SHOWSBYCLIENTS`).
   A operação apresenta os espectáculos seriados por número de bilhetes vendidos. O cálculo do número de bilhetes vendidos por concerto deve ter em consideração o número de espectadores por bilhete, e para festival é a soma da venda de todos os bilhetes de todos os dias. Em caso de empate é usada a data do espectáculo, da mais próxima para a mais distante, e finalmente a ordem alfabética do nome do ~~artista, ou do primeiro artista, no caso de festivais~~ concerto/festival. O cabeçalho da listagem a produzir é a mensagem (`Most sold shows:`). A operação tem sempre sucesso.
3. **Seriar espectáculos por tipo** (comando `SHOWSBYTYPE`).
   É fornecido o tipo (concerto ou festival) para selecionar os espectáculos. O cabeçalho da listagem a produzir é a mensagem (`Concerts:`) ou (`Festivals:`), dependendo do pedido. A operação falha se o tipo de espectáculo não for conhecido (`Unknown type of show`).
4. **Consultar os dados de um espectáculo** (comando `SHOW`)**.**
   É fornecido o nome ~~do artista~~ de um concerto ou nome de um festival e a data de início. O cabeçalho da listagem a produzir é a mensagem (`<show> on <date>:`). A operação falha se: (1) o espectáculo não existir (`Show does not exist`).
5. **Pesquisa por artista** (comando `SEARCH`)
   É fornecido o nome de um artista e devem ser listados todos os espectáculos onde este atue, quer de forma individual, quer em festival. Note-se que aqui não se devem considerar os elementos de bandas na pesquisa. Devem ser seriados primeiro os concertos e depois os festivais. O cabeçalho da listagem é a mensagem (`Concerts of <artist>:`) para os concertos e depois (`Festivals where <artist> will play:`) para os festivais.
6. **Registar utilizador** (comando `REGISTER`)
   São fornecidos o tipo de utilizador (`Admin` ou `Client`) e o *email*. Em caso de sucesso é criado um novo utilizador no sistema e devolvida a sua password (`User was registered: <password>`). A operação falha se: (1) existir um utilizador *logged in* (`User already logged in`); (2) já existir um utilizador com o *email* dado (`User already exists`).
7. **Sair da aplicação** (comando `EXIT`)**.** A operação tem sempre sucesso (`Exiting`).

## 2.2. Comandos apenas para os utilizadores `Admin`

1. **Adicionar um artista/banda** (comando `ADDARTIST`)**.**
   É fornecido o nome do artista/banda, o número de álbuns, a lista de álbuns publicados e o número de elementos. Se o número for superior a 1, trata-se de uma banda e são fornecidos os nomes dos elementos da banda. Em caso de sucesso o artista é registado (`Artist added`). A operação falha se:

(1) não existir um utilizador do tipo `Admin` com a sessão iniciada (`User cannot execute this command`); (2) já existir um artista com esse nome (`Artist name already exists`).

2. **Adicionar um espectáculo** (comando `ADDSHOW`)**.**
   É fornecido o nome, a descrição e o número de bilhetes disponíveis. A aplicação deve depois perguntar se é um concerto ou um festival. No caso de ser um concerto, deve ser fornecido o nome do artista, a data, e o preço do bilhete. No caso de um festival, deve ser fornecido o número de dias do festival, além da data de início ~~e fim~~ do festival. Deve ser definido o alinhamento para cada dia, indicando o número de artistas/bandas e os seus nomes. Finalmente, deve ser dada a relação de preços por número de dias. Em caso de sucesso o espectáculo é adicionado (`Show added`). A operação falha se: (1) não existir um utilizador do tipo `Admin` com a sessão iniciada (`User cannot execute this command`); (2) já existir um espectáculo com esse nome nessa data (`Show already exists`); (3) algum dos artistas não existir na aplicação (`Artist name(s) do(es) not exist(s)`), seguido da listagem dos artistas não registados na aplicação.

## 2.3. Comandos apenas para os utilizadores `Client`

1. **Comprar bilhete** (comando `BUYTICKET`)**.**
   São fornecidos o nome do espectáculo e a data (de início) do espectáculo. Depois disso, a aplicação deve perguntar ao utilizador que o evento é um concerto ou um festival. Se for um concerto, é ainda fornecido o número de bilhetes. Se for um festival, são fornecidas as datas para as quais se quer comprar bilhete. Em caso de sucesso a compra é realizada. Assume-se que o pagamento seria efectuado com os meios normais, não sendo necessário implementar tal comportamento. Após a compra é fornecido o valor total (`Ticket bought with cost <cost>`). A operação falha se: (1) não existir um utilizador do tipo `Client` com a sessão iniciada (`User cannot execute this command`); (2) não existir um espectáculo com nome dado na data inicial indicada (`Show does not exist`); (3) não existirem lugares suficientes (`There are not sufficient seats for the request`).
2. **Listar bilhetes** (comando `MYTICKETS`)**.**
   Este comando não tem argumentos. Em caso de sucesso este comando lista todos os bilhetes que o utilizador comprou. Deve seriar os bilhetes por ordem de antiguidade da sua compra, com o cabeçalho (`My Tickets:`), mas listando primeiro os bilhetes de concertos e depois os festivais. A operação sucede sempre.

## 2.4. Comandos comuns aos utilizadores `Admin` e `Client`

2. **Login** (comando `LOGIN`)**.**
   É fornecido ao sistema o *email* e a *password*. Em caso de sucesso é iniciada a sessão associada ao utilizador, sendo apresentada a mensagem (`Welcome <email>`). A operação falha se: (1) o *email* não existir (`User does not exist`); (2) o utilizador já tiver uma sessão aberta (`User already logged in`); (3) já existir outro utilizador com uma sessão aberta (`Another user is logged in`); (4) a senha não corresponder ao *email* dado (`Wrong password`).
2. **Logout** (comando `LOGOUT`)**.**
   Esta operação não requer argumentos. Em caso de sucesso é dada como terminada a sessão associada ao utilizador que estava *logged in*, sendo apresentada a mensagem (`Goodbye <email>`). A operação falha se: (1) não existir um utilizador *logged in* (`No user is logged in`).

## 2.5. Exemplo de interacção com a aplicação

A aplicação desenvolvida tem que garantir o modelo de interacção ilustrado no exemplo seguinte (o carácter ↵ representa uma mudança de linha):

```
REGISTER↵
ADMIN↵
joao@gmail.com↵
User was registered: admin1.↵
↵
```

```
LOGIN↵
joao@gmail.com↵
admin1↵
Welcome joao@gmail.com↵
↵
ADDARTIST↵
Os Azeitonas↵
2↵
Em Boa Companhia Eu Vou↵
AZ↵
3↵
Mario Marlon Brandao↵
Luisa Nena Barbosa↵
Joao Salsa Salcedo↵
Artist added.↵
↵
ADDARTIST↵
Mariza↵
1↵
Mundo↵
1↵
Artist added.↵
↵
ADDSHOW↵
Mariza ao Vivo em Guimaraes↵
O concerto do século! ↵
2000↵
CONCERT OR FESTIVAL?↵
Concert↵
Mariza↵
2017-10-22↵
35↵
Show added.↵
↵
ADDSHOW↵
O Sol da Caparica↵
O festival de Portugal na Margem Sul↵
20000↵
CONCERT OR FESTIVAL?↵
Festival↵
2↵
2017-07-10↵
2↵
Mariza↵
Os Azeitonas↵
1↵
Os Azeitonas↵
1 40↵
2 65↵
Show added.↵
↵
```

**LOGOUT**↵
Goodbye joao@gmail.com↵
↵
**REGISTER**↵
**CLIENT**↵
**maria@gmail.com**↵
User was registered: client1.↵
↵
**LOGIN**
**maria@gmail.com**↵
**client1**↵
Welcome maria@gmail.com↵
↵
**BUYTICKET**↵
**Mariza ao Vivo em Guimaraes**↵
**2017-10-22**↵
CONCERT OR FESTIVAL? ↵
Concert
**3**↵
Ticket bought with cost 105.↵
↵
**BUYTICKET** ↵
**O Sol da Caparica**↵
**2017-07-10**↵
CONCERT OR FESTIVAL? ↵
Festival
**2**↵
**2017-07-10**↵
**2017-07-11**↵
Ticket bought with cost 65.↵
↵
**BUYTICKET** ↵
**O Sol da Caparica**↵
**2017-07-10**↵
CONCERT OR FESTIVAL? ↵
Festival
**1**↵
**2017-07-11**↵
Ticket bought with cost 40.↵
↵
**MYTICKETS**↵
My Tickets:↵
Mariza ao Vivo em Guimaraes↵
2017-10-22↵
3↵
105↵
O Sol da Caparica↵
2017-07-10↵
2017-07-11↵
65↵
O Sol da Caparica↵
2017-07-11↵
40↵

↵
**LOGOUT**↵
Goodbye maria@gmail.com↵
↵
**SHOWS**↵
All shows:↵
Mariza ao Vivo em Guimaraes↵
Mariza ↵
2017-10-22↵
35↵
1997↵
O Sol da Caparica↵
2017-07-10↵
Mariza↵
Os Azeitonas↵
2017-07-11↵
Os Azeitonas↵
2017-07-10↵
2017-07-11↵
1 40
2 65↵
2017-07-10 19999↵
2017-07-11 19998↵
↵
**SHOWSBYCLIENTS**↵
Most sold shows:↵
O Sol da Caparica↵
2017-07-10↵
Mariza↵
Os Azeitonas↵
2017-07-11↵
Os Azeitonas↵
2017-07-10↵
2017-07-11↵
1 40
2 65↵
2017-07-10 19999↵
2017-07-11 19998↵
Mariza ao Vivo em Guimaraes↵
Mariza ↵
2017-10-22↵
35↵
1997↵
↵
**SHOW**↵
**O Sol da Caparica**↵
**2017-07-10**↵
O Sol da Caparica on 2017-07-10:↵
O Sol da Caparica↵
2017-07-10↵
Mariza↵
Os Azeitonas↵

```
2017-07-11↵
```

```
Os Azeitonas↵
2017-07-10↵
2017-07-11↵
1 40
2 65↵
2017-07-10 19999↵
2017-07-11 19998↵
↵
SHOW↵
Mariza ao Vivo em Guimaraes↵
2017-10-22↵
Mariza ao Vivo em Guimaraes on 2017-10-22↵
Mariza ↵
2017-10-22↵
35↵
1997↵
↵
SEARCH↵
Mariza↵
Concerts of Mariza:↵
Mariza ao Vivo em Guimaraes↵
Mariza ↵
2017-10-22↵
35↵
1997↵
Festival where Mariza will play:↵
O Sol da Caparica↵
2017-07-10↵
Mariza↵
Os Azeitonas↵
2017-07-11↵
Os Azeitonas↵
2017-07-10↵
2017-07-11↵
1 40
2 65↵
2017-07-10 19999↵
2017-07-11 19998↵
↵
EXIT↵
Exiting.
```

## 3. Desenvolvimento

A sua aplicação deve tirar o melhor partido possível da matéria leccionada. Em particular, fazê-la **o mais extensível possível**. Deve ser construída de modo a **minimizar as alterações necessárias**, caso se pretendam acrescentar, mais tarde, **novos tipos de espectáculos, utilizadores ou bilhetes**.

Comece por desenvolver a interface principal da sua aplicação, identificando claramente quais os comandos que a sua aplicação deve suportar, bem como quais as entradas e saídas, não esquecendo as pré-condições (a serem verificadas através de excepções). Depois, identifique as entidades de que vai necessitar para implementar este sistema. Identifique e especifique cuidadosamente as **interfaces** e **classes** de que necessita. Deve documentar o

seu desenvolvimento quer através de um diagrama de classes e interfaces, quer através de documentação adequada no seu código.

Construa o esqueleto da classe Main que trata da entrada e saída dos dados e da interacção com a aplicação. É normal que no princípio o seu programa ainda não faça tudo. Lembre-se da **regra da versão estável**: não tente fazer tudo de uma só vez. Vá fazendo, testando, e avançando por incrementos, à medida que as funcionalidades vão sendo desenvolvidas. Se necessário, crie pequenos programas de teste auxiliares. Desenvolva também as operações de forma incremental.

# 4. Submissão ao Mooshak

Para submeter o seu programa ao Mooshak registe-se no concurso **POO1617-TP2** e siga as instruções apresentadas no site Moodle da disciplina.

Note que para cada comando será apresentada apenas uma única mensagem de saída. Ou seja, as condições que causam falhas num comando devem ser verificadas pela ordem descrita no enunciado e assim que uma dessas condições se verificar não é necessário verificar as condições seguintes.

## 4.2. Testes

Os testes do Mooshak verificam de forma incremental a implementação dos vários comandos.

Os testes que terminam em `_ok.txt` não são testadas as condições onde os comandos podem falhar. Enquanto, testes que terminam em `_pre.txt` testam as condições em que os comandos podem falhar.

1. Comandos testados: REGISTER, LOGIN, LOGOUT, EXIT
    - Ficheiro de teste: 01-in-register_loginout_ok.txt (15 pontos)
    - Ficheiro de teste: 01-in-register_loginout_pre.txt (5 pontos)
2. Comandos testados: ADDARTIST, ADDSHOW, SHOWS
    - Ficheiro de teste: 02-in-addartistshow_shows_ok.txt (20 pontos)
    - Ficheiro de teste: 02-in-addartistshow_shows_pre.txt (10 pontos)
3. Comandos testados: SHOWSBYTYPE, SEARCH
    - Ficheiro de teste: 03-in-showsbytype_search_ok.txt (5 pontos)
    - Ficheiro de teste: 03-in-showsbytype_search_pre.txt (5 pontos)
4. Comandos testados: BUYTICKETS, MYTICKETS
    - Ficheiro de teste: 04-in-buymytickets_ok.txt (15 pontos)
    - Ficheiro de teste: 04-in-buymytickets_pre.txt (5 pontos)
5. Comandos testados: todos os comandos
    - Ficheiro de teste: 05-in-everything_ok.txt (15 pontos)
    - Ficheiro de teste: 05-in-everything_pre.txt (5 pontos)