FCT Boleia

Introdução à Programação
Enunciado do 2º Trabalho Prático, versão 1.0 – 2018-11-16

Notas importantes

Trabalho em Grupo de 2 alunos a desenvolver respeitando os princípios do *Código de Ética*.

Prazo de entrega até às 17h00 do dia 7 de Dezembro de 2018.

Submissão e aceitação do código fonte pelo sistema de avaliação automática Mooshak. Consulte na página da disciplina as regras de submissão de trabalhos ao 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 dos seus métodos com uma explicação do seu significado.

1 Introdução

Este documento descreve o 2º trabalho prático da disciplina de Introdução à Programação do 1º ano do Mestrado Integrado em Engenharia Informática.

Os alunos podem e devem tirar todas as dúvidas com os docentes da disciplina, bem como conversar com os colegas e discutir soluções, mas não podem entregar código de que não são autores.

2 Desenvolvimento da Aplicação *FCT Boleia*

O objectivo deste trabalho é o desenvolvimento de um sistema de partilha de carros, em que se oferecem e pedem "boleias".

2.1 Descrição do Problema

Dar e pedir boleia é uma coisa que já provavelmente fez com alguns dos seus amigos: é simplesmente o facto de deslocar-se com pessoas partilhando um carro e os custos da deslocação (combustível e portagens).

Pretende-se criar um sistema/aplicação para facilitar o uso desta forma de viajar.

O sistema/aplicação deve permitir que as pessoas encontrem e/ou partilhem um carro para se deslocarem juntos, "à boleia". Esta aplicação tem como funcionalidades principais:

- registar uma dada deslocação, que poderá ser partilhada;
- seleccionar uma boleia para uma dada deslocação;
- consultar as possíveis deslocações existentes.

Para poder usar estas funcionalidades, o utilizador deve registar-se na aplicação com o seu email. Assim sendo, o programa pode encontrar-se em dois modos:

- inicial: aqui é possível fazer registos de novos utilizadores, iniciar uma sessão com um utilizador e terminar a execução do programa;
- sessão: aqui é possível registar, partilhar e consultar viagens.

A interação do utilizador com o programa é feita sempre através de comandos, descritos na secção seguinte.

3 Comandos

Nesta secção apresentam-se os vários comandos que o sistema deve ser capaz de interpretar e executar. Nos exemplos apresentados, diferenciamos o `texto escrito pelo utilizador` da retroacção escrita pelo programa na consola, ao executar o comando. Pode assumir que o utilizador não cometerá erros na introdução de argumentos nos comandos, para além dos descritos neste enunciado, ou seja, apenas tem de tratar as situações de erro descritas aqui, pela ordem que são descritas.

Na leitura de comandos o interpretador não deve fazer distinção entre maiúsculas e minúsculas. Por exemplo, para o comando `sai`, o interpretador deve aceitar como válidas as alternativas `SAI`, `sai`, `Sai`, `sAI`, etc. Nos vários exemplos que se seguem, o símbolo ↵ denota a mudança de linha.

Caso o utilizador introduza um comando inexistente, o programa escreve na consola a mensagem Comando inexistente.. Por exemplo o comando inexistente `desconhecido` teria este efeito:

```
> desconhecido↵
Comando inexistente.↵
>
```

Vários comandos têm argumentos. Pode assumir que o utilizador apenas usa argumentos em número e de tipos correctos. No entanto, pode acontecer que os argumentos passados a um desses comandos tenham algum valor inválido no contexto do problema. Por esse motivo, teremos de testar esses argumentos exactamente pela ordem especificada neste enunciado.

3.1 Comando `ajuda`

Informa sobre os comandos disponíveis no programa. O comando não necessita de argumentos e tem sempre sucesso. Este comando tem 2 formatos distintos, dependendo do modo em que o programa está. No caso de ainda não ter-se realizado um login (modo inicial), o formato do comando é o seguinte:

```
> ajuda↵
ajuda - Mostra os comandos existentes↵
termina - Termina a execucao do programa↵
regista - Regista um novo utilizador no programa↵
entrada - Permite a entrada ("login") dum utilizador no programa↵
>
```

No caso do programa estar em modo sessão (já foi feito o comando "entrada"), o formato do comando é o seguinte:

```
emailUtilizador > ajuda↵
ajuda - Mostra os comandos existentes↵
sai - Termina a sessao deste utilizador no programa↵
nova - Regista uma nova deslocacao↵
lista - Lista todas ou algumas deslocacoes registadas↵
boleia - Regista uma boleia para uma dada deslocacao↵
consulta - Lista a informacao de uma dada deslocacao↵
remove - Retira uma dada deslocacao↵
emailUtilizador >
```

Note que, quando o programa está em modo sessão no prompt está o email do utilizador que fez login (comando entrada). Caso seja o utilizador com email *fb@fct.unl.pt* a executar o comando ajuda, o formato seria:

```
fb@fct.unl.pt > ajuda↵
ajuda - Mostra os comandos existentes↵
sai - Termina a sessao do utilizador no programa↵
nova - Regista uma nova deslocacao↵
lista - Lista todas ou algumas deslocacoes registadas↵
boleia - Regista uma boleia para uma dada deslocacao↵
consulta - Lista a informacao de uma dada deslocacao↵
remove - Retira uma dada deslocacao↵
fb@fct.unl.pt >
```

3.2 Comando `termina`

Termina a execução do programa. O comando não necessita de argumentos. Este comando só tem sucesso se o programa estiver em modo inicial.

No caso do programa estar em modo inicial, o formato do comando é o seguinte:

```
> termina↵
Obrigado. Ate a proxima.↵
```

Um exemplo da execução do comando `termina`, quando o programa está em modo sessão com o utilizador fb@fct.unl.pt.

```
fb@fct.unl.pt > termina↵
Comando inexistente.↵
fb@fct.unl.pt >
```

3.3 Comando `regista`

Regista um novo utilizador no programa. Este comando só tem sucesso se o programa estiver em modo inicial, e não exisitir um utilizador com o email dado. Para realizar um registo deve ser dado sempre o email. Este email tem que ser único, e nesse caso é pedido o nome e a password do utilizador.

No caso do programa estar em modo inicial e o email ser único, o formato do comando é o seguinte (parâmetros com estilo **`bold`**):

```
> regista email utilizador↵
nome (maximo 50 caracteres): nome completo utilizador↵
password (entre 3 e 5 caracteres - digitos e letras): password utilizador↵
Registo efetuado.↵
>
```

Assuma que o nome não ultrapassa os 50 caracteres e é sempre válido. No entanto, a password deve ser verificada. Caso não seja válida, o programa deve notificar com a mensagem "Password incorrecta.", e voltar a pedir a mesma um máximo de 3 vezes. Por exemplo:

```
> regista fb@fct.unl.pt↵
nome (maximo 50 caracteres): Fernanda Maria Barbosa↵
password (entre 3 e 5 caracteres - digitos e letras): ip↵
Password incorrecta.↵
password (entre 3 e 5 caracteres - digitos e letras): ip2018↵
Password incorrecta.↵
password (entre 3 e 5 caracteres - digitos e letras): ip18↵
Registo efetuado.↵
>
```

Caso não seja dada uma password válida nas 3 tentativas o registo não será efetuado.

```
> regista fb@fct.unl.pt↵
nome (maximo 50 caracteres): Fernanda Maria Barbosa↵
password (entre 3 e 5 caracteres - digitos e letras): ip↵
Password incorrecta.↵
password (entre 3 e 5 caracteres - digitos e letras): ip2018↵
Password incorrecta.↵
password (entre 3 e 5 caracteres - digitos e letras): ip1834↵
Password incorrecta.↵
Registo nao efetuado.↵
>
```

Um exemplo da execução do comando `regista`, em que o email do utilizador **fb@fct.unl.pt** já existe. Neste caso, escreve-se na consola "Utilizador ja existente.".

```
> regista fb@fct.unl.pt↵
Utilizador ja existente.↵
Registo nao efetuado.↵
>
```

Um exemplo da execução do comando `regista`, quando o programa está em modo sessão com o utilizador **fb@fct.unl.pt**.

```
fb@fct.unl.pt > regista email utilizador↵
Comando inexistente.↵
fb@fct.unl.pt >
```

3.4 Comando `entrada`

Permite a entrada ("login") dum utilizador no programa. Este comando só tem sucesso se o programa estiver em modo inicial, e exisitir um utilizador com o email dado. Para realizar

a entrada na aplicação deve ser dado sempre o email. Este email tem que ser de um utilizador já registado. Neste caso é pedido a password, tendo o utilizador 3 tentativas para a editar correctamente.

No caso do programa estar em modo inicial e o email ser dum utilizador registado, o formato do comando é o seguinte:

```
> entrada email utilizador↵
password: password utilizador↵
emailUtilizador >
```

Note que, quando o comando `entrada` tem sucesso, o prompt seguinte tem o email do utilizador que fez login. Caso a password não seja correcta, o programa deve notificar com a mensagem "Password incorrecta.", e voltar a pedir (no máximo mais 2 vezes) até que seja dado um valor correcto. Por exemplo:

```
> entrada fb@fct.unl.pt↵
password: ip↵
Password incorrecta.↵
password: ip2018↵
Password incorrecta.↵
password: ip18↵
fb@fct.unl.pt >
```

Caso não seja dada a password correcta, a entrada não é efetuada.

```
> entrada fb@fct.unl.pt↵
password: ip↵
Password incorrecta.↵
password: ip2018↵
Password incorrecta.↵
password: ip1834↵
Password incorrecta.↵
>
```

Um exemplo da execução do comando `entrada`, em que o email do utilizador **fb@fct.unl.pt** não existe. Neste caso, escreve-se na consola "Utilizador nao existente.".

```
> entrada fb@fct.unl.pt↵
Utilizador nao existente.↵
>
```

Um exemplo da execução do comando `entrada`, quando o programa está em modo sessão com o utilizador **fb@fct.unl.pt**.

```
fb@fct.unl.pt > entrada email utilizador↵
Comando inexistente.↵
fb@fct.unl.pt >
```

3.5 Comando `sai`

Termina a sessao do utilizador no programa. O comando não necessita de argumentos. Este comando só tem sucesso se o programa estiver em modo sessão.

Um exemplo da execução do comando `sai`, quando o programa está em modo sessão com o utilizador **fb@fct.unl.pt**.

```
fb@fct.unl.pt > sai↵
Obrigado. Ate a proxima.↵
>
```

Um exemplo da execução do comando `sai`, quando o programa está em modo inicial.

```
> sai↵
Comando inexistente.↵
>
```

3.6 Comando **nova**

Regista uma nova deslocacao. Pode-se registar uma `nova` deslocação sempre e quando o programa esteja em modo sessão. Esta deslocação é registada pelo último utilizador que entrou no programa com o comando `entrada`.

Quando se regista a deslocação deve ser dada a seguinte informação: locais de origem e destino, data e horário da viagem (dia, mês, ano, hora estimada de partida e duração em horas estimada de viagem) e número de lugares do carro disponíveis para boleia. Note que a hora estimada é um valor inteiro entre 0 e 23, enquanto que a duração é um valor real positivo. Este registo tem sucesso se não existir uma deslocação deste utilizador com a mesma data, isto é no sistema o utilizador só pode ter uma deslocação por dia.

Se o utilizador com email *fb@fct.unl.pt* (utilizador com nome *Fernanda Barbosa*, dado no registo de utilizador) estiver a registar uma deslocação, o formato geral do comando é o seguinte:

```
fb@fct.unl.pt > nova↵
local origem↵
local destino↵
dia-mês-ano hora duração número lugares para boleia↵
Deslocacao registada. Obrigado Fernanda Barbosa.↵
fb@fct.unl.pt >
```

Exemplo de um registo de deslocação poderia ser:

```
fb@fct.unl.pt > nova↵
Lisboa↵
Alvalade do Sado↵
20-12-2018 15 1.45 5↵
Deslocacao registada. Obrigado Fernanda Barbosa.↵
fb@fct.unl.pt >
```

Por exemplo, suponha agora que o mesmo utilizador pretende registar uma deslocação entre Porto e Monte da Caparica no dia 20-12-2018 pelas 15 horas com uma duração estimada de 3.75 horas, e 5 lugares disponíveis. Neste caso, o registo não é efetuado pois já existe uma deslocação nessa data para este utilizador.

```
fb@fct.unl.pt > nova↵
Porto↵
Monte da Caparica↵
20-12-2018 15 3.75 5↵
Fernanda Barbosa ja tem uma deslocacao registada nesta data.↵
Deslocacao nao registada.↵
fb@fct.unl.pt >
```

Caso exista erros nos dados fornecidos pelo utilizador, o registo não é efetuado e escreve a mensagem de erro "Dados invalidos". Os erros possíveis são: data invalida, hora invalida, duração e lugares disponíveis com valores não positivos. Um exemplo desta situação é:

```
fb@fct.unl.pt > nova↵
Porto↵
Monte da Caparica↵
20-13-2018 15 3.75 5↵
Dados invalidos.↵
Deslocacao nao registada.↵
fb@fct.unl.pt >
```

Um exemplo da execução do comando `nova`, quando o programa está em modo inicial.

```
> nova↵
Comando inexistente.↵
>
```

3.7 Comando `lista`

Lista todas ou algumas deslocacoes registadas. Este comando só terá sucesso se o programa esteja em modo sessão. Caso não seja dada nenhuma informação, deve-se listar a informação referente às deslocações do último utilizador que entrou no programa com o comando `entrada`. Caso seja dado uma data, deve-se listar a informação referente às deslocações de todos os utilizadores na data indicada. Esta listagens estão ordenadas por data, e em caso de empate na data pela ordem alfabética crescente do email do utilizador que as registou.

O utilizador com email *fb@fct.unl.pt* (utilizador com nome *Fernanda Barbosa*, dado no registo de utilizador) executa o comando `lista`. Caso não tenha registado nenhuma deslocação:

```
fb@fct.unl.pt > lista↵
Fernanda Barbosa nao tem deslocacoes registadas.↵
fb@fct.unl.pt >
```

Caso tenha registado 2 deslocações:

```
fb@fct.unl.pt > lista↵
Lisboa↵
Alvalade do Sado↵
20-12-2018 15 1.45 5↵
Boleias registadas: 0↵
↵
Porto↵
Monte da Caparica↵
21-12-2018 10 3.75 5↵
Boleias registadas: 2↵
↵
fb@fct.unl.pt >
```

O utilizador com email *fb@fct.unl.pt* (utilizador com nome *Fernanda Barbosa*, dado no registo de utilizador) executa o comando `lista`, dando uma data. Caso não existam deslocações para a data indicada:

```
fb@fct.unl.pt > lista 13-4-2018↵
Fernanda Barbosa nao existem deslocacoes registadas para 13-4-2018.↵
fb@fct.unl.pt >
```

Caso exista 4 deslocações registadas nessa data. Os utilizadores que registaram tem os seguintes emails *ar@fct.unl.pt*, *bt@fct.unl.pt*, *fb@fct.unl.pt* e *mtmp@fct.unl.pt*:

```
fb@fct.unl.pt > lista 20-12-2018↵
ar@fct.unl.pt↵
Lisboa↵
Costa da Caparica↵
20-12-2018 15 0.25 5↵
Boleias registadas: 0↵
↵
bt@fct.unl.pt↵
Porto↵
Monte da Caparica↵
20-12-2018 10 3.75 5↵
Boleias registadas: 2↵
↵
fb@fct.unl.pt↵
Porto↵
Monte da Caparica↵
20-12-2018 10 3.75 5↵
Boleias registadas: 1↵
↵
mtmp@fct.unl.pt↵
Braga↵
Monte da Caparica↵
20-12-2018 10 3.75 5↵
Boleias registadas: 3↵
↵
fb@fct.unl.pt >
```

É de notar que esta listagem é ordenada pelo email do utilizador que registou as deslocações.

Caso a data seja inválida, esse facto é dito ao utilizador da seguinte forma:

```
fb@fct.unl.pt > lista 0-12-2018↵
Data invalida.↵
fb@fct.unl.pt >
```

Exemplos da execução do comando `lista`, quando o programa está em modo inicial.

```
> lista↵
Comando inexistente.↵
>
> lista 13-4-2017↵
Comando inexistente.↵
>
```

3.8 Comando `consulta`

Lista a informacao de uma dada deslocação. Lista a informação de uma `dada` deslocação sempre e quando o programa esteja em modo sessão, e essa deslocação exista.

Por exemplo, se o utilizador com email *fb@fct.unl.pt* (utilizador com nome *Fernanda Barbosa*, dado no registo de utilizador) executar o comando `consulta`, indicando o email do utilizador que registou a deslocação, e a data da deslocação, é escrito na consola a informação da deslocação e o número de lugares disponíveis para boleia.

```
fb@fct.unl.pt > consulta mtmp@fct.unl.pt 20-12-2018↵
Braga↵
Monte da Caparica↵
20-12-2018 10 3.75 5↵
Lugares vagos: 2↵
fb@fct.unl.pt >
```

Caso não exista a deslocação que está a ser consultada, esse facto é dito ao utilizador da seguinte forma:

```
fb@fct.unl.pt > consulta mtmp@fct.unl.pt 21-12-2018↵
Deslocacao nao existe.↵
fb@fct.unl.pt >
```

Caso não exista o utilizador dado, esse facto é dito ao utilizador da seguinte forma:

```
fb@fct.unl.pt > consulta naoexiste@gmail.com 21-12-2018↵
Utilizador inexistente.↵
fb@fct.unl.pt >
```

Caso a data seja inválida, esse facto é dito ao utilizador da seguinte forma:

```
fb@fct.unl.pt > consulta mtmp@fct.unl.pt 0-12-2018↵
Data invalida.↵
fb@fct.unl.pt >
```

Exemplo da execução do comando `consulta`, quando o programa está em modo inicial.

```
> consulta mtmp@fct.unl.pt 21-12-2018↵
Comando inexistente.↵
>
```

3.9 Comando **boleia**

Regista uma boleia para uma dada deslocacao. Regista mais um lugar ocupado para uma `dada` deslocação sempre e quando o programa esteja em modo sessão, e essa deslocação exista, tenha lugares vagos e não tenha sido registada pelo último utilizador a executar o comando `entrada`.

Se o utilizador com email *fb@fct.unl.pt* (utilizador com nome *Fernanda Barbosa*, dado no registo de utilizador) executar o comando `boleia`, indicando o email do utilizador que registou a deslocação, e a data da deslocação, é registada a boleia, caso exista lugares disponíveis para a deslocação. Vejamos um exemplo desta situação:

```
fb@fct.unl.pt > consulta mtmp@fct.unl.pt 20-12-2018↵
Braga↵
Monte da Caparica↵
20-12-2018 10 3.75 5↵
Lugares vagos: 2↵
fb@fct.unl.pt > boleia mtmp@fct.unl.pt 20-12-2018↵
Boleia registada.↵
fb@fct.unl.pt > consulta mtmp@fct.unl.pt 20-12-2018↵
Braga↵
Monte da Caparica↵
20-12-2018 10 3.75 5↵
Lugares vagos: 1↵
fb@fct.unl.pt >
```

Caso exista a deslocação mas não tenha lugares vagos, esse facto é dito ao utilizador da seguinte forma:

```
fb@fct.unl.pt > consulta mtmp@fct.unl.pt 20-12-2018↵
Braga↵
Monte da Caparica↵
20-12-2018 10 3.75 5↵
Lugares vagos: 0↵
fb@fct.unl.pt > boleia mtmp@fct.unl.pt 20-12-2018↵
Fernanda Barbosa nao existe lugar. Boleia nao registada.↵
fb@fct.unl.pt > consulta mtmp@fct.unl.pt 20-12-2018↵
Braga↵
Monte da Caparica↵
20-12-2018 10 3.75 5↵
Lugares vagos: 0↵
fb@fct.unl.pt >
```

Caso exista a deslocação mas tenha sido registada pelo próprio utilizador, esse facto é dito ao utilizador da seguinte forma:

```
fb@fct.unl.pt > consulta fb@fct.unl.pt 20-12-2018↵
Braga↵
Monte da Caparica↵
20-12-2018 10 3.75 5↵
Lugares vagos: 2↵
fb@fct.unl.pt > boleia fb@fct.unl.pt 20-12-2018↵
Fernanda Barbosa nao pode dar boleia a si propria. Boleia nao registada.↵
fb@fct.unl.pt > consulta fb@fct.unl.pt 20-12-2018↵
Braga↵
Monte da Caparica↵
20-12-2018 10 3.75 5↵
Lugares vagos: 2↵
fb@fct.unl.pt >
```

Caso não exista a deslocação, esse facto é dito ao utilizador da seguinte forma:

```
fb@fct.unl.pt > boleia mtmp@fct.unl.pt 21-12-2017↵
Deslocacao nao existe.↵
fb@fct.unl.pt >
```

Caso não exista o utilizador dado, esse facto é dito ao utilizador da seguinte forma:

```
fb@fct.unl.pt > boleia naoexiste@gmail.com 21-12-2018↵
Utilizador inexistente.↵
fb@fct.unl.pt >
```

Caso a data seja inválida, esse facto é dito ao utilizador da seguinte forma:

```
fb@fct.unl.pt > boleia mtmp@fct.unl.pt 0-12-2018↵
Data invalida.↵
fb@fct.unl.pt >
```

Exemplo da execução do comando `consulta`, quando o programa está em modo inicial.

```
> boleia mtmp@fct.unl.pt 21-12-2018↵
Comando inexistente.↵
>
```

3.10 Comando **remove**

Retira uma dada deslocacao. Pode-se eliminar uma `dada` deslocação sempre e quando o programa esteja em modo sessão, essa deslocação tenha sido registada pelo último utilizador que entrou no programa com o comando `entrada`, e não exista boleias registadas.

Quando se remove a deslocação deve ser dada a data da deslocação. Caso o utilizador tenha registado uma deslocação nesta data e ainda ninguém se tenha registado para boleia, essa deslocação será eliminada. Por exemplo, se o utilizador com email *fb@fct.unl.pt* (utilizador com nome *Fernanda Barbosa*, dado no registo de utilizador) tiver as seguintes deslocações:

```
fb@fct.unl.pt > lista↵
Lisboa↵
Alvalade do Sado↵
20-12-2018 15 1.45 5↵
Boleias registadas: 0↵
↵
Porto↵
Monte da Caparica↵
21-12-2018 10 3.75 5↵
Boleias registadas: 2↵
fb@fct.unl.pt >
```

Caso o utilizador trate de remover a deslocação do dia 20-12-2018, a deslocação é eliminada com sucesso:

```
fb@fct.unl.pt > remove 20-12-2018↵
Deslocacao removida.↵
fb@fct.unl.pt >
```

Caso o utilizador trate de remover a deslocação do dia 21-12-2018, a deslocação não é eliminada porque já tem boleias registadas:

```
fb@fct.unl.pt > remove 21-12-2018↵
Fernanda Barbosa ja nao pode eliminar esta deslocacao.↵
fb@fct.unl.pt >
```

Caso o utilizador trate de remover uma deslocação do dia 22-12-2018, a deslocação não é eliminada porque não existe:

```
fb@fct.unl.pt > remove 22-12-2018↵
Deslocacao nao existe.↵
fb@fct.unl.pt >
```

Caso a data seja inválida, esse facto é dito ao utilizador da seguinte forma:

```
fb@fct.unl.pt > remove 0-12-2018↵
Data invalida.↵
fb@fct.unl.pt >
```

Um exemplo da execução do comando `remove`, quando o programa está em modo inicial.

```
> remove 23-12-2018↵
Comando inexistente.↵
>
```

4 A Entrega

O seu programa deve ser entregue no concurso do Mooshak para o efeito (Concurso IP201819_TP2), até ao dia 7 de Dezembro de 2018 pelas 17:00 (hora de Lisboa).

Cada grupo de 2 alunos (equipa de trabalho) deve registar-se neste concurso. O nome do utilizador deve ser o seu **número de aluno_número de aluno** (o primeiro número a colocar é o menor) no grupo correspondente ao seu turno (por exemplo, os alunos nº 3435 e nº 3434 do turno p1 deve ter como nome utilizador no mooshak **3434_3435** no grupo **TP1**). Só serão aceites como entregas para avaliação, os programas cujo utilizador no concurso do Mooshak siga estas regras.

Para o trabalho ser avaliado, os alunos devem ter cumprido o código de ética do DI (ver na página do moodle), e ter, no final do prazo, uma entrega no concurso do mooshak. Pode submeter o seu código mais que uma vez. Será a última submissão que será avaliada.

A avaliação do seu trabalho tem 2 componentes, sendo a nota final calculada como a soma das notas dessas componentes:

- Avaliação da correcção dos resultados produzidos: até 12 valores
  - Submissão ao concurso obtendo a pontuação máxima (100 pontos), com pelo menos 2 classes (Main, Partilha) bem definidas terá 12 valores nesta componente. Note que muito provavelmente vai necessitar de mais do que duas classes para fazer este projecto **bem**.
  - Uma submissão com erros em algumas das funcionalidades terá uma classificação correspondente às funcionalidades correctamente implementadas.
- Avaliação da qualidade do código: 8 valores.

Por exemplo, um aluno que entregue uma aplicação que receba 90 pontos terá 90% dos 12 pontos, a que se somam entre 0 e 8 pontos pela qualidade do código.