Practical-Exam-04: Sorting and Searching Algorithms

Due 20 May by 16:55 **Points** 100
Available 20 May at 15:00 - 20 May at 17:00 2 hours

This assignment was locked 20 May at 17:00.

Introduction

This session ends at. 14:55 (local session) and 16:55 (online session)

Make sure to complete your submission in time.

Submit to SVN & Websubmission (for every problem solved)

Note that on the top of each file you will see an indication whether any change is necessary or not.

Submission

- Create the repository on the SVN server
- You must checkout only the folder practical-exam-04 from your server
- No other folders from your SVN will be allowed during the practical exam.
- Check if your repository is being track by WebSubmission before start solving the exam.

```
svn mkdir -m "first assessment commit" --parents https://version-control.adelaide.edu.au/svn/<
YOUR-ID>/2021/s1/fcs/week-10/practical-exam-04
```

Assessment

- **This is a practical exam - your work must be entirely your own.**
- Marks for this practical exam contribute 2 marks.
- Marks will be awarded later by your workshop supervisor (30%) and websubmission marker (70%).
- **Due date: 5 minutes before end of your session**
- **Do Not Forget** To Submit on **WebSubmission (https://cs.adelaide.edu.au/services/websubmission/index.php?sub_course=fcs)** and select the

correct time slot

- **Late penalties:** Only work submitted during your enrolled practical session from a Linux system in the practical lab will be marked.

Regarding functional marks, please consider:

```
(1) only codes that can compile will be marked;
(2) only codes that are in the suggested directory will be marked;
(3) only codes submitted to SVN and WebSubmission before the deadline will be marked;
(4) only codes containing your signature on the top of the file will be marked by tutors;
(5) you will have your markers decreased in 3 points if *.class file present in your folder
s;
```

Note

- To acquire full marks **(1)** all your functionalities must be working perfectly, **(2)** your code has to be well and proportionally commented, and **(3)** your code must follow correct indentation (4 spaces for every new indentation level) and **(4)** you have to use all the content from latest lectures.
- We argue that you are not just asked to solve a problem but use the more sophisticated way to solve it. For instance, you can solve a problem using ten variables, but it will always be better to solve the same problem with an array.

Practical Exam 04

Download: project.zip (https://myuni.adelaide.edu.au/courses/74996/files/10490426?wrap=1) (https://myuni.adelaide.edu.au/courses/74996/files/10490426/dowr download_frd=1)

```
How to unzip "project.zip"

1. Move project.zip from Download folder to ...week-10/practical-exam-04/
2. In the terminal,
    2.1 Navigate to practical-exam-04
    for instance: cd  practical-exam-04/


    2.2 Unzip project.zip into;
    unzip project.zip

    2.3 Remove project.zip
    rm project.zip

    2.4. svn add project
```

```
3. svn commit (to get the starting files into your directory)

4. cd to project (now you are ready to start)
```

Hungry Donalds Fast Food

Dear students,

In this practical exam, you are a developer that works for the company **FCS-SA** Systems. Your company's main client is the fast food chain Hungry Donalds. This morning, your IT manager emailed you a new requirement for the system and a few issues reported by your company's client.

For this exam you must:

1. Read the attached emails. The one from the user (Beyonce Smith) lists two problems with the code in Project (look in the text for "Problem 1" and "Problem 2"). The manager email (from A. Boss) lists a third problem ("Problem 3").
2. Fix the three issues reported (committing as you go).
3. Write a reply email to your manager with a clear explanation of the problems and how you fixed these (this email will allow you to recover a proportion of the functionality marks if you do not many marks for automatic testing).

The emails are:

The Client Email (https://myuni.adelaide.edu.au/courses/74996/files/10490646/download?wrap=1) (https://myuni.adelaide.edu.au/courses/74996/files/10490646/download?download_frd=1) (can also be found in src/user_email.txt in project folder)

The Manager's Email (https://myuni.adelaide.edu.au/courses/74996/files/10490430/download?wrap=1) (https://myuni.adelaide.edu.au/courses/74996/files/10490430/download?download_frd=1) (can also be found in src/it_management.txt in project folder)

Write your reply 'email' to:

src/your_explanation_email.txt

*Do **NOT** open your email or send anything. Simply write your response in the text file as below (and add and commit it to svn)*

```
Dear manager,

The problem (1), was due to <your explanation>, I fixed doing <your explanation>;
The problem (2), was due to <your explanation>, I fixed doing <your explanation>;
The problem (3), was due to <your explanation>, I fixed doing <your explanation>;

Kind regards, <your name> <your id>
```

You will be marked on this response. Even if you have not completed all the problems, you should write the 'email' and commit it to SVN as you make progress this response will serve as a fall-back for some marks in case you miss out on some marks for functionality. (Note that this document will become **important** as a way to get **some** marks if you miss out on more than 50% of your marks for functionality).

Running the Starting Code

```
Instructions to run the starting code:


1. Navigate to your project folder;
2. Run:

     javac Main.java
     java Main

Instructions for use the software:

1. Function (1): load staff members, you MUST run it as the first action when starting the sy
stem.;
2. Function (2): display staff members. After loading staff members, you can display their in
fo on the screen;


Other functions: check Menu::Help;
```

Problem 01: Sorting Staff by Name

Please, you are asked to follow the client requirement on the **problem (1),** all the information is reported on the email. In addition, you are also asked to implement accessors and mutators for basic classes;

The built-in String method **.compareTo()** may be helpful.
https://www.tutorialspoint.com/java/java_string_compareto.htm
(http://www.tutorialspoint.com/java/java_string_compareto.htm)

If there are two strings, *s1* and *s2*. Then *s1.compareTo(s2)* will give:

```
if s1 == s2 : 0
if s1 > s2: an int more than 0
if s1 < s2: an int less than 0
```

```
'''

Signatures:

On this practical exam, you are NOT asked to design any signature other than accessors and m
utators;
In addition, every file in the project.zip has a hint whether you need to perform changes or
not.
```

```
Requirements:
1. Set accessors and mutator for all properties;
2. Resolve the problem (1) reported by our client;
3. Do not crash the system;
```

```
Repository

save this project and it's files in project/
add to your svn repository

Note: keep a track of your version. By the end of the session time you MUST submit a copy tha
t can be compiled.


Important
1. Your classes MUST NOT contain a public static main function.
2. You MUST NOT perform changes on the files:
    a. BinarySearch.java
    b. LinearSearch.java
    c. Search.java
    d. Main.java
    e. Sort.java
    f. SortInterface.java
3. If you want to perform Unit Test, please use the class Test.java
'''
```

Problem 02: Fixing the behavior when Searching for staff members

Please, you are asked to follow the client requirement on the **problem (2),** all the information is reported on the email.

```
'''

Signatures:

On this practical exam, you are not asked to design any signature other than accessors and mu
tators;
In addition, every file in the project.zip has a hint whether you need to perform changes or
not.

Requirements:
1. Resolve the problem (2) reported by our client;
2. Do not crash the system;

Repository

save this project and it's files in project/
add to your svn repository

Note: keep a track of your version. By the end of the session time you MUST submit a copy tha
t can be compiled.
```

```
Important
1. Your classes MUST NOT contain a public static main function.
2. You MUST NOT perform changes on the files:
    a. BinarySearch.java
    b. LinearSearch.java
    c. Search.java
    d. Main.java
    e. Sort.java
    f. SortInterface.java
3. It's likely that the bug in this code is hidden in the file Company.java; There are seve
ral ways of solving this issue, feel free to use your creativity.
4. If you want to perform Unit Test, please use the class Test.java
'''
```

Problem 03: Implement new features (Selection Sort)

To acquire marks: you need **sortIntByIndex**, the other methods are optional.

Please, you are asked to follow the manager requirement on the **problem (3),** all the information is on the email.

```
'''

Signatures:

On this practical exam, you are not asked to design any signature other than accessors and muta
In addition, every file in the project.zip has a hint whether you need to perform changes or no

Requirements:
1. Resolve the problem (3) required by your manager;
2. Do not crash the system;

Repository

save this project and it's files in project/
add to your svn repository

Note: keep a track of your version. By the end of the session time you MUST submit a copy that 
mpiled.


Important
1. Your classes MUST NOT contain a public static main function.
2. You MUST NOT perform change on the files:
    a. BinarySearch.java
    b. LinearSearch.java
    c. Search.java
    d. Main.java
    e. Sort.java
    f. SortInterface.java
3. It's likely that the bug in this code is hidden in the file Selection.java and Company.ja
4. If you want to perform Unit Test, please use the class Test.java
'''
```

Practical Exam 04 Rubric

Criteria	Ratings				Pts
Functional	**70 Pts** **Excellent** Your code (1) perform all the functions correctly, (2) use latest concepts learned in class, (3) has a clear, creative and sophisticated way of solving the problem.	**35 Pts** **Fair** Your code (1) Peforms half of the functionality correctly (2) use concepts learned in class, (OR the code is close to correct (but fails some tests) AND you have commited an excellent report in your "email" text file in which you precisely diagnose and describe a precise fix to each problem.)		**0 Pts** **No marks** You code (1) does not exist.	70 pts
Code Style	**30 Pts** **Excellent** Your code (1) has the right proportion of comments and place line and block comments correctly, (2) follow correct indentation every new indentation level, (3) has good variable naming, (4) has clear organization between tabs and is easy to read.	**15 Pts** **Good** Your code (1) has useful comments and place line and block comments correctly, (2) follow indentation, (3) has good variable naming.	**5 Pts** **Fair** Your code (1) has comments, (2) has variables, (4) has clear organization.	**0 Pts** **No marks** You code (1) does not exist.	30 pts
					Total points: 100