Concordia University
COMP 249 – Winter 2024
Object-Oriented Programming II - Assignment # 1
Due 11:59 PM – Friday, February 23rd, 2024

Purpose: The purpose of this assignment is to help you review some of the main topics covered so far, including: classes, loops, arrays, arrays of objects, static attributes, static methods, inheritance, and polymorphism.

IMPORTANT NOTE: IN THIS ASSIGNMENT, YOU ARE NOT PERMITTED TO USE ANY OF THE JAVA BUILT-IN CLASSES SUCH AS ArrayLists, LinkedList, HashMaps, etc. Using any of these will result in zero marks! In other words, you need to code whatever is needed by yourself!!! Of course, you can use the classes String and Scanner.

General Guidelines When Writing Programs:

- Include the following comments at the top of your source codes
  ```
  // -----------------------------------------------------------------------
  // Assignment (include number)
  // Question: (include question/part number, if applicable)
  // Written by: (include your name and student id)
  // -----------------------------------------------------------------------
  ```
- In a comment, give a general explanation of what your program does. As the programming questions get more complex, the explanations will get lengthier.
- Include comments in your program describing the main steps in your program.
- Display a welcome message which includes your name(s).
- Display clear prompts for users when you are expecting the user to enter data from the keyboard.
- All output should be displayed with clear messages and in an easy-to-read format.
- End your program with a closing message so that the user knows that the program has terminated.

Description

FunReadings Library is asking you to write a Java program that manages all the items in their library and their leases as well as returns to and from clients. Your program is to be used only by library employees, and not clients. The library has books, journals, and media (for example: DVDs). All items

have an ID, a name, author(s), and year of publication. A client has an ID, name, phone number, and email address. There might be many copies from each item in the library. However, different copies will have different IDs. In general, an item's ID is unique.

A journal also has a "volume number", a book has "number of pages", while a media has a "type" (audio/video/interactive). Books IDs start with "B" (e.g. B1, B2, …), journals IDs start with "J" (e.g., J1, J2…), Media IDs start with "M" (e.g., M1, M2…). A good menu should be provided so that the user of your program can perform the following operations:

- add an item, delete an item, change information of an item, list all items in a specific category (book, journal, or media), and print all items (from all categories).
- add a client, edit a client, and delete a client.
- lease an item to a client and return an item from a client.
- show all items leased by a client.
- show all leased items (by all clients).
- Display the biggest book (see below).
- Make a copy of the books array (see below).

Part I: UML

Draw a single UML representation for the hierarchy of the above-mentioned classes. Your representation must also be accurate in terms of UML representation of the different entities and the relation between them. You must use software to draw your UML diagrams (no handwriting/hand drawing is allowed). In case additional classes and/or data attributes are needed to reflect how the above classes are related in real-life, you can add these classes.

Part II: Implementation

1. Implement the above-mentioned classes using inheritance and according to the specifications/requirements stated below:

- You must have 3 different Java packages for the classes: the first package will include the

driver class, the second package will include the class for `client`, and the other classes will reside in the third package.

- For each of the classes, you must have at least three constructors, a default constructor, a parameterized constructor (which will accept enough parameters to initialize ALL the attributes of the created object from this class) and a copy constructor. For instance, if any of these classes has 7 attributes (including the ID), then the parameterized constructor must accept 6 parameters to initialize all its 6 attributes – the ID is set automatically. The copy constructor creates a new object that is an exact copy of the passed object, with the exception of the ID.
- An object creation using the default constructor must trigger the default constructor of its ancestor classes, while creation using parameterized constructors must trigger the parameterized constructors of the ancestors.
- For each of the classes, you must include at least the following methods: accessors, mutators, **toStrin**g(), and **equals()** methods.
- The **toString()** method must return a clear description and information of each object
- The **equals()** method must first verify if the passed object (to compare to) is null and if it is of a different type than the calling object. The method would clearly return false if any of these conditions is true. Otherwise, it compares all the attributes EXCEPT the ID to see if the two objects are equal.
- For all classes, you **must** use the appropriate access rights, which allow most ease of use/access **without compromising "Information Hiding"**. Do not use most restrictive rights unless they make sense!
- When accessing attributes from a base class, you must take full advantage of the permitted rights. For instance, if you can directly access an attribute by name from a base class, then you must do so instead of calling a public method from that base class to access the attribute.

2. Write a driver program (that contains the **main()** method), which will utilize all of your classes, and trigger the execution of all the methods you have written. You should always keep in mind that if a code is written but never triggered and tested, then it may not work correctly when needed; so, test all your code!

3. Besides the **main()** method, the driver will also include two more methods:

    - A method called ***getBiggestBook()***, which, as the name suggests, will find the book with the highest number of pages. If many books have the same highest number of pages, the method can return anyone of them.
    - A method called ***copyBooks(),*** which will make a deep copy of the array of books passed as parameter.
    - **NOTE:** it is up to you to decide if these methods are static or not and their use depends on this decision.

In the **main()** method, you must ask the user if they want to get the menu (as requested in page 2 above) or if they want to run a predefined/hard-coded scenario. If the user selects the menu, your program will take them through the menu. If the user selects the pre-defined scenario, your program must:

- create at least 3 objects from each type of items and 3 users, and display their information (you must take advantage of the **toString()** method).
- test the equality of some of the created objects using the **equals()** method. You should test at least the equality of two objects from different classes, two objects from the same class with different values, and two objects of the same class with similar values. In other words, you should include enough test cases to test all the scenarios in your implementation.
- create an array for each one of the types of items. Create another array for all items. Note that these arrays might be partially used. That is, the number of effective elements in the array are less than the capacity of the array.

- call the ***getBiggestBook()*** with the array of books and display the result.
- Finally, call the method ***copyBooks()*** on the array of Media.

Submitting Assignment 1

- For this assignment, you are allowed to work individually or in a group of a maximum of 2 students (i.e. you and one other student). Groups of more than 2 students will lead to zero (0) mark for each group member!
- You can work with a student from any section.
- Submit only ONE version of an assignment. If more than one version is submitted; then the first one will be graded, and all others will be disregarded.
- Naming convention for zip file: Create one zip file, containing all the source files and produced documentations for your assignment, using the following naming convention:
    - The zip file should be called *a#_StudentName_StudentID*, where # is the number of the assignment and *StudentName/StudentID* is your name and ID number, respectively. Use your "official" name only - no abbreviations or nick names; capitalize the usual "last" name. For example: for the first assignment, student Mike Simon, with ID 12345678, would submit a zip file named like: *a1_Mike-SIMON_12345678.zip.*
    - If working in a group, only one of the group members can upload the assignment. Do **NOT** upload the file for each member! Add all IDs and names to the name of the file and inside the file.
- You must submit your assignment using Moodle under Assignment 1 Submission folder. Assignments uploaded to an incorrect submission folder will not be checked and will result in a mark of 0 as if no submission was made.
- **IMPORTANT**: make sure you submit way before the closing date to avoid "surprises". Please also note that we cannot submit it for you as the submission must come from your account.

We will not accept any submission by email for whatever reason. Excuses like "my system crashed when I was submitting and when it restarted it was past the deadline by 2 minutes", “I thought my friend will submit but they did not”, "I forgot to submit", “I suddenly slept and woke up 3 minutes after the closing time”, or similar will not be accepted and will go **unanswered**.

IMPORTANT (Please read very carefully): Additionally, which is very important, a demo will take place with the markers afterwards. Markers will inform you about the details of demo time and how to book a time slot for your demo. If working in a group, both members must be present during demo time. Different marks may be assigned to teammates based on this demo.

- If you fail to demo, a zero mark is assigned regardless of your submission.
- If you book a demo time, and do not show up, for whatever reason, you will be allowed to reschedule a second demo but a penalty of 50% will be applied.
- Failing to demo at the second appointment will result in zero marks and no more chances will be given under any conditions.

Evaluation Criteria for Assignment 1 (10 points)

UML	**1.5**
Correct use of packages	**1**
Correct implementation of classes (including: Constructors, equals, toString, static attributes/methods…)	**3**
getBiggestBook() and copyBooks()	**2**
Overall quality of code (completeness, names of class, names of methods, names of variables, indentations…)	**2.5**
Total	**10 points**