Semana de 21 a 25 Fevereiro

Exercício

André Souto e Isabel Nunes

Unidade Curricular de
Laboratórios de Programação

2021/2022

Exercício

Objetivos

- Familiarização com o Eclipse
- Leitura, manipulação e escrita de conteúdo de ficheiros em Java usando as classes `Scanner` e `PrintWriter` respetivamente

Antes de Começar

De modo a poder realizar este exercício deverá recordar as últimas aulas de IP em que estas classes foram abordadas e familiarizar-se com os métodos disponíveis para as classes `Scanner` e `PrintWriter`. Pode ainda ler os textos `ApontamentosScanner` e `ApontamentosStringStringBuilder` acessíveis no Moodle, nos materiais desta semana.

No exemplo seguinte demonstra-se uma utilização básica de canais de leitura e escrita para ficheiros de texto, lendo inteiros de um ficheiro `myInput.txt` (um por linha) e escrevendo no segundo ficheiro `myOutput.txt` apenas os que são ímpares. Recorde que:

1. Não é possível prever em tempo de compilação a existência de um ficheiro com o nome `myInput.txt`. Por essa razão o método `main` pode lançar uma exceção em tempo de execução.

   Outra exceção que pode ser lançada deve-se ao facto de não haver garantias de permissão de escrita no ficheiro `resultado.txt`.

   Para já não se tratam as exceções, apenas se avisa que podem ocorrer, colocando na assinatura do método a expressão `throws IOException`.

2. Pode-se usar um `Scanner` que "trabalha" sobre um ficheiro que contém os dados a serem lidos.

   ```
   Scanner myReader = new Scanner(new File("myInput.txt"));
   ```

3. Existem muitas formas de escrever num ficheiro. No exemplo utiliza-se um `PrintWriter` como canal de escrita:

   ```
   PrintWriter myWriter = new PrintWriter("myOutput.txt");
   ```

4. É necessário no final fechar os canais de comunicação através do método `close` de cada uma das classes.

No exemplo seguinte são usados os métodos `boolean hasNextInt()` e `int nextInt()` da classe `Scanner`. Poderá relembrar na API desta classe outros métodos com as mesmas funcionalidades mas que trabalham com tipos de dados diferentes como, por exemplo, os métodos `boolean hasNextLine()` e `String nextLine()`, `boolean hasNext()` e `String next()`, entre outros.

```
import java.io.File;
import java.io.IOException;
import java.io.PrintWriter;
import java.util.Scanner;

/**
 * This class exemplifies reading from one text file and
 * writing to another
 */
public class FileReadWriteExample {

    /**
     * Open two files: one to read from, and another to write
     * into. The values of the first file are read, and the
     * ones that are odd are written into the second file
     */
    public static void main(String[] args) throws IOException {

        // open a reading channel
        Scanner myReader = new Scanner(new File("myInput.txt"));
        // open a writing channel
        PrintWriter myWriter = new PrintWriter("myOutput.txt");

        // while the input file still has values to read that
        // represent integers, read the value, and write it to
        // the output file if it is odd
        while (myReader.hasNextInt()){
            int value = myReader.nextInt();
            if(value % 2 != 0){
                myWriter.println(value);
            }
        }

        myReader.close();
        myWriter.close();
    }
}
```

ATENÇÃO

Se não tiver a instrução de fecho (`close`) sobre o `PrintWriter`, nada é escrito no ficheiro.

Agora que já relembraram as matérias necessárias, mãos à obra!

Mas antes de começar... algo muito importante!

Quando temos um ciclo infinito que escreve para um ficheiro, rapidamente o disco fica cheio e a seguir não conseguimos escrever mais nada. Isto aconteceu a alguns alunos no ano passado.

Para evitar esta situação:

- Enquanto estão a desenvolver o método, escrevam para o `System.out`;
- Se houver um ciclo infinito, rapidamente se aperceberão disso e poderão terminar a execução (basta clicar no quadrado encarnado na janela da consola);
- Quando tiverem a certeza que o método já está correto, alteram a escrita para ficheiro, usando então um objeto `PrintWriter`.

Finalmente, vamos a isto!

Primeiro vai criar um projeto java no Eclipse. Vamos chamar a este projeto `exercise0Project`.

De seguida vai copiar os ficheiros por nós fornecidos, reunidos numa pasta `studentsExercise0`, para este novo projeto. Para isso, clicar com o botão direito do rato em cima do `exercise0Project`, escolher **Import → General → File System** e clicar em "Next".

Os ficheiros que importou foram os seguintes:

- `RunExercise0.java`
- `TestsExercise0.java`
- `myNumbers.txt`
- `myText.txt`

No diálogo seguinte carregar no botão "Browse" da caixa "From directory" e escolher a pasta `studentsExercise0`. Clicar "Finish".

Agora "arrastar" os dois ficheiros `.java` para dentro da pasta `src` do projeto.

Sobre o ficheiro `TestsExercise0.java`, com o botão direito do rato selecionar **Build Path → Configure Build path**. Na aba *Libraries*, escolher *Classpath*, carregar no botão "AddLibrary" à direita, escolher *JUnit*, botão "Next" e de seguida botão "Finish". Finalmente, botão "Apply and close".

As classes `RunExercise0.java` e `TestsExercise0.java` ainda apresentam erros pois ainda não foi criada a classe `Exercise0`.

Finalmente, vai criar a classe `Exercise0`, onde vai programar os métodos pedidos de seguida:

- `public static void copyText (String fileIn, String fileOut) throws FileNotFoundException` que copia o conteúdo do ficheiro de texto de nome `fileIn` para um novo ficheiro de texto de nome `fileOut`.

- `public static void writeSquares (String fileIn, String fileOut) throws FileNotFoundException` que escreve num ficheiro de nome `fileOut` o quadrado de todos os inteiros contidos no ficheiro de nome `fileIn`. Assuma que no ficheiro `fileIn` cada linha contém apenas um valor inteiro.

- `public static void writeMultiples (String fileIn, String fileOut, int n) throws FileNotFoundException` que guarda num ficheiro de texto de nome `fileOut` todos os inteiros contidos no ficheiro de texto de nome `fileIn` que são múltiplos de `n`. Assuma que no ficheiro `fileIn` cada linha contém apenas um valor inteiro.

- `public static void lowerUpper (String fileIn, String fileOut) throws FileNotFoundException` que copia as linhas do ficheiro de texto de nome `fileIn` para um novo ficheiro de texto de nome `fileOut`, convertendo as letras todas em minúsculas e todas em maiúsculas, linha sim linha não.

- `public static void commonElements (String fileIn, String fileOut, int[] values) throws FileNotFoundException` que copia para um ficheiro de texto de nome `fileOut` os inteiros que aparecem num ficheiro de texto de nome `fileIn` que também aparecem no vetor `values`. Assuma que no ficheiro de nome `fileIn` cada linha contém apenas um valor inteiro. *Exemplo*: se o ficheiro de nome `fileIn` contiver os números 3, 1, 2, 3, 100, 10, 5, 3 e `values` for {3, 1, 10}, então no ficheiro de nome `fileOut` deverá escrever 3, 1, 3, 10, 3 (em linhas separadas).

ATENÇÃO. IMPORTANTE:
Tem disponível a classe `RunExercise0` e os ficheiros `myText.txt` e `myNumbers.txt` para testar a correção dos seus métodos.

- Na classe `RunExercise0` ponha em comentário (usando //) as linhas onde é feita a invocação dos métodos;
- De cada vez que implementa mais um método na classe `Exercise0`, volte a retirar os // da instrução que o invoca na classe `RunExercise0` para poder testá-lo;
- Para conseguir ver o(s) ficheiro(s) que os métodos criam, clique com o botão direito do rato no nome do projeto e escolha “Refresh”; para ver o conteúdo dos ficheiros de texto basta abri-los com um duplo clique.

Use também o conjunto de testes JUnit definidos na classe `TestsExercise0.java` para aferir a correção dos seus métodos.

Entrega

Embora este exercício não seja para avaliação e não precise de o entregar, pode fazer a sua entrega no Moodle de LabP, para experimentar, logo abaixo dos *links* para os materiais do exercício. Assim, para a próxima vez já não terá dificuldades.
Antes de entregar, certifique-se da correção da formatação, da correção da documentação, etc.

Deve criar o ficheiro `fcxxxxxExercise0.zip`, onde `xxxxx` é o seu número de aluno, contendo os ficheiros:
- `RunExercise0.java`
- `Exercise0.java` e
- `TestsExercise0.java`

ATENÇÃO: o zip deverá conter somente os ficheiros (não pode conter pastas)

Para criar o *zip* pode utilizar o ambiente gráfico ou a linha de comando. Se usar a linha de comando no Linux, a instrução será:

```
zip fcxxxxxExercise0.zip *.java
```

Posteriormente, submeta este zip no Moodle.