COURSEWORK ASSIGNMENT **UNIVERSITY OF EAST ANGLIA**
School of Computing Sciences

UNIT :	CMP-4008Y Programming 1
ASSIGNMENT TITLE :	Module Assessment Exercises 1

DATE SET	**:** 06/11/15	
DATE OF SUBMISSION	**:** 03/12/15	
RETURN DATE	**:** 11/01/16	
ASSIGNMENT VALUE	**:** 20%	
SET BY	**:** Geoff McKeown	**SIGNED:**
CHECKED BY	**:** Anthony Bagnall	**SIGNED:**

Aim:

Subject specific

Testing of basic principles of Java programming. Implementation of some small Java applications.

Transferable skills

Computer programming; software testing.

Learning outcomes:

Subject specific

Development of basic competence in object oriented/algorithmic programming in Java, familiarity with the associated concepts and terminology.

Assessment criteria:

Part marks are shown on the exercise sheet.

Description of the Assignment

Written work

1. Which of the following are not valid Java identifiers, and why?

   (a) `wolVes`

   (b) `United(there_is_only_one)`

   (c) `_87`

   (d) `5_3`

   (e) `Real_ale`

   (f) `isFound?by`

   [5 marks]

2. A class `Television` has the following fields:

```
        private TelevisionManufacturer supplier;
        private String code
        private int screenSize; // in inches
        private String type; // e.g.  plasma screen
```

   Assume that the class `TelevisionManufacturer` is available and that this class contains an `equals` method.

   (a) Define a class variable, `totalTVs`, whose purpose is to keep track of the total number of `Television` objects constructed during execution of a program that uses the `Television` class.

   (b) Declare a default constructor for this class.

   (c) Declare a constructor for this class which has a formal parameter corresponding to each field.

   (d) Declare an accessor method called `getScreenSize` whose purpose is to return the value of the `screenSize` field of `this Television`.

   (e) Declare a mutator method that sets the type of `this Television` to a given value.

   (f) Declare a method to determine whether or not `this Television` has been supplied by a given manufacturer.

   [15 marks]

Lab work

1. Consider the following algorithm.

```
(1) Randomly select a 4-digit positive number excluding
    1111, 2222, ..., 9999, with leading zeros  if
    necessary
    // thus 158 would be represented by 0158 and
    // 9 would be represented by 0009
(2) Let bigger denote the number obtained by taking
    the four digits obtained in step (1) in decreasing
     order
(3) Let smaller denote the number obtained by taking
    the four digits obtained in step (1) in increasing
    order
(4) Obtain a new 4-digit number by subtracting smaller
    from bigger, inserting leading zeros if necessary
    // e.g. 7666 - 6667 = 999, so the 4-digit value
    is 0999
(5) Repeat from step (2) for a maximum of 10 iterations
    or until two successive 4-digit numbers  are
    identical, whichever occurs first.
```

   Implement and test this algorithm in Java, using the `Random` class to generate test values.

   [30 marks]

2. **[This exercise is worth a total of 50% of the marks]**

   This exercise is concerned with simulating the activity of a library with respect to its book-stock. You are required to write a Java application which contains a class called `LibraryBook` together with a `main` method class called `LibrarySimulation`.

   The data about each `LibraryBook` consists of:

   - author(s) (e.g. Lewis and Loftus)
   - title (e.g. Java Software Solutions)
   - number of pages (e.g. 832)
   - library classification (e.g QA98)
   - number of times borrowed (e.g 53)
   - current status
   - number of pending reservations (may not exceed 3)

   Once it has been initially set, the current status of a `LibraryBook` is one of the following:

   `REFERENCE_ONLY, ON_LOAN, AVAILABLE_FOR_LENDING`

If the status of a `LibraryBook` is initially set to `REFERENCE_ONLY`, its status will not change throughout the simulation.

If a `LibraryBook` is currently on-loan, a reservation may be placed on that `LibraryBook` but only up to a maximum of three pending reservations.

(a) Define an `enum` type in the `LibraryBook` class corresponding to the status list given above.

[2 marks]

(b) Define the fields for the `LibraryBook` class corresponding to the data items given above.

[3 marks]

(c) Define a class (`static`) variable to keep track of the total number of `LibraryBook`s currently on loan.

[2 marks]

(d) Define a single constructor with the following header:

```
/**
 * Constructor with arguments for a LibraryBook's author(s),
 * title and number of pages
 * @param bookAuthor    the names of the author(s) of this
 *                      LibraryBook
 * @param bookTitle     the title of this LibraryBook
 * @param bookPages     the number of pages of this
 *                      LibraryBook
 */
 public LibraryBook(String bookAuthor,
                      String bookTitle, int bookPages)
```

The instance variable for any field for which there is no corresponding formal parameter should be initialised to 0 (for primitive variables) and to `null` (for reference variables).

[3 marks]

(e) Define all of the following `public` methods:

- methods for "getting" the author(s), the title, the number of pages, the library classification and the number of times borrowed for a

  `LibraryBook`;

[4 marks]

- a method with the following header for "setting" the library classification of a

  `LibraryBook`:

```
/**
 * A  method to reset the Library classification of this
 * LibraryBook
 * @param bookClass     the proposed new classification
 * @return              true,  if the proposed new
 *                             classification has at
 *                             least 3 characters to which
 *                             the Library classification is
 *                             reset.
 *                      false, otherwise.
 */
public boolean setClassification(String bookClass)
```

[3 marks]

- a method, `setAsReferenceOnly`, to designate a `LibraryBook` which has not already been designated as either reference-only or as available for lending, as reference-only;

[2 marks]

- a method, `setAsForLending`, to designate a `LibraryBook` which has not already been designated as either reference-only or as available for lending, as available for lending; [2 marks]

- a `boolean` method `isAvailable` to determine whether or not `this LibraryBook` is available for lending;

[2 marks]

- a method with the following header for reserving `this LibraryBook`, if possible:

```
/**
 * If possible, reserves this LibraryBook.
 * This is only possible if this LibraryBook is currently on loan
 * and less than 3 reservations have been placed since this went
 * on loan.
 * @return      true,  if a new reservation has been made for this.
 *              false, otherwise
 */
public boolean reserveBook()
```

[3 marks]

- a method, `borrowBook`, to simulate `this LibraryBook` being issued out to a borrower;

[2 marks]

- a method, `returnBook`, to simulate the return of `this LibraryBook` by a borrower;

[4 marks]

- a `toString` method that returns a description of a `LibraryBook` such that the output obtained when the statement `System.out.println(courseText);` is executed looks something like:

```
Title: Java Software Solutions
Author: Lewis and Loftus
Pages: 832
Classification: QA99
```

[3 marks]

Include the following static method in your `LibrarySimulation` class:

```
/**
 * A method to generate a collection of LibraryBook objects to use as
 * test data in your simulation
 * @return      an array of LibraryBook objects
 *
 */
 public static LibraryBook [] generateBookStock(){
    String [] authorsList =
        { "Lewis and Loftus",  "Mitrani", "Goodrich",
          "Lippman", "Gross", "Baase", "Maclane", "Dahlquist",
          "Stimson", "Knuth", "Hahn", "Cormen and Leiserson",
          "Menzes", "Garey and Johnson"};
    String [] titlesList =
          { "Java Software Solutions", "Simulation",
            "Data Structures", "C++ Primer", "Graph Theory",
            "Computer Algorithms", "Algebra", "Numerical Methods",
            "Cryptography","Semi-Numerical Algorithms",
            "Essential MATLAB", "Introduction to Algorithms",
            "Handbook of Applied Cryptography",
            "Computers and Intractability"};
    int [] pagesList = {832, 185, 695, 614, 586, 685, 590, 573, 475,
                    685, 301, 1175, 820, 338};

    int n = authorsList.length;

    LibraryBook [] bookStock = new LibraryBook[n];

    for(int i = 0; i < n; i++){
           bookStock[i] = new LibraryBook(authorsList[i],
                                titlesList[i], pagesList[i]);
       }

    // set library classification for half of the LibraryBooks
    for(int i = 0; i < n; i=i+2){
        bookStock[i].setClassification("QA" + (99 - i));
    }
```

```
        // set approx. two thirds of LIbraryBooks in test data as
        // lending books
        for(int i = 0; i < 2*n/3; i++)
            bookStock[i].setAsForLending();

        // set approx. one third of LibraryBooks in test data as
        // reference-only
        for(int i = 2*n/3; i < n; i++)
            bookStock[i].setAsReferenceOnly();

        return bookStock;
    }
```

(f) Define a third static method in your `LibrarySimulation` class, with the following header:

```
/**
* @param bookStock          the stock of LibraryBooks in the library
* @param numberOFevents     the size of the events table to be
* @return                   generated table of events generated during
*                           the simulation
*/
 public static String [] runSimulation(LibraryBook [] bookStock,
                                                int numberOFevents,)
```

This static method is to implement an algorithm for simulating the activity of the library with respect to its book-stock. The algorithm generates a sequence of random events. Each successive event is determined by the current status of a randomly selected `LibraryBook`. At each stage in this process, each `LibraryBook` is equally-likely to be selected. There are six types of event:

- the selected library book is currently unclassified and is now given its proper classification (BOOK IS CLASSIFIED);
- the selected library book is already classified but is reference-only and is therefore not available for lending (REFERENCE ONLY BOOK);
- the selected library book is classified and available for lending and is now loaned out (BOOK IS LOANED OUT);
- the selected library book is currently on-loan and is being returned to the Library (BOOK IS RETURNED);
- the selected library book is currently on-loan and a reservation is placed on that book (RESERVATION PLACED FOR ON-LOAN BOOK);
- the selected library book is currently on-loan but further reservations are not allowed on that book (BOOK IS ON-LOAN BUT CANNOT BE RESERVED).

A maximum of three reservations may be placed on an on-loan `LibraryBook`. When a `LibraryBook` that has been on-loan is returned, if it has at least one reservation placed on it, then the `LibraryBook` remains on-loan, but with the number of reservations placed on it reduced by 1.

As can be seen from the above list of possible events, when the simulation leads to the selection of an on-loan `LibraryBook`, there are two possible situations: either the `LibraryBook` is being returned or an attempt is being made to place a reservation on the `LibraryBook`. Your algorithm should randomly choose between these situations (hint: generate a random integer in the range $\{0, 1\}$ with 0 corresponding to a return and 1 to an attempted reservation).

The output from the `runSimulation` method is an array, each element of which is an event summary and has the form:

```
<event number> <number of books currently on loan before
current event is processed> <library book classification>:
<type of event>
```

A ***pseudo-code*** high-level description of the algorithm that `runSimulation` should implement is given below:

```
let n = number of library books in library book-stock;
// assume we have library book 0 to library book (n-1)
initialise events list;
for each event number from 1 to the required number of events do
  select a library book at random from the book-stock;
     // i.e.  generate a random number from 0 ..  (n-1)
  Use the status of the selected library book to determine
     the type of event and process that event;
  add the event summary to the events list
endfor
```

[15 marks]

The print-out of the events list from a typical run of your program might look like:

```
run:
0 0 --- BOOK IS CLASSIFIED
1 0 QA97 BOOK IS LOANED OUT
2 1 QA95 BOOK IS LOANED OUT
3 2 QA99 BOOK IS LOANED OUT
4 3 QA93 REFERENCE ONLY BOOK
5 3 QA95 BOOK IS RETURNED
6 2 QA95 BOOK IS LOANED OUT
7 3 QA97 BOOK IS RETURNED
8 2 --- BOOK IS CLASSIFIED
9 2 QA99 BOOK IS RETURNED
10 1 QA98 BOOK IS LOANED OUT
11 2 QA95 BOOK IS RETURNED
12 1 QA93 REFERENCE ONLY BOOK
13 1 QA98 BOOK IS RETURNED
14 0 QA93 REFERENCE ONLY BOOK
15 0 QA98 BOOK IS LOANED OUT
.       .       .
```

$\vdots \qquad \vdots \qquad \vdots$

[5 marks]

Assessment

Marks will be awarded as follows

20% Written work

30% Lab work exercise 1

50% Lab work exercise 2

Submission instructions overleaf.

Submission Procedure

To the hub: A paper submission of your solutions to ALL questions.

via blackboard: a zipped Netbeans project with your code for the lab work questions.

Written coursework should be submitted by following the standard CMP practice. Students are advised to refer to the Guidelines and Hints on Written Work in CMP (`https://intranet.uea.ac.uk/computing/Links/Reports?_ga=1.7481330.1383599.1413214592`).