CSE105 Coursework 3: S1 2017

IMPORTANT

We will give you a Netbeans project folder configured correctly for this project. You must use this project for Coursework 3. We will give you this project folder this week. You can start working on the basic application before that, but you must transfer your code into the project when it is available.

The Problem

Community Support Groups
Your city has a number of Community Support Groups (5), which bring people together to help each other in the local community. Each group can have up to 500 volunteers. We will call these Groups 1, 2, 3, 4, 5.

Volunteers
Each volunteer has three skills from a possible 5. The skills are represented by five letters. Each letter is a String. The letters are A B C D E. Each volunteer has three skills. Duplicates are allowed: AAA, BBC, CDE are all possible skillsets. In order to be a Volunteer, your class must implement the Volunteer interface (this will be done for you in the project folder).

Skill Sorter
Write a Class to automatically allocate a volunteer to a group according to two criteria:

1. The number of each skill within a single group should be fairly well balanced.
2. The total number of all skills in each group will be fairly well balanced.

See examples at end of document.

The details of this algorithm are up to you to implement. It is not expected to be 100% effective; if you **look at the marking criteria below** you will see that you can pass this coursework with no working algorithm at all. A good attempt at balancing between groups will be awarded with high marks, and even a 'minimally effective' algorithm could get 70% if everything else is very good. We will explore this more in the week 12 lecture.

Volunteer Management Program
Your program should also have an interface to add further functionality: Volunteers can also be moved between groups.

Your program should display and update the people in each group. This can be done in the console window or in a Frame (window) using the techniques you have learned this semester.

The program should save information to a text file on shutdown, and read it back at startup, re-creating the same group allocation.

Menu Functions:

- Add volunteer
- Move volunteer
- Delete volunteer
- Delete all volunteers
- Display groups

The names, arguments and return types will be defined in the given project files, and must not be altered.

Task Requirements

Your program should have the following functionality:

- Add a volunteer – a menu interface that will receive the volunteer's skills, and automatically use a skill sorter to choose a group, and will assign them to one of the 5 groups

- Move a volunteer – a menu interface to choose 1 volunteer from a group and move them from one group to a different group

- Delete volunteer – a menu interface that will chose a volunteer and delete them from the system

- Delete all volunteers – clear all groups

- Save and exit – Exit the program and save all group data to file

- Display groups – You may display all volunteers in their groups in whatever format you prefer

- Load Data – When the program is started, it should search for an appropriate file and load all data into Community Groups

- Skill Sorter – After a volunteer has been input, the skill sorter should assign the volunteer to an appropriate group so that the balance of skills between all groups is as equal as possible.

Submission Instructions (required to achieve 40%+)

The submitted file is a valid NetBeans project folder, compressed into a .zip archive only (**not** RAR), and the project will compile and run in Netbeans with no further configuration and debugging.
The submission instructions are followed exactly.

Marking bands and descriptors

	Core Program Interface	*Sorting algorithm and display*
80%	Well designed* and highly effective	Well designed* and effective
70%	Good usability	Present and effective
60%	All functions work as specified	Present and minimally effective
50%	Most functions work as specified	Present and working, maybe ineffective
40%	The project compiles. The basic menu functions work.	The group display and sorting algorithm may be absent or not functional

*Notes: 'Well-designed' in this assessment means as far as possible adhering to the design norms and principles we have taught this semester, such as (but not limited to):

- Breaking functionality into manageable units through appropriate use of classes and methods
- Using meaningful and consistent names for variables, methods and classes
- Efficient algorithms
- Giving a good user experience (where relevant)
- Good exception handling and validation

5 marks may be deducted for each of the following:

- Runtime errors
- The use of *continue*.
- Identifier naming not conforming to commonly accepted Java programming conventions.
- Insufficient comments in the Java code.

Submission

You must submit your work on the CSE105 ICE page before 8.00pm, Sunday week 13.

You must submit 3 files:

- A Word document describing how your sorting algorithm works. This can be in pseudocode or a flow diagram, a list, or plain English (or a combination of the above. This document must NOT be in a ZIP archive. The file name must start with your student number.
- One **.txt** file for **the SkillSorter class** file. This documents must NOT be in a ZIP archive.
- One ZIP archive of your Netbeans project folder. Make sure this contains ALL the resources your application needs to run. If your program does not compile and run from this file only, you cannot pass (See Core Task Requirements). The ZIP file name must start with your student number.

This assignment is individual work. Plagiarism (e.g. copying materials from other sources without proper acknowledgement) is a serious academic offence. Plagiarism will not be tolerated and will be dealt with in accordance with the University Code of Practice on Assessment. Individual students may be invited to explain parts of their code in person, and if they fail to demonstrate an understanding of the code, no credit will be given for that part of the code.

Classes & Interfaces
Please note: you will be given a Netbeans Project with all of these interfaces etc already created and implemented for you. You do not have to do this yourself.
You should have classes for

- CommunityGroup. **Override toString to print out the total number of each skill contained in that CommunityGroup.**
- VolunteerImpl implements Volunteer
    - public ArrayList<String> getSkills();
    - public int getValue();
- SkillSorterImpl implements SkillSorter
    - public void receiveVolunteer (Volunteer);
    - public Object[] getCommunityGroups();

In addition to this, you will need to create, in the same project:

- A class for the input and interface.
- Any other classes you need.

Examples:
Well-balanced groups

Group 1	Group 2	Group 3	Group 4	Group 5
Skill A: 9 Skill B: 11 Skill C: 8 Skill D: 12 Skill E: 12	Skill A: 13 Skill B: 7 Skill C: 8 Skill D: 8 Skill E: 9	Skill A: 13 Skill B: 10 Skill C: 10 Skill D: 7 Skill E: 9	Skill A: 10 Skill B: 12 Skill C: 11 Skill D: 10 Skill E: 13	Skill A: 9 Skill B: 10 Skill C: 7 Skill D: 12 Skill E: 10
52	45	49	56	48

Poorly balanced 1: Totals balance well, but skills within groups completely unbalanced

Skill A: 50 Skill B: 0 Skill C: 0 Skill D: 0 Skill E: 0	Skill A: 0 Skill B: 50 Skill C: 0 Skill D: 0 Skill E: 0	Skill A: 0 Skill B: 0 Skill C: 50 Skill D: 0 Skill E: 0	Skill A: 0 Skill B: 0 Skill C: 0 Skill D: 50 Skill E: 0	Skill A: 0 Skill B: 0 Skill C: 0 Skill D: 0 Skill E: 50
Total: 50	Total: 50	Total: 50	Total: 50	Total: 50

Poorly balanced 2: Skills within groups well balanced, totals unbalanced

Skill A: 5 Skill B: 4 Skill C: 6 Skill D: 5 Skill E: 3	Skill A: 10 Skill B: 12 Skill C: 11 Skill D: 9 Skill E: 10	Skill A: 15 Skill B: 14 Skill C: 16 Skill D: 13 Skill E: 17	Skill A: 0 Skill B: 2 Skill C: 1 Skill D: 3 Skill E: 0	Skill A: 20 Skill B: 18 Skill C: 15 Skill D: 22 Skill E: 20
Total: 23	Total: 52	Total: 75	Total: 5	Total: 95