Session 2 - OOP

Exercise 1:

Your neighborhood is creating a new library and wants to create a digital system for storing, lending, and returning books. Your task is to model the library with books. Books can be borrowed and they can be returned. Each book has a title, an author, and a number of copies. The library keeps track of books. Customers should be able to print the list of all books in the library, and how many copies of them are still available.

1. Create a new package under your RemedialJava project, called session2.oopLibrary. Inside this package you can create the new classes you'll need for this exercise (Library and Book).

2. Create a program that models the Library and add 5 titles/books to it. Then simulate the use of the library (as if there are customers taking actions): print the possible books, borrow and return some of them (you can do this either using a loop or a sequence of commands). Make sure to print when there are requests to borrow a book (that may not be available) or attempts to return a book, as well as when books are indeed borrowed or returned. Periodically also print the contents of the library.

3. Run it in Eclipse (Run → Run As → Java Application).

Exercise 2:

Go back to your library program and apply better the encapsulation principle.

Exercise 3:

We will now simulate a restaurant, where clients are provided with an electronic menu, that they can use to place their order. The menu has 10 dishes. Dishes have a name, a price and calorie content. Orders consist of choosing 3 dishes (one as a starter, one as a main, and one as a desert), by giving their position on the menu. When an order is placed, we should see the bill cost and their food energy content.

Before coding each part, think (or use pen/paper) to decide your class architecture.

Part one
1. Create a package named session2.restaurant. Write a program for creating one menu, assign the menu 10 dishes, and try out different orders to see how much they cost and what energy content they have.
2. Try to ensure encapsulation, and verify variable values. Print relevant information at any step (e.g., dish creation, dish order, etc.).
3. As a first step consider placing orders using an index of the dish in the menu.

Part two
Consider now using the dish name to make an order (think about using method overloading).

Part three
How would you go about adding multiple menus in this program (to accommodate many clients)? Items across menus should be the same. Create a new package and classes to try this.

Part four
Just for a bit of practice. Imagine you want to reverse the order of dishes in the menu, how would you do it?