Laboratório 18 - Leitura e Escrita de Ficheiros II (27/10/2014)

1.

```
package lab18;

import java.io.BufferedInputStream;
import java.io.BufferedOutputStream;
import java.io.BufferedReader;
import java.io.BufferedWriter;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.FileOutputStream;
import java.io.FileReader;
import java.io.FileWriter;
import java.io.IOException;
import java.io.InputStream;
import java.util.StringTokenizer;

/**
 * This class implements utility methods to perform operations on files.
 * @author FilipaG
 */
public class FileUtils {
```

```
        /**
         * The method copyTextFile copies the contents of a text file to another. This method
         * receives as parameter the path of the original file and the path where the copy will be.
         * @param source
         * @param dest
         * @return true if the copy was successful, false otherwise.
         */
        public static boolean copyTextFile(String source, String dest)
        {
            try(BufferedReader reader = new BufferedReader(new FileReader(source));
                BufferedWriter writer = new BufferedWriter(new FileWriter(dest)))
                {
                    String nextLine = reader.readLine();
                    while(nextLine != null)        //enquanto a linha não for nula
                    {
                        writer.write(nextLine);
                //impressão no ficheiro de output da linha copiada do ficheiro de Input

                        writer.newLine();       //criação de uma nova linha
                        nextLine = reader.readLine();
                        //Leitura da próxima linha do ficheiro de input
                        }
                        reader.close(); // fechar o BufferedReader
                        writer.close();   //fechar o PrintReader
            }
            catch (FileNotFoundException e)
        //excepção: caso o ficheiro não seja encontrado
            {
                System.out.println("File not found or could not be opened");
                e.printStackTrace();
                return false;
            }
            catch (IOException e)
            {
                System.out.println("Error reading from file");
                e.printStackTrace();
                return false;
            }
            return true;
        }
```

2.

```
        /**
         * The copyBinaryFile method makes a copy of a binary file. This method receives as
parameter
         * the path of the original file and the path where the copy will be.
         * @param source
         * @param dest
         * @return true if the copy was successful, false otherwise.
         */
        public static boolean copyBinaryFile(String source, String dest)
        {
           try(BufferedInputStream input = new BufferedInputStream(new
FileInputStream(source));BufferedOutputStream output = new BufferedOutputStream(new
FileOutputStream(dest)))
           {
               int nextDataByte = input.read();
               while(nextDataByte != -1)
               {
                  output.write(nextDataByte);   //impressão no ficheiro de output
                  nextDataByte = input.read();
               }
               input.close();
               output.close();
           }
           catch(FileNotFoundException e)
           {
               System.out.println("File not found or could not be opened");
               e.printStackTrace();
               return false;
           }
           catch (IOException e1)
           {
               System.out.println("Error reading the file");
               e1.printStackTrace();
               return false;
           }
           return true;
        }
```

3.

```
        /**
         * The parseTextFile method makes parse of a text file into an array of Strings
as indicated
         * by the delimiter parameter.
         * @param source
         * @param delim
         * @return
         */
        public static String[] parseTextFile(String source, String delim)
        {
           try(BufferedReader reader = new BufferedReader(new FileReader(source)))
// criar um objecto do tipo BufferReader que tem o contrúdo fo ficheiro source
           {
               String text = "";
               String nextLine = reader.readLine();
               //String que tem o conteúdo da próxima linha

               while(nextLine != null)
               // este ciclo cria uma String com o conteúdo total do ficheiro
               {
                  text+= nextLine;
                  nextLine = reader.readLine();
               }

               StringTokenizer tokenizer =  new StringTokenizer(text, delim);
//instanciação de um objeto do tipo StringTokenizer que "parte" o conteúdo da string
```

```
text em tokens delimitados pela string "delim"

            String[] parsedText = new String[tokenizer.countTokens()];
            // array onde serão colocados todos os tokens da string text

            int i = 0;
            while (tokenizer.hasMoreElements())
//enquanto houver elementos no tokenizer armazena-os como elementos do array parseText
            {
               parsedText[i] = tokenizer.nextToken();
            // System.out.println(parsedText[i]);   -> utilizei apenas para testar se o
      ciclo estava a fazer o que era pedido
               i++;
            }
            reader.close();
            return parsedText;
         }
         catch (FileNotFoundException e)
         {
            return null;
         }
         catch (IOException e)
         {
            return null;
         }
      }
```

```
/**
   * This method reads the fileIn file and copy the file text to fileOut, but
   * substitutes with newWord whenever a specific word WordToFind occur. The method
   * WordToFind tests whether the word is part of a longer word, analyzing the
   * character immediately before the beginning of the word and immediately following the
   * end of wordToFind.
   * @param fileIn
   * @param fileOut
   * @param wordToFind
   * @param newWord
   * @return
   */
      public static boolean copyAndReplace(String fileIn, String fileOut, String
wordToFind, String newWord) {

         try (BufferedReader br = new BufferedReader(new FileReader(fileIn));
            PrintWriter pw = new PrintWriter(fileOut)) {
            String line = null;
            String newLine="";
            while ((line = br.readLine()) != null) {

            // verifica se a linha tem a palavra referente à wordToFind
               while(line.contains(wordToFind)) {

//verificar se a string wordTofind não faz parte de nenhuma palavra, ou seja, se o
caracter que a antecede e a sucede são " "

//obter o caracter que está depois da palavra WordToFind
                  char afterWordToFind =
line.charAt(line.indexOf(wordToFind)+wordToFind.length());

//índice da posição onde inicia a nova string (na primeira posição após a palavra até ao
fim da linha)
                  int beginIndex = line.indexOf(wordToFind)+wordToFind.length();

//índice da posição onde finaliza a string (na primeira posição da palavra wordToFind)
                  int endIndex = line.indexOf(wordToFind);
```

```
// Se WordToFind é a primeira palavra da linha, verificar se o caracter após o
wordToFind é letra
                    if ((line.indexOf(wordToFind)==0)&& (!isALetter(afterWordToFind)))
                    {
                       newLine = newWord;
                    }

              // Se wordToFind contém o último caracter ficheiro
                    else if ((beginIndex==line.length())&&(!isALetter(line.charAt(endIndex-1))))
                    {
                       newLine += newWord;
                    }

              // beforeWordTofind=line.charAt(endIndex-1))
                    else if ((!isALetter(afterWordToFind))&&(!isALetter(line.charAt(endIndex-
1))))
                    {
                       newLine += line.substring(0,endIndex) + newWord ;
// A linha já obtida, substituir a palavra "wordToFind" pela palavra "newWord" e
adicionar
                    }
                    else
// Se a palavra "wordToFind" fizer parte de uma palavra maior, não se faz qualquer
sibstituição
                    {
                       newLine += line.substring(0,beginIndex);
                    }

                    line = line.substring(beginIndex);   //a linha a analisar passa a
ser apenas da primeira posição após a ocorrência da palavra wordToFind até ao fim da
linha.//é esta nova linha que será novamente analisada para verificar a ocorrência da
palavra wordToFind
                }
                newLine +=line;
                pw.println(newLine);  //impressão das linhas de texto no ficheiro
fileOut
                    }

                br.close(); // fechar o BufferedReader
                pw.close();   //fechar o PrintReader

             } catch (FileNotFoundException e)        // se o ficheiro não for
encontrado
             {
                System.out.println("File not found or could not be opened");
                e.printStackTrace();

             } catch (IOException e) {
                System.out.println("Error reading from file");
                e.printStackTrace();
             }
             return true;
       }
```

```
   /**
    * This method verifies if the string wordTofind is not a part of any word, that is,
if the character that
    * precedes or succeeds is not a letter.
    * @param char1
    * @return true if the character is a letter, and false otherwise
    */
       public static boolean isALetter(char char1){
          if ((char1 > 96 && char1 < 123 )||(char1 > 64 && char1 < 91))// testar se está
entre a e z ou A e Z
          {
             return true;
```

```
            }
            return false;
        }
```

```
    public static void main (String[] args){
        String FileIn = "C:\\Users\\Filipa\\workspace\\sessão21_10\\src\\lab18\\text.txt";
        String FileOut = "C:\\Users\\Filipa\\workspace\\sessão21_10\\src\\lab18\\text_copy.txt";

          copyTextFile(FileIn, FileOut);

        String binarySource =
"C:\\Users\\Filipa\\workspace\\sessão21_10\\src\\lab18\\ImageTest.jpg";
        String binaryDest =
"C:\\Users\\Filipa\\workspace\\sessão21_10\\src\\lab18\\ImageTest_copy.jpg";
        copyBinaryFile(binarySource, binaryDest);

        parseTextFile(FileIn, "|");

       String fileIn = "C:\\Users\\Filipa\\workspace\\sessão21_10\\src\\lab17\\ZeZambeze.txt";
       String fileOut = "C:\\Users\\Filipa\\workspace\\sessão21_10\\src\\lab18\\JoseZambeze.txt";
       String wordToFind = "Ze";
       String newWord = "Jose";

       copyAndReplace(fileIn, fileOut, wordToFind, newWord);

             }
   }
```

Testes:

```
package lab18;

import java.io.BufferedReader;
import java.io.BufferedWriter;
import java.io.FileNotFoundException;
import java.io.FileReader;
import java.io.FileWriter;
import java.io.IOException;
import org.junit.Test;
import org.junit.Assert;

/**
 * Case tests for {@link FileUtils} class.
 *
 * @author Challenge.IT
 *
 * Copyright (c) 2014, Challenge.IT and/or its affiliates. All rights reserved.
 * DO NOT ALTER OR REMOVE COPYRIGHT NOTICES OR THIS FILE HEADER.
 *
 * This code is distributed in the hope that it will be useful for learning purposes,
but WITHOUT
 * ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or
 * FITNESS FOR A PARTICULAR PURPOSE.
 * */

public class FileUtilsTest1
{
       @Test
       public void shouldCopyTextFileToAnotherPath()
       {
          // Arrange
          String source =
"C:\\Users\\Filipa\\workspace\\sessão21_10\\src\\lab18\\text.txt";
          String dest   =
"C:\\Users\\Filipa\\workspace\\sessão21_10\\src\\lab18\\text_copy.txt";

         // Act

         // Assert
```

```
            Assert.assertTrue(FileUtils.copyTextFile(source, dest));
        }


        @Test
        public void shouldCopyBinaryFileToAnotherPath()
        {
            // Arrange
            String source =
"C:\\Users\\Filipa\\workspace\\sessão21_10\\src\\lab18\\ImageTest.jpg";
            String dest   =
"C:\\Users\\Filipa\\workspace\\sessão21_10\\src\\lab18\\ImageTest_copy.jpg";

            // Act

            // Assert
            Assert.assertTrue(FileUtils.copyBinaryFile(source, dest));
        }


        @Test
        public void shouldParseTextFile()
        {
            // Arrange
            String source =
"C:\\Users\\Filipa\\workspace\\sessão21_10\\src\\lab18\\text.txt";
            String delim = "|";

            // Act
            String[] parsedText = FileUtils.parseTextFile(source, delim);

            // Assert
            Assert.assertNotNull(parsedText);
            Assert.assertTrue(parsedText.length == 3);
            Assert.assertEquals(parsedText[0], "RANDSTAD FORMACAO");
            Assert.assertEquals(parsedText[1], "LABORATORIO 18");
            Assert.assertEquals(parsedText[2], "JAVA IO");
        }


        @Test
        public void shouldCopyAndReplaceAWordToAnotherPath()
        {
            //Arrange
            String fileIn =
"C:\\Users\\Filipa\\workspace\\sessão21_10\\src\\lab17\\ZeZambeze.txt";
            String fileOut =
"C:\\Users\\Filipa\\workspace\\sessão21_10\\src\\lab18\\JoseZambeze.txt";
            String wordToFind = " Ze ";
            String newWord = " Jose ";

            Assert.assertTrue(FileUtils.copyAndReplace(fileIn, fileOut, wordToFind,
newWord));
        }


        @Test
        public void shouldNotHaveWordToFind() throws IOException
        {
//Para não estar a testar num ficheiro inteiro, podemos criar uma String com os casos
problemáticos. Depois inserimo-la num ficheiro para correrrmos o método copyAndReplace e
comparamos a string que esperamos como resultado com o conteúdo do ficheiro produzido
pelo método.
            String line1 = "Ze foi fazer um Safari ao Zambeze";
            String line1out = "Jose foi fazer um Safari ao Zambeze";
```

```
          String fileIn =
"C:\\Users\\Filipa\\workspace\\sessão21_10\\src\\lab18\\OriginalTest.txt";
          String fileOut =
"C:\\Users\\Filipa\\workspace\\sessão21_10\\src\\lab18\\ReplaceTest.txt";

          BufferedWriter bw = new BufferedWriter(new FileWriter(fileIn));
          bw.write(line1);   //escrita da linha no ficheiro
          bw.close();
// podemos fazer flush ou close -> o objectivo é que seja feita a descarga do conteúdo
do buffer no ficheiro

          FileUtils.copyAndReplace(fileIn, fileOut, "Ze", "Jose");
          BufferedReader br = new BufferedReader(new FileReader(fileOut));
          Assert.assertEquals(line1out, br.readLine());        // verificação se a
String produzida é igual à string esperada
       }
}
```