CO320 Assignment 2 (Version 2019.11.05)

Introduction

This assignment is designed to allow you to practice writing class definitions that make use of collections and loops. In particular, you will be using the list-based collections `ArrayList` and `LinkedList`. You will hand the assignment in via the CO320 Moodle page.

Download the starting project from:

https://www.cs.kent.ac.uk/~djb/co320/LOCAL-ONLY/assign2.zip

Date set: 21st October 2019
Deadline: 23:55 18th November 2019
Weighting: 20% of the module's coursework mark.

Please try to read through the assignment as soon as possible so that you know what you need to understand to complete it, and can think about how best to organise your time.

Plagiarism and Duplication of Material

The work you submit must be your own. We will run checks on all submitted work in an effort to identify possible plagiarism, and take disciplinary action against anyone found to have committed plagiarism.

Some guidelines on avoiding plagiarism:

- One of the most common reasons for programming plagiarism is leaving work until the last minute. Avoid this by making sure that you know what you have to do (that is not necessarily the same as how to do it) as soon as an assessment is set. Then decide what you will need to do in order to complete the assignment. This will typically involve doing some background reading and programming practice. If in doubt about what is required, ask a member of the course team.

- Another common reason is working too closely with one or more other students on the course. *Do not* program together with someone else, by which I mean do not work together at a single PC, or side by side, typing in more or less the same code. By all means *discuss* parts of an assignment, but do not thereby end up submitting the same code.

- It is not acceptable to submit code that differs only in the comments and variable names, for instance. It is very easy for us to detect when this has been done and we will check for it.

- **Never** let someone else have a copy of your code, no matter how desperate they are. Always advise someone in this position to seek help from their class supervisor or lecturer. Otherwise they will never properly learn for themselves.

- It is not acceptable to post assignments on sites such as RentACoder and we treat such actions as evidence of attempted plagiarism, regardless of whether or not work is payed for.

Further advice on plagiarism and collaboration is available from https://www.cs.kent.ac.uk/teaching/student/assessment/plagiarism.local

You are reminded of the rules about plagiarism that can be found in the Stage I Handbook. These rules apply to programming assignments. We reserve the right to apply checks to programs submitted for assignment in order to guard against plagiarism and to use programs submitted to test and refine our plagiarism detection methods both during the course and in the future.

The Task

Imagine that you are writing code to manage a database of student-loan data. You have been provided with the `StudentLoan` class as a starting point and your task is to write the `LoanCompany` class:

- The `StudentLoan` has one field of type `String` for a student's ID and one field of type `int` called `amount` recording the outstanding amount of that student's loan. Further details are given below. You can use the 'test fixture to object bench' menu option of the `LoanTest` class to put a small sample of `StudentLoan` objects onto the object bench for help with testing your code.

- The `LoanCompany` class remains to be written. It must be able to store a list of `StudentLoan` objects and provide a range of functionality on the list. Full details are given below.

To complete this task you will need to be familiar with collection classes and iteration, both of which are covered in chapter 4 of the course text. In particular, you will need to make use of the `ArrayList` class and the `LinkedList` class, both of which are available in the `java.util` package. You will also need to understand how to write *for-each* and *while loops*. You will find it instructive to study the source code of the `music-organizer` projects, which are explored in chapter 4 and are used in the practical classes.

You will also find it useful to look at the Java API documentation, which can be found here:

https://docs.oracle.com/javase/8/docs/api/

In particular, select the link to `java.util` in the right-hand pane, and then scroll down to the link to `ArrayList<E>` to read about the methods of the `ArrayList` class.

The LoanCompanyTest and LoanTest classes

Two additional classes are included with the `StudentLoan` and `LoanCompany` classes. These are called `LoanTest` and `LoanCompanyTest`. They are not part of your assessment and you do not have to add any further code to them. Rather, they are placeholders that can later be used to help you test the accurate implementation of the `LoanCompany` class.

Towards the end of assessment period, an extended version of `LoanCompanyTest` will be made available to help you test your implementation of `LoanCompany`. However, as noted above, you can already use the ‘test fixture to object bench’ menu option of the `LoanCompanyTest` class to put a sample of `StudentLoan` objects onto the object bench to help with testing as you develop your program.

The StudentLoan class

The provided class is fairly simple and should not require any change. It consists of:

- A constructor that takes one parameter of type String and one of type int. These values are used to initialise the two fields and they represent the ID of the student and the amount of their loan. Note that the value for the loan is checked to ensure that it is not negative. If the value is negative, then the constructor prevents the object being created by ‘throwing an exception’. You do not need to understand the details of what this means at this point in the course, but you can read more about exception throwing in Chapter 14 of the course book if you wish to.

- An accessor for each field.

- A method called `payOff` that takes a single integer parameter representing an amount of the loan to be paid off. The method checks to make sure that the amount is sensible and within the bounds of the outstanding amount.

- A method called `getDetails` that returns a formatted representation of the student’s ID and amount owing. Note that this method returns the string rather than printing it.

Requirements

Your task is to write `LoanCompany` as described below. The work will be marked out of 100 and provisional marks are indicated against each element of the assessment. Pay careful attention to the exact details of the requirements; such as the names of methods and the parameters required.

Commenting and style [10 marks]

All the code you write must be well laid out and the fields, methods and constructors must be properly javadoc documented, using the style that is used in this course. If in doubt, a style guide is available:

https://www.bluej.org/objects-first/styleguide.html.

Adherence to these is included in the marking scheme.

The LoanCompany class – basic methods [30 marks]

The class must use either an `ArrayList` or `LinkedList` to store any number of `StudentLoan` objects. In the class write:

- `addLoan`: a method taking a `StudentLoan` object as a parameter and storing it the company's list of loans.

- `getNumberOfLoans`: a method that returns (not prints) the number of `StudentLoan` objects in the list.

- `repay`: a method that takes two integer parameters and returns a `boolean` result. The parameters are an index and an amount, in that order. If the index is valid for the list, this method must call the `payOff` method of the `StudentLoan` at the given index, passing the given amount as its parameter. It must then return true. If the index is not valid the method must do nothing to the list or any object in it, and return `false`.

- `removeLoan`: a method that takes an integer index as its parameter, removes the `StudentLoan` at that index in the list and returns the removed object as the method's result. If the index is not valid the value `null` must be returned instead. No error message must be printed if the index is not valid.

The LoanCompany class – iterative methods [30 marks]

- `list`: a method that prints details of all the `StudentLoan` objects in the list, one per line. The printed list must **not** include the index of items or any other information.

For instance:

```
ABC123 owes 5000
XYZ123 owes 200
```

Use the `getDetails` method of `StudentLoan` for the formatting.

- `showOutstanding`: a method that prints details of only those students who have a non-zero amount in their loan.

The LoanCompany class – challenge methods [30 marks]

Important note: This element is considerably harder than the others and might require you to seek additional information to complete it. You should consider carefully whether the time it could take you to complete it is justified given the percentage of marks available for it.

- Write a method called `find` that takes a single String parameter representing a student ID. The method must find the first `StudentLoan` in the list whose ID **exactly and completely** matches the parameter, and return the matching `StudentLoan` as the method's result. If there is no match then the method must return `null` and not print anything.

  NB: An exact match means that the full ID of a student must match the method's parameter. For instance, a parameter "ZZZ" matches an ID of "ZZZ" but a parameter "Z" does not.

- Write a method called `removeClearedLoans` that removes all `StudentLoan` objects from the list that have a zero loan amount left. This method must return a `LinkedList` of the removed `StudentLoan` objects. If there are no matching objects in the list then an empty list must be returned – not a `null` value.

  Make sure that you test your method with a list containing a mixture of cleared and uncleared loans.

Starting the assignment

This section contains some advice on how to get started with the assignment.

Take a copy of the assignment project:

- http://www.cs.kent.ac.uk/~djb/co320/LOCAL-ONLY/assign2.zip

Unzip the file in your own working area. The project already contains an outline of the `LoanCompany` class, as well as the `StudentLoan` class.

The LoanCompany class

Open the source you have been given of the `LoanCompany` class to see what your starting point is. It should look something like the following:

```
/**
 * A class to store a list of StudentLoan objects.
 *
 * @author (your name)
 * @version (a version number or a date)
 */
public class LoanCompany
{
    /**
     * Constructor for objects of class LoanCompany
     */
    public LoanCompany()
    {
    }

    ...
}
```

Place your own name and the date in the appropriate places of the comment area.

In the first assignment, we encouraged you to keep checking whether your code still compiles after each change. So, make sure that you have everything correct at this starting point before going any further. We won't keep repeating this advice from now on.

Add a field

Add a field to `LoanCompany` for storing the list of `StudentLoan` objects. You will need to use an import statement at the top of the file. (See the `MusicOrganizer` class in chapter 4 if you are uncertain how to do this.)

Constructor

Complete the constructor for the class. The constructor must take no parameters. The constructor must initialise the list field.

Methods

The order of the methods listed in the requirements section is a reasonable guide to the order in which you should try to implement them. For instance, start with a method to add to the list, then a method that tells you how many objects are in the list. After adding each method, compile the code, fix any errors and test that it works. Use inspectors and the debugger to give yourself confidence that the code is working as you think it should.

Thorough testing will be very important in ensuring that your class is working correctly. You may find it tedious to keep creating the `StudentLoan` objects needed to do this, but it is essential and using the ‘test fixture’ feature of BlueJ will make this easier – you can make your fixtures as complex as you wish.

Getting help

If you need help with the assignment, there are various places you can go. You will not lose marks if you ask for help.

- You can ask questions on the module’s anonymous Q&A page and it is ok to post code there because I will edit it before publishing.
- You can ask for assistance from your class supervisor.
- You can ask the module’s lecturer.
- You can turn up at the homework club: https://www.cs.kent.ac.uk/teaching/student/support/homework/

Be very careful about asking for help from other students on the course, or students who have taken the course before, other than as described above. If in doubt about what is appropriate, check the plagiarism guidelines above and/or ask the module’s lecturer.

Finally

Before you submit your assessment, thoroughly test the whole class to make sure that nothing you added later has broken anything added earlier. If you are unable to complete the class – even if you cannot get it to compile – still submit what you have done because it is likely that you will get at least some credit for it.

David Barnes, 21st October 2019
Updated 5th November 2019 to clarify the nature of an exact match for the `find` method.