CloudSharing

Programação Orientada pelos Objectos

Enunciado do 1º e 2º Trabalho Prático, versão 1.6 – 2013-03-28

Contacto: carla.ferreira@fct.unl.pt

Notas importantes:

Este enunciado diz respeito aos 2 primeiros trabalhos de POO1213, que constituem as Fases 1 e 2 respectivamente do mesmo problema. Fornece directivas para ambas as fases, estando prevista uma versão 2.0 do enunciado, para refinar detalhes sobre a Fase 2.

Prazo de entrega – Fase 1 (1º trabalho prático): até às 17h00 do dia 2 de Abril de 2013.

Prazo de entrega – Fase 2 (2º trabalho prático): até às 17h00 do dia 15 de Abril de 2013.

Realização em grupos: grupos de 2 alunos inscritos no mesmo turno prático (ou do mesmo docente se este autorizar).

Material a entregar – Fase 1

- **Moodle**: submissão via Moodle do ficheiro zip contendo o **código fonte**, devidamente comentado, de todas as interfaces desenvolvidas, o respectivo **Javadoc** e o **diagrama de interfaces**. Por favor inclua o seu nome e número de aluno em todos os seus ficheiros fonte (*.java), em cláusulas `@author`. Compacte todos estes artefactos num ficheiro cujo nome tem de ser da forma " POO1213_PZ_XXXXX_YYYYY", com XXXXX < YYYYY, onde XXXXX e YYYYY representam os números dos membros do grupo e Z indica o turno prático (exemplo: POO1213_P1_12345_54321.zip).

Material a entregar – Fase 2

- **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**. O nome do ficheiro compactado tem as mesmas regras que a fase anterior.
- **Mooshak**: submissão e aceitação do código fonte pelo sistema de avaliação automática Mooshak. Na versão 2.0 do enunciado serão dados mais detalhes sobre esta avaliação. Consulte na página da disciplina as regras de submissão de trabalhos ao Mooshak.

Recomendações: A boa documentação do seu código fonte (em qualquer das fases) 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 CloudSharing

1.1. Descrição do problema

O objectivo deste trabalho é o desenvolvimento de uma aplicação que permita a gestão de um sistema para armazenar ficheiros na cloud (e.g. Dropbox). Cada utilizador registado tem acesso a uma conta onde pode adicionar ficheiros, sendo que esta conta tem uma capacidade máxima que não pode ser excedida. Os ficheiros armazenados no sistema podem ser consultados através de um website. Para além disso, também é possível partilhar ficheiros com outros utilizadores registados. Para cada ficheiro também é possível consultar o autor (nome da conta) da última actualização.

Cada conta (ou utilizador) tem um email (identificador único da conta) e os ficheiros dos quais é proprietário. Existem dois tipos de contas, as contas básicas gratuitas ("basic") e as contas premium com anuidade ("premium"). As contas premium têm 5 GB de espaço e podem partilhar os seus ficheiros com outros utilizadores registados. As contas básicas têm 2GB de espaço e os seus ficheiros não podem ser partilhados. No entanto podem modificar ficheiros partilhados por outros utilizadores com contas premium. Outra diferença entre as contas básicas e premium, é a contabilização do espaço usado. Nas contas premium o espaço ocupado pelos ficheiros partilhados não é contabilizado, enquanto nas contas básicas o espaço ocupado pelos ficheiros partilhados é contabilizado a 50%. Por exemplo, para uma conta básica com 100MB em ficheiros proprietários e 180MB em ficheiros partilhados, considera-se que esta conta usa 190MB dos 2GB disponíveis.

Cada ficheiro tem um nome, uma dimensão (em MB), uma conta (ou utilizador) proprietário e o nome da conta que fez a última actualização. Um ficheiro é identificado univocamente pelo seu nome e pela conta a que pertence.

A aplicação deve então permitir:

1. **Criar uma conta** (comando `ADD`)**.** São fornecidos o nome da conta (o email do utilizador) e o tipo de conta (básica ou premium). A operação falha se: (1) já existir uma conta com esse email.
2. **Adicionar um ficheiro** (comando `UPLOAD`)**.** São fornecidos o nome da conta, o nome do ficheiro e a dimensão do ficheiro em MB. A operação falha se: (1) a conta não existir; (2) já existir um ficheiro com esse nome nessa conta; ou (3) a dimensão do ficheiro exceder a capacidade da conta.
3. **Partilhar um ficheiro** (comando `SHARE`)**.** São fornecidos o nome da conta a que o ficheiro pertence, o nome da conta a adicionar à partilha e o nome do ficheiro. A operação falha se: (1) uma das contas não existir; (2) o ficheiro não existir; (3) a conta não permitir partilhar ficheiros; (4) a partilha já existir; ou (5) a dimensão do ficheiro exceder a capacidade da conta.
4. **Actualizar ficheiro** (comando `UPDATE`). São fornecidos o nome da conta proprietária, o nome da conta que vai efectuar actualização, e o nome do ficheiro a actualizar. Assume-se que a actualização não altera a dimensão do ficheiro. Note que se for a conta proprietária do ficheiro a fazer a actualização, os dois nomes de conta são iguais. A operação falha se: (1) uma das contas não existir; (2) a primeira conta não for proprietária do ficheiro; ou (3) o ficheiro não pertencer ou não for partilhado com a segunda conta.
5. **Apresentar a conta que tem menos espaço livre** (comando `MINSPACE`)**.** Se existirem várias contas que satisfazem este critério, devolve a conta mais antiga. A operação falha se: (1) não existirem contas no sistema.
6. **Listar todos os ficheiros proprietários e partilhados de uma conta** (comando `LIST FILES`). É fornecido o nome da conta. A operação falha se: (1) a conta não existir.

7. **Listar todas as contas** (comando `LIST ALL`)**.** A operação tem sempre sucesso.
8. **Listar todas as contas de um dado tipo** (comando `LIST X`, onde `X` é `PREMIUM` ou `BASIC`)**.** A operação tem sempre sucesso.
9. **Listar a conta (utilizador) que efectuou a última actualização sobre um dado ficheiro** (comando `LASTUPDATE`)**.** São fornecidos o nome conta que é proprietária do ficheiro e o nome do ficheiro. A operação falha se: (1) a conta não existir; ou (2) o ficheiro não existir. Se ainda não foi feita nenhuma actualização, o método deve devolver a conta proprietária do ficheiro.
10. **Sair da aplicação** (comando `EXIT`)**.** A operação tem sempre sucesso.

1.2. Exemplo de interacção com a aplicação

A aplicação desenvolvida (Fase 2) tem que garantir modelo de interacção ilustrado no exemplo seguinte (o caracter ↵ representa uma mudança de linha):

```
MINSPACE↵
> No accounts.↵
↵
ADD cf@gmail.com premium↵
> Account was added.↵
↵
ADD cf@gmail.com basic↵
> Account already exists.↵
↵
UPLOAD cf@gmail.com poo.txt 2↵
> File uploaded into account.↵
↵
UPLOAD poo@gmail.com poo.txt 2↵
> Account does not exist.↵
↵
UPLOAD cf@gmail.com poo.txt 2↵
> File already exists in the account.↵
↵
UPLOAD cf@gmail.com ada.pdf 7000↵
> File size exceeds account capacity.↵
↵
UPLOAD cf@gmail.com contactBook.pdf 3000↵
> File uploaded into account.↵
↵
ADD dp@fct.unl.pt basic↵
> Account was added.↵
↵
ADD bf@fct.unl.pt premium↵
> Account was added.↵
↵
SHARE cf@gmail.com bf@fct.unl.pt poo.txt↵
> File was shared.↵
↵
SHARE bf@fct.unl.pt dp@fct.unl.pt poo.txt↵
> File does not exist.↵
↵
UPLOAD dp@fct.unl.pt zoo.zip 1000↵
> File uploaded into account.↵
↵
LASTUPDATE dp@fct.unl.pt zoo.zip↵
> Last update: dp@fct.unl.pt↵
↵
```

```
SHARE cf@gmail.com dp@fct.unl.pt poo.txt↵
> File was shared.↵
↵
SHARE mg@gmail.com sc@fct.unl.pt ada.pdf↵
> Account does not exist.↵
↵
SHARE cf@gmail.com dp@fct.unl.pt ada.pdf↵
> File does not exist.↵
↵
SHARE dp@fct.unl.pt cf@gmail.com zoo.zip↵
> Account does not allow file sharing.↵
↵
SHARE cf@gmail.com dp@fct.unl.pt poo.txt↵
> File already shared.↵
↵
SHARE cf@gmail.com dp@fct.unl.pt contactBook.pdf↵
> File size exceeds account capacity.↵
↵
UPDATE cf@gmail.com dp@fct.unl.pt poo.txt↵
> File was updated.↵
↵
UPDATE sc@gmail.com dp@fct.unl.pt teste1.doc↵
> Account does not exist.↵
↵
UPDATE cf@gmail.com dp@fct.unl.pt zoo.zip↵
>File does not exist.↵
↵
UPDATE dp@fct.unl.pt cf@gmail.com zoo.zip↵
> File not shared.↵
↵
MINSPACE↵
> Account with least free space: cf@gmail.com↵
↵
LIST FILES cf@gmail.com↵
> Account has 2 files:↵
> poo.txt (2 MB)↵
> contactBook.pdf (3000 MB)↵
> Account has 0 shared files:↵
↵
LIST ALL↵
> All accounts:
> cf@gmail.com (Premium)↵
> dp@fct.unl.pt (Basic)↵
↵
LIST PREMIUM↵
> Premium accounts:
> cf@gmail.com↵
↵
LIST BASIC↵
> Basic accounts:
> dp@fct.unl.pt ↵
↵
LASTUPDATE cf@gmail.com poo.txt↵
> Last update: dp@fct.unl.pt↵
↵
EXIT↵
> Exiting...↵
↵
```

2. Desenvolvimento

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

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. Depois, identifique as entidades de que vai necessitar para implementar este sistema. Identifique e especifique cuidadosamente as **interfaces** (Fase 1 e 2) e **classes** (Fase 2) 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.