Lab 6 : Files, Exceptions and Objects – Selected Solutions

Exercise 1: Write a program that will generate 15 random characters (a to z) and print them to a file called `randomChars.txt`.

```java
import java.io.File;
import java.io.FileNotFoundException;
import java.io.PrintWriter;

public class RandomChars {
   public static void main(String[] args) {
      File outputFile = new File("randomChars.txt");
      try {
         PrintWriter out = new PrintWriter(outputFile);
         for(int i = 0; i < 15; i++) {
            out.print(getRandomLetter());
         }
         out.close();
      } catch (FileNotFoundException e) {
         // If randomChars.txt can't be written to,
         // print an error message
         System.err.println("Cannot write to " + outputFile);
      }
   }

   public static char getRandomLetter() {
      // range = 26 possible values
      int range = ('z' - 'a' + 1);
      // get random number from 0 to 25
      int offset = (int) (Math.random() * range);
      // get the letter that is <offset> positions from 'a'
      return (char)('a' + offset);
   }
}
```

Exercise 2: Write a program that takes one command-line argument `<filename>`. The program will then open and read the file given by `<filename>.txt`, read each line in the file, and sort the lines in lexicographical order (remember: `Arrays.sort(...)`). The sorted lines will be output to `<filename>-sorted.txt`.

```java
import java.io.File;
import java.io.FileNotFoundException;
import java.io.PrintWriter;
import java.util.Arrays;
import java.util.Scanner;

public class SortFile {
   public static void main(String[] args) {
      String unsortedFilename = args[0] + ".txt";
      String sortedFilename = args[0] + "-sorted.txt";

      String[] fileContents = readContents(unsortedFilename);
      Arrays.sort(fileContents);

      File outputFile = new File(sortedFilename);
      try {
         PrintWriter out = new PrintWriter(outputFile);
         for(int i = 0; i < fileContents.length; i++) {
            out.println(fileContents[i]);
         }
         out.close();
      } catch (FileNotFoundException e) {
         System.err.println("Cannot write to " + outputFile);
      }
   }

   /**
    * Get the contents of a file in a String array
    */
   public static String[] readContents(String filename) {
      File file = new File(filename);
      try {
         int lineCount = getLineCount(filename);
         String[] contents = new String[lineCount];

         Scanner fileScanner = new Scanner(file);
         // Read the contents of the file
         int index = 0;
         while(fileScanner.hasNextLine()) {
            contents[index] = fileScanner.nextLine();
            index = index + 1;
         }

         fileScanner.close();
         return contents;
      } catch (FileNotFoundException e) {
         System.err.println("Cannot read " + filename);
         return null;
      }
   }


   /**
    * Count the number of lines in a file.
    */
   public static int getLineCount(String filename)
         throws FileNotFoundException {
      int count = 0;
      Scanner fileScanner = new Scanner(new File(filename));
      while(fileScanner.hasNextLine()) {
         fileScanner.nextLine();
         count = count + 1;
      }
      fileScanner.close();
      return count;
   }
}
```

Exercise 3: Write a program that takes in three arguments `filename`, `findWord` and `replaceWord`. Your program should replace all instances of `findWord` with `replaceWord`.

Extension 1: Modify the program from Exercise 3 to make the find part case insensitive. For the more adventurous, try accepting a **regular expression** instead of a search word. This can be quite powerful!

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

import java.util.regex.Matcher;
import java.util.regex.Pattern;

public class FindAndReplace {
   public static void main(String[] args) {
      String filename = args[0];
      String findWord = args[1];
      String replaceWord = args[2];

      String fileContents = readContents(filename);
      fileContents = fileContents.replaceAll(findWord, replaceWord);

      // (extension) Using case-insensitive regex:
      Pattern regex = Pattern.compile(findWord, Pattern.CASE_INSENSITIVE);
      Matcher matcher = regex.matcher(fileContents);
      fileContents = matcher.replaceAll(replaceWord);

      File outputFile = new File(filename);
      try {
         PrintWriter out = new PrintWriter(outputFile);
         out.println(fileContents);
         out.close();
      } catch (FileNotFoundException e) {
         System.err.println("Cannot write to " + outputFile);
      }
   }

   /** Get the contents of a file as a single String */
   public static String readContents(String filename) {
      File file = new File(filename);
      try {
         StringBuilder contents = new StringBuilder();

         Scanner fileScanner = new Scanner(file);
         while(fileScanner.hasNextLine()) {
            contents.append(fileScanner.nextLine());
            // add a new line after each line as
            // the scanner removes them
            contents.append(System.lineSeparator());
         }

         fileScanner.close();
         return contents.toString();
      } catch (FileNotFoundException e) {
         System.err.println("Cannot read " + filename);
         return null;
      }
   }
}
```

Exercise 4: In Exercise 7 of Lab 5, you wrote a program that takes two command-line arguments: the score and the maximum possible score. The program will calculate the percentage score of the input numbers.

What kind of things can go wrong in this program? Revise this program to make sure the program can handle all the possible cases that you can think of. This means you'll need to work with **exceptions**.

```java
public class Scores {
   public static void main(String[] args) {
      int inputScore = 0;
      int intputTotal = 0;

      // Read the two integers from arguments
      try {
         inputScore = Integer.parseInt(args[0]);
         intputTotal = Integer.parseInt(args[1]);
      } catch(ArrayIndexOutOfBoundsException e) {
         System.err.println("Not enough arguments entered.");
         return;
      } catch (NumberFormatException e) {
         System.err.println("Please enter integer numbers only.");
         return;
      }

      // Attempt to print the percentage
      try {
         double percentage = calculatePercentage(inputScore, intputTotal);
         System.out.println("Percentage: " + percentage);
      } catch (ArithmeticException e) {
         // Get the error message from the error object 'e'
         System.err.println(e.getMessage());
      }
   }

   public static double calculatePercentage(int score, int total)
         throws ArithmeticException {
      // Let the method called handle the problem by
      // 'throw'ing the exception up the hierarchy
      if(total <= 0) {
         throw new ArithmeticException("Total must be a positive number.");
      }
      return ((double)score / total) * 100.0;
   }
}
```

Exercise 6: Write a program to read in four numbers as command line inputs in the order `x1`, `y1`, `x2`, `y2`, and then output the distance between the points `(x1, y1)` and `(x2, y2)`.

```java
import java.awt.Point;

public class PointDistance {
   public static void main(String[] args) {
      int x1 = Integer.parseInt(args[0]);
      int y1 = Integer.parseInt(args[1]);
      int x2 = Integer.parseInt(args[2]);
      int y2 = Integer.parseInt(args[3]);

      Point p1 = new Point(x1, y1);
```

```
        Point p2 = new Point(x2, y2);

        double distance = p1.distance(p2);
        System.out.println("Distance between points: " + distance);
    }
}
```

Exercise 8: Write a program that will take in a long integer, and print out what the date and time will be in that many milliseconds.

```
import java.util.Date;

public class TimeFromNow {
    public static void main(String[] args) {
        long time = Long.parseLong(args[0]);
        Date now = new Date();
        now.setTime(now.getTime() + time);
        System.out.println("Time will be " + now.toString());
    }
}
```