TP2

Exercise 1: Implementing the CustomStack Class

Design and implement the 'CustomStack' class which includes a constructor that accepts a parameter defining the capacity of the stack. This class should have the following methods:

- AddElement(element): a method that adds a new element to the stack

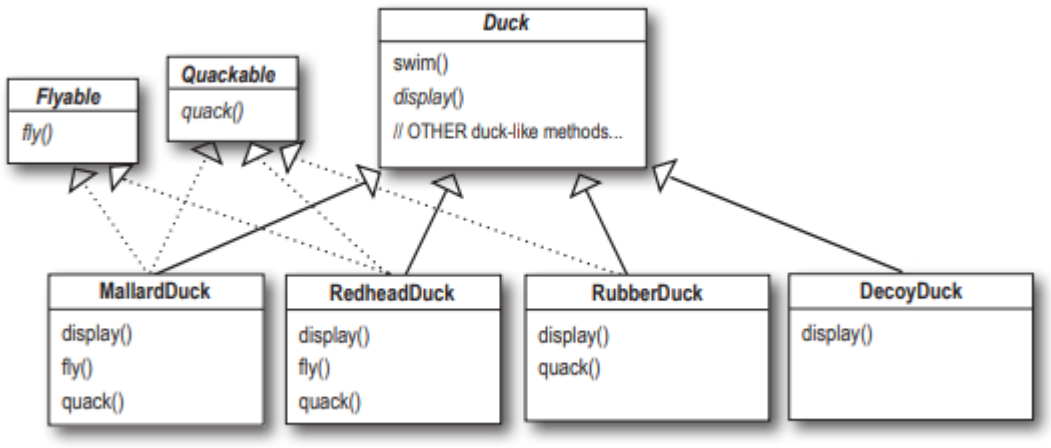
- RemoveElement(): a method that removes an element from the stack

- LastInStack(): a method that returns the most recent addition (top) of the stack

- StackIsEmpty(): a method that returns true if the stack is empty and false if otherwise

- StackIsFull(): a method that returns true if the stack is full and false if otherwise

Exercise 2: (Source : head first design patterns strategy pattern)



Develop the abstract 'Duck' class with the following inheriting subclasses:

- MallardDuck

- RedheadDuck

- RubberDuck

- DecoyDuck

Exercise 3: Library Management System

Your task is to design a library management system with the following requirements:

1. Develop a foundational 'Book' class with properties and methods such as `getTitle()`, `getAuthor()`, and `getYearOfPublication()`.

2. Create two subclasses of 'Book': 'Novel' and 'Textbook'. Each of these should have a `displayInformation()` method that presents details specific to each subclass.

3. Implement a 'Library' class that contains a list of 'Book' instances. This class should have methods to add books to the library inventory, remove books, and list all books currently in the library.

4. Define a 'LibraryUser' interface with methods such as `borrowBook()`, `returnBook()`, etc. Now create 'Student' and 'Teacher' classes in accordance with this interface.

5. Lastly, build a 'LibraryCard' class that should be composed within both 'Student' and 'Teacher' classes.

Exercise 4: Online Music Platform

Design an online music platform with the given specifications:

1. Begin by creating a base 'Song' class with properties and methods common to all songs, such as 'playSong()', 'getSongLength()', etc.

2. Next, implement an 'Album' class that incorporates a list of 'Song' objects. Ensure that this class includes methods to add songs to the album, remove songs, and list all songs in the album.

3. Construct an 'Artist' class which holds a list of 'Album' objects. This class should offer methods to add an album, delete an album, and list all albums associated with the artist.

4. Develop a 'User' interface that includes methods like 'listen()', 'addToPlaylist()', etc. Create classes 'FreeUser' and 'PremiumUser' in alignment with this interface.

5. Finally, design a 'Playlist' class containing a list of 'Song' objects which is associated with a specific user. The class should allow for the addition, removal, and shuffling of songs, and should be tied directly to a 'User' instance.