159.272 Programming Paradigms

Tutorial 3: Implementing Interfaces To Sort Data

Objectives

1. understand how to use interfaces
2. use java.io
3. sort objects
4. use exception handling

Deadline

- see stream
- no late submissions allowed

Instructions

Your task is to create a class that can sort rows in csv[1] files containing student data.

1. download students.csv from stream
2. create an Eclipse project
3. copy students.csv into the Eclipse project folder
4. in this project, create a package nz.ac.massey.cs.pp.tutorial3.id<yourstudentid>, replace <yourstudentid> by your student number, like nz.ac.massey.cs.pp.tutorial3.id42
5. within this package, create a class CSVSorter
    1. this class is executable, i.e. it has a main method
    2. the main method has three parameters:
        1. the first parameter is the name of an csv input file
        2. the second parameter is the name of a csv output file
        3. the third parameter is the sort key, the following values are possible: by_id, by_firstname, by_name, by_program
    3. that means that you can run the program as follows:
       java nz.ac.massey.cs.pp.tutorial3.id42.CSVSorter students.csv sorted.csv by_id
    4. the output file contains the student data from the input file, sorted according to the criteria used (ascending), note that the first line (containing attribute names) should not be changed by sorting
6. create a class Student within the same package
    1. each row in the input csv file should first be converted to an instance of Student
    2. then all Student instances are sorted and saved into the output file
7. to sort student instances, use Arrays.sort(Object[],Comparator)
8. to implement the different sort strategies, implement one class per strategy for the interface java.util.Comparator (e.g., ByIdComparator)
9. if the names of input and output file are the same, the program should throw a java.lang.IllegalArgumentException

Deliverables

1 CSV stands for comma separated file – this is a popular simple file format to store tabular data. Most spreadsheet programs like OpenOffice and MS Office can import/export CSV files.

1. export your Eclipse project as follows to a zip file:
    1. in Eclipse, select Project, then select File > Export > General > Archive File
    2. once the project has been exported, check the zip file created (use WinZip, 7Zip or similar) to make sure that the sources code files, students.csv, .project and .classpath are all included. .project and .classpath might not be visible, you may need to enable "show hidden files" in your zip tool or OS file explorer.
2. upload this zip file to stream

Prerequisites

This is what I expect **internal** students to know **before** they come to the tutorial. This means that I expect internal students to prepare for this tutorial. Failure to do so may result in the tutorial being marked as 0. The reason for this is that we cannot afford to waste time in the labs.

Students are expected to:

1. create packages, projects and executable Java classes in Eclipse
2. understand how runtime parameters are passed to Java programs in Eclipse
3. understand interfaces and inheritance, in particular the use of the Comparable interface discussed in the lectures on inheritance

Grading

1. this tutorial is worth 1%
2. the maximum number of points you can get is 2 (this will be scaled to allow for half points)
3. internal students have to submit **and** to physically attend the tutorial to get points – make sure your name is signed off by the lecturer or tutor before you leave the lab

Hints

1. to find out how a class works, google for the fully qualified class name, and go to http://docs.oracle.com/javase/7/docs/api/
2. classes you might find useful to read from text files line by line:
    1. java.io.BufferedReader, in particular the method readLine
    2. java.util.StringTokenizer , this will be useful to parse a single lines
3. classes you might find useful to write text to files:
    1. java.io.FileWriter
4. if you reference files without the full path name ("students.csv", and note "c:/temp/students.csv"), your java program will look for these files in the project folder (do not hardcode file names in the main method ! they are supplied as parameters)