Beijing-Dublin International College

SEMESTER 1 FINAL EXAMINATION - (2018/2019)

School of Computer Science

COMP2005J Object Oriented Programming Programming Exam

Prof. Pádraig Cunningham
Dr. Seán Russell*

Time Allowed: 180 minutes

Instructions for Candidates:

Answer all questions.

BJUT Student ID:__________________ **UCD Student ID:**__________________

I have read and clearly understand the Examination Rules of both Beijing University of Technology and University College Dublin. I am aware of the Punishment for Violating the Rules of Beijing University of Technology and/or University College Dublin. I hereby promise to abide by the relevant rules and regulations by not giving or receiving any help during the exam. If caught violating the rules, I accept the punishment thereof.

Honesty Pledge:____ **(Signature)**

Instructions for Invigilators

Students are permitted to bring any printed notes and textbooks into the examination.

Information

- The name of your project in eclipse should be your UCD student number
- All questions are based on the same program. Each question adds some new functionality to the code. You should not move to the next question until you have finished the one before it.
- You should only submit working code. If your code does not compile or work correctly you should comment it out. You will **fail** if you submit code that does not compile.
- To improve your grade you should submit well designed code with light comments and correct indentation.
- The main method of the program should be in a class named `Main`.
- There are bonus marks for providing good JUnit test cases for the classes you have written. This is not required to get 100%, but it will improve your score.
- There are files with some input data for the program on moodle. They are in a zip file called InputFiles.zip. Download these at the beginning of the exam. Internet access will be turned off after 10 minutes. If you have not downloaded the input files you will have to create them yourself. These files should be put in the project folder in eclipse, not in the src folder with your code. The password for the zip file is `"study"` (without the quotes).
- The classes you define should have good encapsulation. If you need to access an instance variable you should add a getter or setter method.

Rules

- You are not allowed bring any device (phone, laptop or tablet) into the exam with you. Even having them is against the exam rules and will result in you failing the module.
- You should only open the programs listed below on your computer, any other programs open will be considered **CHEATING**.
  - Eclipse
  - The Java API in a browser
  - Moodle to download the input files of to submit your work at the end
- Any student caught trying to read the code of another student will also be considered **CHEATING** and will fail the exam

Question Topic

The questions in this exam will all be based around the workings of a Library service. Users can borrow books from the library for free, but when returned late they may be charged a fee. All questions will be associated with this concept.

Question 1: Users

Users of the library will all have an account detailing their name and the amount of fees that they owe. Each user has been given a unique id (an integer number) to represent them, that will be used to calculate the fess that they owe at the end of the year.

a. Implement a `User` class in Java to represent a single user. The class should have a constructor and any required getter methods.

(5%)

b. Add a methods to the `User` class called `addFee` and `payFee`. addFee should increase the fees owed by the user by the amount specified in the parameter (int) and payFee should reduce the fees owed by the user by the amount specified in the parameter.

(5%)

c. Override the `toString` method in the `User` class, when called this method should return a String containing the users name, followed by their id in brackets and the amount of fees that they currently owe. For example it should look like this `"Sean Russell has id 1 and owes 5.0 in fees"`.

(5%)

d. Write a **static/class** method called `readUserData` in the class `Main`. This method takes no parameters and should read all of the user data from the file `"users.txt"`. Each line of the file represents a single user and the data is in the following format `<id> <firstName> <familyName>`.

After reading the data from the file a User object should be created for each line and added to a suitable data structure. The user objects will be needed in the final questions so the data structure should be returned when the file is fully read.

The `readUserData` method should be called in the main method and the returned data structure remembered in a variable. Each user object should be printed to the screen in the main method.

The output of the code should look like this, but the order may depend on the data structure that you use:

```
Keyla Sandoval has id 0 and owes 0.0 in fees
Ivan Boyer has id 1 and owes 0.0 in fees
Aria Howard has id 2 and owes 0.0 in fees
Kristina Bonilla has id 3 and owes 0.0 in fees
...
```

(10%)

(Question Total 25%)

Question 2: Books

For each book that can be borrowed, it is given a unique id (unique only for books), a title and an author.

a. Implement a `Book` class in Java to represent a single book. The class should have a constructor and any required getter methods.

(5%)

b. Override the `toString` method for the `User` class, when called this method should return a String containing the books title, followed by the author and the id. For example it should look like this `"ULYSSES has id 0 and was written by James Joyce"`.

(5%)

c. Write a **static/class** method called `readBookData` in the class `Main`. This method takes no parameters and should read all of the book data from the file `"books.txt"`. The details of each book are contained on three lines of the file. The first line contains an integer id representing the book, the second line contains the title of the book and the third line contains the name of the author.

After reading the data from the file a Book object should be created for book in the file and added to a suitable data structure. The book objects will be needed in the final questions so the data structure should be returned when the file is fully read.

The `readBookData` method should be called in the main method and the returned data structure remembered in a variable. Each book object should be printed to the screen in the main method.

The output of the code should look like this, but the order may depend on the data structure that you use:

```
ULYSSES has id 0 and was written by James Joyce
THE GREAT GATSBY has id 1 and was written by F. Scott Fitzgerald
A PORTRAIT OF THE ARTIST AS A YOUNG MAN has id 2 and was written by
    James Joyce
LOLITA has id 3 and was written by Vladimir Nabokov
BRAVE NEW WORLD has id 4 and was written by Aldous Huxley
```

(10%)

(Question Total 20%)

Question 3: Rental

Users can borrow books from the library. When they have borrowed them the library will need to keep track of the user that borrowed the book, the book that was borrowed and the date it should be returned. All dates in the system will be represented as integers counting from 0 to 365.

a. Implement a `Rental` class in Java to link the above information together. The class should also have a constructor and getter methods for the user and book objects and the due date.

(5%)

(Question Total 5%)

Question 4: Library

The books and users are all managed by the library. The library will process a list of transactions that represent books being borrowed and returned as well as fees being paid. Before starting this question, you should read all of the questions and think about what data structures would be best for remembering the users and books. Choosing well will make the code much easier to complete. Within the class the values representing the number of days books can be borrowed for (7 days) and the fee for each late day should be remembered (2 per day).

a. Implement a `Library` class in Java to represent the library. This class should have data structures for remembering users and books as well as the current books out on loan (rental objects). The data structures should be constructed in the constructor of the class.

(5%)

b. Add the methods `addUser` and `addBook` to the library class. These methods should take a single parameter of a Book or a User object and add that to the data structure to be remembered. The method does not need to return anything.

(5%)

c. Add the methods `getUser` and `getBook` to the library class. These methods should take a single integer as a parameter and return the user or book object that has that id. You can assume that the id will always match one of the objects that have been added.

(5%)

d. Add the method `borrowBook` to the library class. This method should take three integers as parameters. The first parameter is the id of a user, the second parameter is the id of a book they have borrowed and the third is the day number they have borrowed it on. The user, book and return date should be recorded in a rental object and remembered in the library class. The return date will need to be calculated. A message should then be printed to the screen that says `"XXXX borrowed the book YYYY and must return it by ZZ", where XXXX is the name of the user, YYYY is the title of the book and ZZ is the date the book should be returned by.`

(5%)

e. Add the method `returnBook` to the library class. This method should take three integers as parameters. The first parameter is the id of a user, the second parameter is the id of a book they have returned and the third is the day number they returned it on. The rental objects should be searched to find the expected return date of the book and if late, fees should be charged to the user. A message should then be printed to the screen that says `"XXXX returned YYYY ZZ days late and and was charged a fee of WW"` or `"XXXX returned YYYY on time and was not charged a fee"`, where XXXX is the name of the user, YYYY is the title of the book and ZZ is the number of days late the book was returned and WW the fee charged to the users account. Once completed, the rental object in question should be removed from the data structure of the library.

(10%)

f. Add the method `processTransaction` to the library class. This method should take a string and three integers as parameters. The first parameter is the type of transaction `"Rental"`, `"Return"` or `"Payment"`. The second parameter is an int representing the user that performed the transaction and the third is an int representing the date that the transaction was carried out on and the final parameter is the book that was used in the transaction (if any) or the amount if the transaction is the type pay. The details of the transaction should be recorded in the user objects in the library. If rental of a book, the `borrowBook` method should be called, if return of a book the `returnBook` method should be called. If the transaction is the payment of a fee, then them message `"XXXX paid fee of YY and has a remaining balance of ZZ"` where XXXX is the name of the user YY the amount they paid and ZZ the amount of money they still owe.

(10%)

g. Add code to the main method to create a Library object. All of the user and book objects created in question 1 and 2 should be added to the library. You should then add code to read the file `"transactions.txt"`, this contains a set of book rental transactions. Each line in the file contains a transaction type and three numbers, the transaction type is a string and the three integers are the user id, the date and then either the book id or amount (depending on the type of transaction). Each transaction read from the file should be passed to library object as parameters of the `processTransaction` method.

   The output of the main method should look something like this:

```
Nicholas Gonzalez borrowed THE HEART OF THE MATTER and must return it
    by 14
Lois Campbell borrowed THE GRAPES OF WRATH and must return it by 14
Johnny Baker borrowed THE STRANGE CAREER OF JIM CROW and must return it
    by 14
William Butler borrowed CATCH-22 and must return it by 15
...
```

(10%)

(Question Total 50%)

Submission

Create a single **zip** file containing your entire project.

How to Submit

Your entire project should be submitted as a zip file. To do this, please follow these instructions:

a. Right-click on your project and choose Export

b. In the menu that appears choose `General -> Archive File` and click next

c. Make sure that only your project is selected with a tick (this means all files)

d. Click on browse to choose the location of the file will be saved and its name

e. Choose the name of the zip file as your UCD student number

f. Click Finish

g. After you have completed this open the zip file to make sure that all of your code is in it. There will be no second chances to submit your work.