Projeto Final INE 5402 - POO I
Quenio C. M. Santos
7 de julho de 2014

1 - Introdução ao Projeto CheckSpace

A seguir, encontram-se o propósito do projeto CheckSpace e os princípios que guiaram sua implementação.

1.1 - Propósito

Um aplicativo que auxilia o usuário na liberação de espaço em seu computador, informando quais são as pastas que podem ser removidas e/ou arquivadas sem prejudicar seu fluxo de trabalho.

1.2 - Princípios

1. *Segurança*: O aplicativo somente fornecerá informações sobre as pastas do usuário e não irá remover (ou arquivar) os arquivos do usuário. Para executar tais tarefas, o usuário deverá utilizar o sistema operacional de seu computador.
2. *Praticidade*: O aplicativo será fácil de executar e as informações serão apresentadas de forma clara. O usuário precisa saber apenas um comando para executar o aplicativo e este guiará o usuário durante o uso com mensagens explicativas.
3. *Eficiência*: O aplicativo deverá executar de forma rápida. Se alguma operação não for instantânea, o usuário deverá ser informado dos passos sendo executados e do progresso da execução.

2 - Cenário de Uso do

Aplicativo

Os computadores pessoais possuem grande capacidade de armazenamento, porém a demanda dos usuários é cada vez maior, agora que se pode armazenar de forma digital vários tipos de mídia, tais como fotos, vídeos, música, cópias de documentos, etc. Assim sendo, é comum que o usuário do computador se surpreenda com uma mensagem do sistema operacional lhe informando que não há mais espaço disponível para armazenamento de arquivos.
Nesta situação, o usuário pode não saber quais arquivos e pastas podem ser removidos e/ou arquivados (por falta de uso) sem que isto lhe cause eventual transtorno quando os arquivos possam ser necessários.

2.1 - Solução

Uma solução possível será instalar um pequeno aplicativo para auxiliar na tarefa de remoção e/ou arquivamento dos arquivos e pastas menos usados.

Este aplicativo deverá utilizará pouco espaço em disco; uma vez que o computador já está deficiente de espaço.

2.2 - Execução

Uma vez instalado, o aplicativo é executado na linha-de-comando digitando-se o comando `check-space`.

O aplicativo então pergunta ao usuário o nome de uma pasta a ser analisada e oferece como sugestão a pasta `home` (ou “pasta do usuário”).

Neste momento, o usuário pode digitar o caminho de uma pasta específica (pasta da qual o usuário suspeita que haja arquivos ou sub-pastas com pouco uso), ou pode apenas digitar `ENTER` para que a sugestão de pasta seja aceita.

2.3 - Análise da Pasta

Uma vez que a pasta foi definida pelo usuário, o aplicativo faz uma análise da pasta, suas sub-pastas e arquivos, capturando as seguintes informações:

- quanto espaço está sendo utilizado pelas sub-pastas e arquivos dentro da pasta-raíz;
- quando foi o último acesso a estas sub-pastas e arquivos.

2.4 - Relatório de Uso

As informações capturadas sobre a pasta definida pelo usuário são apresentadas ao usuário em uma tabela listando os arquivos e sub-pastas da pasta-raíz, contendo as seguintes colunas:

- *Nome*: O nome do arquivo ou sub-pasta.
- *Espaço*: O espaço ocupado pelo arquivo ou sub-pasta.
- *Último Accesso*: a data e hora em que o arquivo ou sub-pasta foi acessado pela última vez.

2.5 - Ordem da Lista

Primeiramente, os itens do relatório são listados em ordem crescente do *Nome*, porém o usuário pode mudar a ordem para decrescente — digitando `od` — e mudar a coluna usada na ordenação — digitando `c2` para a coluna *Espaço* e `c3` para a coluna *Último Acesso*. Pode também retornar à ordem crescente digitando `oc` e à coluna *Nome* digitando `c1`.

Imediatamente após um dos comandos descritos acima ser executado pelo usuário, o *Relatório de Uso* é reimpresso na tela com a devida ordem.

2.6 - Mudando a Pasta Analisada

A qualquer momento durante a execução do aplicativo, o usuário pode mudar a pasta sendo analisada digitando `p`.

Neste caso, o aplicativo vai novamente perguntar ao usuário o caminho da pasta a ser analisada e o usuário deve proceder como foi descrito na seção *Execução*.

2.7 - Pasta Inexistente

Se o usuário digitar um caminho que não leva a qualquer pasta do computador, então uma mensagem informa o usuário que
`A pasta digitada não se encontra neste computador.`

Logo após, o aplicativo novamente pergunta ao usuário o caminho de uma pasta e a execução procede como descrito na seção *Execução*.

2.8 - Informando o Progresso da Análise

A pasta sendo analisada pode conter centenas ou até milhares de arquivos, o que pode tornar demorada a captura de informações do *Relatório de Uso*.

Quando isto ocorrer, o usuário será informado na linha-de-comando qual arquivo ou sub-pasta está sendo analisada pelo aplicativo naquele momento.

2.9 - Comandos Disponíveis

A qualquer momento durante a execução do aplicativo, todos os comandos disponíveis ao usuário serão apresentados na tela com uma frase explicativa.

Se o usuário digitar um comando desconhecido pelo aplicativo, o usuário será informado que
`O comando digitado não foi reconhecido. Por favor, tente novamente.`
e assim o usuário poderá repetir o comando cuidando com a sintaxe.

2.10 - Saindo do Aplicativo

Quando o usuário estiver satisfeito com as informações fornecidas pelo aplicativo sobre o uso de suas pastas, esse pode encerrar a execução digitando o comando `s`.

3 - Componentes do Aplicativo

O aplicativo será dividido em componentes, sendo cada um responsável por um aspecto do aplicativo. Cada seção a seguir descreve um componente do aplicativo.

3.1 - Console

Este componente é responsável por imprimir e ler linhas no terminal do computador.

Sua classe principal é chamada *Console* e é utilizada pelos demais componentes sempre que é necessário apresentar informações ao usuário ou requisitar comandos deste.

3.2 - Comando

O componente de comando será responsável pela interação com o usuário e com os demais componentes do aplicativo. Esse receberá os comandos do usuário e executará a função correspondente.

A abstração central deste componente é a interface *Command*, que representa um comando que pode ser executado pelo usuário. Existem algumas implementações:

- *RootFolderProcessingCommand*: reconhece comando `p` como descrito na seção anterior, permitindo ao usuário definir uma pasta, processando esta e finalmente imprimindo o *Relatório de Uso*.
- *ReportOrderDirectionCommand*: reconhece os comandos `od` e `oc` ordenando o *Relatório de Uso* em ordem descrente ou crescente, respectivamente.
- *ReportOrderColumnCommand*: reconhece os comandos iniciados com `c` e ordena a coluna correspondente do *Relatório de Uso*.
- *ExitCommand*: reconhece o comando `s` e termina a execução do aplicativo.

A classe principal de controle é *CommandLine*, que processa os comandos executados pelo usuário na linha-de-comando. Para cada linha digitada pelo usuário, esta classe:

- Passa a linha para cada um dos comandos disponíveis até achar um comando que reconheça a linha digitada.
- Executa o comando que reconheceu a linha.
  CommandLine também é responsável por imprimir uma linha explicativa para cada um dos comandos disponíveis.

Todas as instâncias de *Command* e a instância de *CommandLine* vão interagir com o usuário através da classe *Console*.

3.3 - Relatório

Este componente implementa a impressão do *Relatório de Uso* descrito na seção *Cenário de Uso do Aplicativo*.

Sua classe principal é *UsageReport* que representa *Relatório de Uso* e sabe imprimi-lo através da classe *Console*.

As classes *ReportSorter* e *ReportOrder* definem a ordem de listagem do *Relatório de Uso* — crescente ou decrescente — e a coluna que define a ordem.

3.4 - Processamento

O componente de processamento se encarrega de introspectar a pasta-raíz e capturar toda a informação necessária sobre as sub-pastas e arquivos para que se possa imprimir o *Relatório de Uso*.

A classe principal deste componente é chamada *RootFolderProcessor*. Ela executa o processamento, gerando como resultado a seguinte estrutura de objetos da seguintes classes:

- *RootFolder*: representa a pasta-raíz na qual foi feita a introspecção.
- *RootFolderItem*: representa uma sub-pasta ou um arquivo dentro da pasta-raíz.
  O resultado da execução de *FolderProcessing* retorna uma instância de *RootFolder* que contém uma ou mais instâncias de *RootFolderItem*, cada uma representando um item encontrado na

pasta-raíz da instância de *RootFolder*.

A classe *RootFolderItem* contém os seguintes atributos (ou propriedades em Java):

- *Name*: o nome do item.
- *Space*: o espaço utilizado pelo item em *bytes*.
- *LastAccess*: um *timestamp* do último acesso efetuado a este item.

Se a instância de *RootFolderItem* está representando uma pasta dentro da pasta-raíz:

- o atributo *Space* é a soma do espaço ocupado por todos os arquivos encontrados nesta pasta, incluindo os arquivos em suas sub-pastas;
- e o atributo *LastAccess* é a *timestamp* do arquivo encontrado nesta pasta, ou em suas sub-pastas, que foi acessado mais recentemente.

Estas informações serão utilizadas por *UsageReport* para imprimir o *Relatório de Uso*.

4 - Fases do Projeto

A implementação se dará em duas fases e algumas tarefas serão executadas em cada fase como descrito nas seções a seguir.

Na primeira fase do projeto, os aspectos de interação com o usuário e impressão do *Relatório de Uso* devem estar implementados.

Somente a segunda fase do projeto se preocupará com o processamento, ou seja, a captura de informações de cada sub-pasta e arquivo para que o relatório de uso seja autêntico. É também nesta fase que se implementará a ordenação das linhas do *Relatório de Uso*.

Espera-se que o código-fonte ainda esteja em forma rudimentar na primeira fase do projeto, sendo refinado na segunda fase.

4.1 - Phase I

Abaixo estão listadas as tarefas da primeira fase do projeto:

1. Implementar a classe *Console*.
2. Implementar a classe *CommandLine* e a interface *Command*.
3. Implementar as classes *RootFolder* e *RootFolderItem*.
4. Implementar a classe *UsageReport* e classes auxiliares.
5. Implementar a classe principal do sistema.
6. Implementar *ExitCommand*.
7. Implementar *RootFolderProcessingCommand* e uma versão preliminar de *RootFolderProcessor*.

4.2 - Phase II

Abaixo estão listadas as tarefas da segunda fase:

1. Implementar *ReportOrderDirectionCommand* e *ReportOrderColumnCommand*.
2. Implementar o método `sort()` de *ReportSorter*.
3. Finalizar a implementação de *RootFolderProcessor*.
4. Alinhar as colunas da tabela de *UsageReport*.