CSC8001
Assessed Coursework 2 (week 9, 10 and 11): Interactive system

Aims:

- To gain experience at designing an interactive system of practical importance.
- To reinforce your knowledge about the standard `java.util.ArrayList<E>` class.
- To gain experience at using `java.lang.Comparable<E>` interface, inheritance and generic classes in Java.

Background:

Sometimes we want to keep the items in a list in sorted order, which we can achieve with a sorted list. The fundamental difference between a sorted list and an unsorted one is the "adding an item"/insertion method. Having a definition of a class for lists, we can obtain a sorted list class by altering the existing or adding new insertion method.

Specification:

Task 1:

Derive a `SortedArrayList<E>` class from the `java.util.ArrayList<E>` class in such a way that the items of a sorted array list are sorted in ascending order. This subclass of `ArrayList<E>` class will be needed to complete Task 2 (see Additional assumptions (2) below). For simplicity, you can provide only your new insertion method in the `SortedArrayList<E>` class. It is not a part of this project to consider all the mutator methods from `ArrayList<E>` class and see whether/how they should be overridden to make sure that a sorted list preserves its order when they are invoked on the list.

Task 2:

You are asked to write a program to help a librarian in their daily work. Your program should read a list of books and a list of users from a file. The content of the input file should have the following form: The first line contains an integer representing the number of books in the library, and is followed by the information about the books (two lines for every book: one line containing the title and the second one containing the author's name). The next line contains an integer representing the number of library users, followed by the information about the users (one line for every user with their first name and surname). Example file content is given below:

```
5
Concurrent Programming
C. R. Snow
Concurrent Programming
Stephen J. Hartley
Java Gently
Judith Bishop
Petri Nets
Wolfgang Reisig
Finite Transition Systems
Andre Arnold
4
Anna Smith
Zoe Brown
```

First name THEN Surname!!! find the last space!!!!!!!!

```
John Williams
John Smith
```

The program should be able to store the information about books and users:

1. For each book, the information required is: the title, author's name, whether it is on loan or not, and if it is on loan, who borrowed it. We assume that every book has only one author and that there are no two books in the library with the same title and author.
2. For each user, the library should know the first name, surname and the number of books currently held by the user. We assume that at any time a user can hold at most 3 books. We also assume that no two users share both the first name and the surname.

After the initial information has been read from the file, the librarian will be working with the program interactively. The program should display a menu on the screen offering a choice of possible operations, each represented by a lower case letter:

- f - to finish running the program.
- b - to display on the screen the information about all the books in the library.
- u - to display on the screen the information about all the users.
- i - to update the stored data when a book is issued to a user.
- r - to update the stored data when a user returns a book to the library.

You should implement all the above operations.

Additional assumptions:

1. When f is selected, the program stops running and all the data are lost. The program could be extended to save all the data to a file, but this is not a part of the project!
2. To store books and users you should use `SortedArrayList<E>` class. Books should be sorted in the ascending order of surnames of authors. You can assume that each author has only one surname and that it is always given after the first name(s) and/or initial(s). If there are several books by authors with the same surname, their order in the sorted list is not important. Users should be sorted in the ascending order of their surnames. If two users have the same surname then the first names should decide their order. You can assume that each user has exactly one first name and exactly one surname.
3. When a book is to be issued to a user, it must be checked whether the user is a valid user, and that the book is on the list of the books in the library. If not, an appropriate message should be displayed on the screen. If the request is a valid one, the program should check whether the book is on loan or not. If the book is currently on loan, a note to the user who is holding the book should be printed to a file informing that the book was requested by another user and should be returned as soon as possible. If the book is available, the stored information should be updated accordingly.
4. When a book is returned by a user, it must be checked whether the user is a valid user, and that the book is on the list of the books in the library and has been borrowed by this user. If not, an appropriate message should be displayed on the screen. If the request is a valid one, the stored information should be updated accordingly.

Submission Notes:

You should submit your Java files through NESS as well as two text (.txt) files showing that you have tested your program. One text file should be the output file used by your program, and the second one should contain the record of librarian's interactions with the program displayed in the terminal window. To produce the text file containing the content of the terminal window in BlueJ you can use "Save to file…" from the "Options" menu in the Terminal Window. If your input file contained different data than the example input data provided in this specification, then you also need to submit your input file (as a text file).

Deadline: The deadline for this coursework is **16:00 on Friday, 13th December 2013**.