Donation Management (DM) System

Design Draft Due: 6:00PM, Wednesday (in Week 9), 01 May 2019
Prototype Demo: 6:00PM, Wednesday (in Week 10), 08 May 2019
Completed Assignment Due: 6:00PM, Wednesday (in Week 12), 22 May 2019

GENERAL DESCRIPTION

This **INDIVIDUAL** software development assignment consists of THREE PARTS, and is worth 22% of the assessment of this course. In this individual assignment, you will develop Java software for electronic donator records management.

1. **PART ONE** (2 Marks): submit a design draft of the assignment in week 9. Your tutor will provide you feedback and inputs to further improve your software design and development. Your design draft should describe:
   - all the classes needed and their responsibilities,
   - the fields and methods of each class,
   - the relationship of the classes.
2. **PART TWO** (2 Marks): demonstrate a prototype of your software using lab computers. At this stage, you must demonstrate that:
   - your code can be compiled via command line,
   - your programs are able to read from and write to files,
   - your programs could "update" and "delete" records,
   - your software is able to manipulate simple inputs, and to produce appropriate outputs accordingly.
3. **PART THREE (18 Marks) is the completion of the software development assignment.**

SOFTWARE DESCRIPTION

In this individual assignment, you will design and develop a Java software package to process the donator information records in different suburbs according to the given instructions/commands.

Your Java software **MUST**

- Read and validate the contents of TWO input text files (i.e. **record file** and **instruction file**), and
- Process the donator information records in the **record file** according to the instructions in the **instruction file**. The instructions should be performed sequentially (i.e. one after the other).
- When your program ends, your software saves the resulting records list (after being processed) to TWO output files (that are the **result file** containing the processed donator records list and the **report file** containing the query results).

DESCRIPTION AND FORMAT OF INPUT DONATOR RECORDS

When your DM system starts up, it will firstly read in **valid** donator records in the input records file; and then process the valid donator records according to the instructions in the instructions file.

1. The input donator record file contains zero or multiple donator records in the predefined format:
   - **Separators:** blank line(s) are used to separate the records.
   - **Invalid fields:** fields with error(s). Invalid fields should be ignored and not read into the system.
   - **Invalid records:** records missing any compulsory field or with invalid compulsory field. Invalid records will be discarded and will not be read and processed by your DM system.

2. An input donator record consists of seven possible fields to specify necessary information of a donator and his/her recipients. There are two compulsory fields, i.e., name, birthday; and the rest five fields are optional.
    1) *name* (compulsory): donator's name, in the form of a string of forename(s) and surname, all along one line. A name cannot include numeric and punctuation characters.
    2) *birthday* (compulsory): donator's birthday, in the format of "dd-mm-yyyy". Example: 11-07-2014.
    3) *address*: is a string that may span more than one line and should end with the abbreviation of a state in Australia (for example *NSW, VIC*).
    4) *postcode*: in the form of 4 numeric digits.
    5) *phone*: in the form of a sequence of **8** numeric digits. Leading zeros **cannot be ignored**.
    6) *recipient:* types of recipient, for instance, *the disabled*, *animal protection organization*, *environment protection organization*, etc. Different types of recipients helped by the same donator are spaned in the same line while separated by comma(s).
    7) *donation:* amount of money donated to a corresponding recipient. There is a one-to-one relationship between the recipient and the donation.
3. Format
    - Compulsory fields are required for all records when they are read in from the input records file. That is, you may have a donator record with *name and birthday*, or another record with all seven fields.
    - Each field begins on a new line and starts with the field name.
    - Single/Multiple white spaces are used to separate field name and the field value/content.
    - Fields may occur in *any order*.
    - If the number of recipient and donation is mismatched, you may discard the unmatched recipient(s) or donation(s) at the end of the field.
    - Example: "donatorsample1.txt" is a sample donator record file, with records as below:

```
name          Josephine Esmerelda Bloggs
birthday      13-05-1980
phone         99887766
recipient     the disabled, animal protection org
donation      100, 200
address       102 Smith St, Summer hill, NSW
postcode      2130


name          Pac Man
birthday      27-03-1992
address       27 New Street, Elwood,
              VIC
Postcode      3184
phone         00333976
```

DESCRIPTION AND FORMAT OF INPUT INSTRUCTIONS

Your software needs to read in and parse the **valid** instructions in instructions file and process the valid records with valid instructions.

1. The instructions file contains zero or multiple instructions:
    - There are four possible instructions: "update", "donate", "delete", and "query".
    - Each instruction occurs on a new line.
    - Each instruction begins with one of the instructions followed by a list of parameters.
2. **Description of instructions**

- **Update** a person's basic information (not include recipient and donation field)
  - For instance, the instruction
    update *name Jo Bloggs; birthday 08-07-1998; phone 88884444; postcode 2008; address 9001 Chester Crescent, Chatswood, NSW;*
    is supposed to add a record to your list for a donator with name *"Jo Bloggs"*, birthday *08-07-1998*, phone number *88884444*, postcode *2008*, address "*9001 Chester Crescent, Chatswood, NSW*". **Note** the use of a semicolon (";") to separate the fields in the instruction.
  - Your system will check whether this is an existing record:
    - if the donator's name and birthday are identical to those of an existing record, the existed record will be **updated** by the new information. E.g., update or merge the fields with the given values;
    - otherwise a new valid record needs to be automatically created and **added** to your list by your system.

- **Donate** money to a corresponding recipient(s) using the donator's *name* and *birthday*
  - For instance, the instruction
    donate *Sam Brown; 11-09-1990; the disabled, 200; animal protection org, 300;*

    indicates donation of $*200* and $*300* to recipients *the disabled* and *animal protection org* respectively by "*Sam Brown*" (birthday *11-09-1990)*. The amount of donation for each recipient should be added if the recipient(s) was/were already existed in the donor's record, otherwise recipient and donation fields should be initialized with the given information. Ignore the instruction if there is no such donator in your system.

- **Delete** a donator from your system using the donator's *name* and *birthday*
  - For instance, the instruction
    delete *Jeff Vader; 29-04-1987*
    indicates: deleting the record with name "*Jeff Vader*" and birthday *29-04-1987*.

- **Query** the statistics using the following instructions and save the query results to ***reportFile***. Query results from different queries should be separated using dash lines (---).

  (1) Query the records by name and save all the query results to the report file. The format of the query instruction is listed as below:
  - ***query name*** *David Joans*

  Note: Sort the donators in ascending order by their birthdays if there are multiple donators that have the same name.

  (2) **Query** top n donators and save query results to **reportFile**.
  - ***query top*** *n*

  indicates: query top *n* donators by total amount of donation and save the query results to **reportFile**. If there are multiple donators that have the same amount of donations, sort the donators in ascending order by their names. For example:

```
------query top 3------------
Jeff Vader; 29-04-1987; 5000
Jo Bloggs; 08-07-1998; 5000
Sam Brown; 11-09-1990; 2000
--------------------------
```

Note: you are required to implement a sorting method instead of invoking a Java *sort* API.

(3) Query the statistics of **recipients** and save the query results to ***reportFile***. The format is:

- ***query recipients***

indicates: for each type of recipients in the system, query the suburb(s) where the largest amount of donation (from related donators) was received. For instance, if there are three types of recipients in the system e.g. `the disabled`, `animal protection organization`, `environment protection organization`, the results as below should be appended to ***reportFile***:

```
------query recipients---------------------------------
The disabled: 2000; postcode 2050
Animal protection organization: 3000; postcode 2018, 1058
Environment protection organization: 1000; postcode 2000
-----------------------------------------------------
```

The above result represents that `the disabled` collected the largest amount of money ($2000) from all relevant donators in the suburb 2050 and …

FORMAT OF OUTPUT FILES

1. Your software needs to save the resulting data collection to the files including results file and report file.
2. The output files should have all the necessary resulting records.
3. Each field should fit on ONE line only.
4. Records should be separated by blank lines.
5. The format of output records should be consistent, e.g. fields are listed in same order and date format are consistent.
6. Report File: the results of different queries should be separated by a dash line with query instruction as shown in previous section.

SOFTWARE EXCUTION

Your software must be executed using command line in the following format:

java DM19S1. DM recordFile instructionFile resultFile reportFile

- DM19S1 is the package containing your assignment software.
- DM is the name for the main class.
- Two input files specified by command line arguments:
  - recordfile –containing the list of donator records
  - instructionfile –containing the set of instructions/operations to be carried out by your software
- Two input files specified by command line arguments:
  - resultfile – saving the processed donator records
  - reportfile – saving the results from query commands
- DO NOT implement a graphical user interface (GUI). DO NOT hard code the path/name of the input/output files as they will change between runs of the program. Some example invocations using the Command Prompt:

```
java DM19S1.DM u:\records\rec01.txt u:\ins01.txt w:\out\res01.txt
W:\report01.txt

java DM19S1.DM c:\rec03.txt d:\ins07.txt result.txt report.txt
```

IMPORTANT NOTES

1. Your code **must** make use of at least one collection e.g. ArrayList.
2. Your system must be able to handle both normal cases and difficult cases.
3. You **MUST NOT** build a graphical user interface.
4. You need to do systematic testing of the classes you create.
5. Your program must run on the computers in your lab classes, and must be demonstrated during your laboratory class to your tutor in week 10 and in week 12. **Being absent from the demos will incur a penalty of 4 marks.** If you miss the labs in weeks 10 and/or 12 because of serious illness or misadventure, you should request ***Special Consideration*** using the appropriate forms within a week.
6. Recall that **the University takes plagiarism very seriously**.

MARKING SCHEME

There are 4 parameters on which your submission will be marked:

1) **Design** [4 marks]: 2 marks at PART 1 (in week 9), and 2 marks at the final submission.
2) **Prototype Demo** [2 marks]: in week 10
3) **Implementation and testing** [14 marks]: at the final submission.
4) **Standard of coding** [2 marks]: at the final submission.

SUBMISSION

1. **PART 1 – Design Draft**
   During your lab class in Week 9 you must hand in a design document to your tutor.
   - An assignment cover sheet with declaration and your signature, and
   - Documentation of your software design, e.g. UML diagrams.
2. **PART 2 – Prototype Demonstration**
   You will demonstrate your program to your tutor in Week 10. Your program should have the basic functionalities at this stage, and should be able to handle normal cases (i.e. cases where all field names/values are correct). You will get the mark as long as your program runs smoothly by being invoked from the command prompt using the sample testing files released in Week 9. This part will assess your progress and provide you with more feedback.
3. **PART 3 – Complete Assignment**
   - Submit before 6:00PM, Wednesday, 22 May 2019 (Week 12).
   - Late submissions will incur a penalty of **2 marks per day**.
   - Program submission: You **MUST** upload a single ***.zip*** archive onto Canvas (COMP9103 Assignment Submission) before the deadline. The .zip archive must be named with your login ID, e.g. if your login name is "abcd1234" then the archive must be called "abcd1234.zip". It should contain:
     - Your program (all the java source code) in correct architecture
   - Turnitin submission (Canvas):
     - You must ALSO submit the documentation of your final software design, e.g. UML diagram and description of your design, in PDF through Turnitin.