INSTRUCTION:

Design and implement a Java program for the following description of a *Library Item Management* system. You must apply `Command` Design Pattern.

A library item management system is an application to manage items in a library item database. Typical library items include books, CDs, videos and DVDs. These items are grouped into categories and a given item can belong to one or more categories. For example, a new movie video may belong to both the `Video` category and the `NewReleases` category.

An `Item` class represents a typical library item and a `Category` class represents a category of items. A `Category` object maintains a list of its current member items. Similarly, an `Item` object maintains the list of all categories which it is part of. (Hints: You can use a `HashMap` to store the current member items for a `Category` object, and also a `HashMap` to store the current categories of an `Item` object.)

For simplicity, the library item management system deals only with adding and deleting items. Applying the `Command` pattern, the action to be taken to process *add item* and *delete item* requests can be designed as implementers of a common `CommandInterface` interface. The `CommandInterface` provides an abstraction for the processing to be carried out in response to a typical library item management request such as add or delete item. The `CommandInterface` implementers, namely, `AddCommand` and `DeleteCommand` represent the *add* and the *delete* item request, respectively.

The `ItemManager` class acts as an *invoker* for the application. It invokes operations on the receiver objects (`Item` and `Category`). It

- Contains a `Command` object within
- Invokes the `Command` object's `execute` method as part of its `process` method implementation
- Provides a `setCommand` method to allow client objects to configure it with a `Command` object

The `CommandTest` client class uses the invoker `ItemManager` to get its *add item* and *delete item* requests processed. To keep the application simple, no database access logic is implemented. Both `Item` and `Category` objects are implemented to simply display a message.

1. Draw a UML class diagram to show your design for the *Library Item Management* system.

2. Implement your design in Java. Test your implementation and produce the output below.

```
Item 'Duet' has been added to the 'CD' Category
Item 'Duet' has been added to the 'New Releases' Category
Item 'Duet' belongs to these categories: [CD, New Releases]
Item 'Duet' has been deleted from the 'New Releases' Category
Item 'Duet' belongs to these categories: [CD]
```

3. Add a *move* functionality that allows an item to be moved from one category to another. Implement the *move* functionality as a combination of *delete* followed by an *add* operation. Both delete and add operations must be executed together as a unit to provide the move functionality. Test your implementation and produce the output below:

```
Item 'A Beautiful Mind' has been added to the 'CD' Category
Item 'A Beautiful Mind' belongs to these categories: [CD]
Move the item from the CD category to Book category
Item 'A Beautiful Mind' has been deleted from the 'CD' Category
Item 'A Beautiful Mind' has been added to the 'Book' Category
Item 'A Beautiful Mind' belongs to these categories: [Book]
```