Homework: Streams and Files

This document defines the homework assignments from the Java Fundamentals Course @ Software University. Please submit as homework a single `zip` / `rar` / `7z` archive holding the solutions (source code) of all below described problems. The solutions should be written in C#.

Problem 1. Sum lines

Write a program that reads a text file and prints on the console the sum of the ASCII symbols of each of its lines. Use **BufferedReader** in combination with **FileReader**.

lines.txt	Output
Kvo staa? Nishto, ti? Chuk, maina.	824 989 1035

Problem 2. ALL CAPITALS!

Write a program that reads a text file and changes the casing of all letters to upper. The file should be overwritten. Use **BufferedReader**, **FileReader**, **FileWriter, and PrintWriter**.

lines.txt	lines.txt
Kvo staa? Nishto, ti? Chuk, maina.	KVO STAA? NISHTO, TI? CHUK, MAINA.

Problem 3. Count character types.

Write a program that reads a list of words from the file **words.txt** and finds the count of vowels (*гласни букви*), consonants (*съгласни*) and other punctuation marks. Since English is a bit tricky, assume that **a, e, i, o, u** are vowels and all others are consonants. Punctuation marks are (**!,.?**). Do not count whitespace.

Write the results in file **count-chars.txt**.

words.txt	count-chars.txt
Thanks to us, you owe it to the Chinese.	Vowels: 13 Consonants: 17 Punctuation: 2

Problem 4. Copy .jpg File

Write a program that copies the contents of a .jpg file to another using **`FileInputStream`**, **`FileOutputStream`**, and **byte[] buffer**. Set the name of the new file as **my-copied-picture.jpg.**

Problem 5. Save an ArrayList of doubles

Write a program that saves and loads the information from an ArrayList to a file using **`ObjectInputStream`**, **`ObjectOutputStream`**. Set the name of the new file as **doubles.list**

Problem 6. *Save a Custom Object in a file

Write a program that saves and loads the information from a custom Object that you have created to a file using `ObjectInputStream, ObjectOutputStream`. Create a **class Course** that has a **String field** containing the **name** and an **integer field** containing the **number of students** attending the course. Set the name of the new file as **course.save.**

Problem 7. *Create Zip Archive

Write a program that reads three txt files **words.txt**, **count-chars.txt** and **lines.txt** and create a zip archive named **text-files.zip.** Use **FileOutputStream**, **ZipOutputStream**, and **FileInputStream.**

Problem 8. ***CSV Database

Write a console application that keeps records in three files **students.txt** and **grades.txt.** Data should be comma-separated – **{**student-id, first-name, last-name, age, home-town**}**. (e.g. **5,Georgi,Ivanov,14,Novi Pazar**). Grades should be in format **{**student-id, course1 grades, course2 grades**}** (e.g. **5,Math 2.00 2.00 3.50,Literature 4.00 5.25).** The relation between the two files is the student id.

Implement the following commands:

- **Search-by-full-name**
  - Example: Search-by-full-name Georgi Ivanov ->
    - Georgi Ivanov (age: 14, town: Novi Pazar)
    - # Math: 2.00, 2.00, 3.50
    - # Literature 4.00, 5.25
  - Search-by-full-name Georgi Mamarchev ->
    - **Student does not exist**
- **Search-by-id**
  - Example: Search-by-id 5 ->
    - Georgi Ivanov (age: 14, town: Novi Pazar)
    - # Math: 2.00, 2.00, 3.50
    - # Literature 4.00, 5.25
  - Search-by-id 8 ->
    - **Student does not exist**
- **Delete-by-id** -> deletes the student and his grades or returns "Student does not exist"
- **Update-by-id** -> updates student's info/grades or returns "Student does not exist"
- **Insert-student**
  - Example: **Insert-student Georgi Mamarchev 19 Sofia** -> adds a new student and assigns to him/her the greatest id + 1.
    - 6,Georgi,Mamarchev,19,Sofia
- Insert-grade-by-id
  - Example: **Insert-grade-by-id 5 Math 4.00**
    - 5,Math 2.00 2.00 3.50 **4.00**,Literature 4.00 5.25
  - **Insert-grade-by-id 8 Literature 6.00**
    - Student does not exist

HINT FOR DELETE, UPDATE, and INSERT: Read the contents of the file that will be changed and keep them in appropriately structured HashMap. Perform the necessary operations and then overwrite the file with the new data.