Semana 0

Exercício

Isabel Nunes, André Souto

Unidade Curricular de
Laboratório de Programação

2019/2020

Exercício

Objetivos

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

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`.

No exemplo seguinte demonstra-se uma utilização básica de canais de leitura e escrita para ficheiros de texto, lendo inteiros de um ficheiro `teste.txt` (um por linha) e escrevendo no segundo ficheiro `resultados.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 `teste.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 apenas se avisa que tais exceções 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 leitor = new Scanner(new File("teste.txt"));
```

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

```
PrintWriter escritor = new PrintWriter("resultado.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;

/**
 * Exemplo de como ler um ficheiro de texto com inteiros (um por linha),
 * processar o seu conteudo, e escrever o resultado noutro ficheiro
 * de texto.
 */
public class ExemploLeituraEscritaFichDemo {

    /**
     * Abrir um ficheiro para leitura e outro para escrita.
     * Ler os valores do primeiro e escrever os que sao impares no
     * segundo ficheiro
     */
    public static void main(String[] args) throws IOException {

        // canal de leitura
        Scanner leitor = new Scanner(new File("teste.txt"));
        // canal de escrita
        PrintWriter escritor = new PrintWriter("resultado.txt");

        // enquanto o ficheiro nao terminar (soh contem inteiros)
        while (leitor.hasNextInt()){
            int valor = leitor.nextInt();
            if(valor % 2 != 0){
                escritor.println(valor);
            }
        }

        leitor.close();
        escritor.close();
    }
}
```

O que fazer

Neste primeiro trabalho deve ler atentamente os outros documentos referentes à Semana 0, especialmente o documento que explica o processo de submissão e a introdução ao Eclipse.

Usando como referência o ficheiro `RunSemana0.java`, escreva a classe `ExemploFicheiros.java` com métodos com as seguintes assinaturas:

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

- **`public static void guardaMultiplos (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 minusculasMaiusculas (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 elementosEmComum (String fileIn, String fileOut, int[] vals) 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 `vals`. 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 `vals` for {3, 1, 10}, então no ficheiro de nome `fileOut` deverá escrever 3, 1, 3, 10, 3 (em linhas separadas).

Pode usar os ficheiros `texto.txt` e `numeros.txt` fornecidos para testar a correção dos seus métodos. Tem disponível uma classe `RunSemana0` para o ajudar na tarefa, bem como um conjunto de testes definidos na classe `ExemploFicheirosTest.java` para aferir a correção dos seus métodos.

Entrega

Antes de entregar, certifique-se da correção da formatação, da correção da documentação, etc…
Deve criar o ficheiro **semana0.zip**, contendo os ficheiros:

- RunSemana0.java
- ExemploFicheiros.java e
- ExemploFicheirosTest.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.

Posteriormente, deve seguir as indicações dadas no documento que descreve os procedimentos de submissão.