CSC240 – SPRING 2013
ASSIGNMENT #4

Exceptions

This final assignment involves making changes in the assignment 3 code so that we can get some experience using exceptions. In order to do this we will make some changes to the Set<T> methods we used in assignment #3.

The add, displayAnObject and editAnObject methods in the generic class Set<T> (displayAnObject and editAnObject were implemented as part of assignment #3) will be modified so that they throw (but do not catch!) exceptions as indicated below:

public void add(T anObject) throws DuplicateObjectException …

public void displayAnObject(T aT) throws CannotFindException

public void editAnObject(T editedT) throws CannotEditException

Your program will implement the three Exception classes cited in the above method headers. In other words, you must write the fairly simple code for each of these three new Exception classes: DuplicateObjectException, CannotFindException and CannotEditException. Let's now discuss how each of these three kinds of Exceptions will be thrown and handled in your assignment #4.

The idea here is that the Set<T> add, displayAnObject and editAnObject methods will throw the relevant exception if they cannot succeed in their tasks. Thus, if we call the Set<T> add method and the parameter, anObject, is already in the Set, then the add method will throw (and not handle) a DuplicateObjectException. The main application, which calls add when the user chooses menu choice 1 or 2, will catch the DuplicateObjectException, providing the user with feedback and returning the user to the main menu.

Note that add, displayAnObject and editAnObject are all now void methods!!! So, the application cannot check the return value from these to see if the method was successful. When add fails, it will be because a DuplicateObjectException is thrown. When displayAnObject fails, it will be because a CannotFindException has been thrown. When editAnObject fails, it will be because a CannotEditException has been thrown. All the application needs to do is to catch (handle) these three kind of exceptions which the Set<T> methods will throw but not catch.

The Wholly Nutritious Food Data Menu is unchanged from assignment #3. Here it is again:

Wholly Nutritious Food Data Menu
1 – Add a Frozen Food Product
2 – Add a Canned Food Product
3 – Display names of all Frozen Food Products
4 – Display names of all Canned Food Products
5 – Display data for a Frozen Food Product
6 – Display data for a Canned Food Product
7 – Edit data for a Frozen Food Product
8 – Edit data for a Canned Food Product
9 – Quit

From the user's perspective the program behavior will be almost identical to the behavior for assignment #3 except now it would be a good idea to give the user messages that refer to the kind of exception that was thrown and caught. For example, if the user enters the name of a Frozen Food product that is not in the Set the feedback should say:

CannotFindException thrown.
The Frozen Food product you specified is not in the Set.

Use a similar strategy to provide feedback for the other exceptions that are thrown.

After the user enters a menu choice, you should use a switch statement that will call a method that will handle the data processing for that menu choice. You will receive a pseudocode handout that lays out this strategy. The idea is that the switch construct will be in a try block followed by three catch clauses, one for each of the Exception classes that you will be implementing.

Exceptions can be thrown when the user chooses one of the following menu choices: 1, 2, 5, 6, 7 and 8. When an exception is thrown by the relevant method in the Set generic class, the exception will be caught after the switch (was will be shown in the pseudocode handout). The exception handlers will just provide the user with appropriate feedback. Then, the user will be asked to enter another menu choice. The exception handlers (i.e., the catch clauses) will be rather simple. They won't ask the user to enter new FrozenFood data or new CannedFood data, they'll just give the user feedback and return the user to the menu.

grading

Your program will be graded using a grading checklist that is available on D2L. Your program should satisfy the items in the grading checklist as much as possible. (Hopefully, 100%.) In addition, your program will be graded on:

- quality of the documentation
- clarity of your code
- the use of appropriate conventions of style
- correct use of classes and objects
- timely submission of your work