Lab 09.2: Creating an Inventory System

Objective

This lab will introduce the second of the larger systems that you will be creating. This system is a set of classes that manages an inventory. You will define a class hierarchy using inheritance to manage an inventory for a CD/Book store.

Overview

In this lab you will:

- Create the necessary packages
- Create the necessary system classes
- Add the appropriate attributes
- Provide the appropriate method definitions
- Utilize inheritance

Step by Step Instructions

Description of the Inventory System

This system will model a store that sells books and CDs and needs to create a system to track its inventory.

All items in inventory need to track a title, price and quantity. This will be managed by a generic class named **Item**.

For books, we also need to know the author, publisher and category. This will be managed by a **Book** class that inherits from **Item**.

For pop music CDs we also need to know the artist and release date. This will be managed by a **CD** class that inherits from **Item**.

The artists on the pop music CDs are assumed to be bands with a band name and several members. The store wants to track the band name, each member's name, and the musical instrument each member plays. This will be managed by a standalone **Artist** class.

For classical CDs we also need to know the composer, performer(s), location of recording and release date. This will be managed by a **ClassicalCD** class that inherits from **Item**.

Create a Java Project and Package

1. Create a new Java Project within the same workspace named **Store**.

2. Create a package named **com.javaoo.store**.

Exercise 1: Creating the Inventory System Classes

3. Create each of the following classes in the **com.javaoo.store** package:
    a. **Item**
    b. **Artist**
    c. **Book** which is a sub-class of **Item**
    d. **CD** which is a sub-class of **Item**
    e. **ClassicalCD** which is a sub-class of **Item**

Exercise 2: Adding Attributes and Methods

4. For each of the inventory system classes, add the necessary attributes and methods as specified below.

5. **Item**:

    a. Declare the following **private** attributes.
        i. `title` which is a `String`
        ii. `price` which is a `double`
        iii. `quantity` which is an `int`
    b. Create getter and setter methods for each of the attributes. In most IDEs, this can be done quite easily by right-clicking in the editor and choosing **Source > Generate Getters and Setters**.

6. **Artist**:

    a. Declare the following **private** attributes.
        i. `name` which is a `String`
        ii. `memberNames[]` which is an array of 20 `Strings`
        iii. `memberInstruments[]` which is an array of 20 `Strings`
    b. For now, create a getter and setter method for the **name attribute only**

7. **Book**:

    a. Declare the following **private** attributes.
        i. `author` which is a `String`
        ii. `publisher` which is a `String`
        iii. `category` which is a `String`
    b. Create getter and setter methods for each of the attributes.

8. **CD**:

    a. Declare the following **private** attributes.
        i. `artist` which is an object of type `Artist`
        ii. `releaseDate` which is an object of type `Date`
    b. You will need to import the `java.util.Date` class
    c. Create getter and setter methods for each of the attributes.

9. **ClassicalCD**:

    a. Declare the following **private** attributes.
        i. `composer` which is a `String`
        ii. `performers[]` which is an array of 5 `Strings`
        iii. `recordingLocation` which is a `String`
        iv. `releaseDate` which is an object of type `Date`
    b. You will need to import the `java.util.Date` class
    c. Create getter and setter methods for each of the attributes **except for `performers[]`.**

10. At this point, all of your code should compile. We haven't created a `main()` method yet so we can't execute any of it but ensure that you do not have any compilation errors before continuing.

Exercise 3: Working with Array Attributes

11. For the `performers[]` attribute of the **ClassicalCD** class we are going to create a method called `addPerformer(String performer)`. This method will need to know how many elements in the array are already being used. The length attribute won't help us – it just tells us how many slots there are for holding data. So:
    a. Create a private attribute named `performerCount` which is of type `int` and initialize it to zero.
    b. Create the `addPerformer()` method.
    c. Within the body of `addPerformer()` test to make sure there is still room in the `performers[]` array. Use `performers.length` and `performerCount` to manage this test.
    d. If there is no room left in the array, display a message warning the user.
    e. If there are available slots in the array, add the performer name.
    f. Ensure that you increment `performerCount` !

12. We also want a method to **display** all of the performers for a **ClassicalCD**. Write a method called `showPerformers()` which loops through the

elements of the array and prints them to the console. USE THE `performerCount` attribute! Do not print out empty array slots.

Exercise 4: Creating Constructors and Testing

13. Create a new package named **com.javaoo.store.drivers** . Create a new class named **InventoryDriver** in the **com.javaoo.store.drivers** package. Ensure that this class has a `main()` method.

14. In the **InventoryDriver** main method create an array of 50 **Items** named `myInventory`.

```
Item[] myInventory = new Item[50];
```

15. In the **Item** class, create a constructor which accepts three parameters in its signature. These parameters should be `title`, `price` and `quantity`. In the constructor use the set methods to set up each of these attributes. Also, create a constructor that takes no arguments. This is needed in order to allow our other code to compile for now. We will address this in more detail later. When done, it should look as follows.

```
/**
 *
 *
 */
public Item() {

}

/**
 * @param title
 * @param price
 * @param quantity
 */
public Item(String title, double price, int quantity) {
    this.setTitle(title);
    this.setPrice(price);
    this.setQuantity(quantity);
}
```

16. In the **Book** class, create a constructor which accepts six parameters in its signature. These parameters should be `title`, `price`, `quantity`, `author`, `publisher` and `category`. Pass `title`, `price` and `quantity` to the super-class constructor. In the **Book** constructor use the set methods to set up the other three attributes.

```
/**
 * @param title
 * @param price
 * @param quantity
 */
public Book(String title, double price, int quantity,
        String author, String publisher, String category) {
    super(title, price, quantity);
    this.setAuthor(author);
    this.setPublisher(publisher);
    this.setCategory(category);
}
```

17. Back in **InventoryDriver** `main()` method, create five **Book** objects using the constructors you just created. Give the books any titles that you want. Here is an example:

```
//Add some books to Inventory
myInventory[0] = new Book("Godzilla on Holiday", 24.95,5,
        "Wesley Wynham-Price", "Ransom House","FICTION");
myInventory[1] = new Book("Loch Ness Memories", 49.95,1,
        "Fred MacMurray", "Penquin Press","FICTION");
myInventory[2] = new Book("MVS JCL",89.95,3,
        "Steve Balmer", "Microsoft Press","NON-FICTION");
myInventory[3] = new Book("Lingo in a Nutshell", 19.95,8,
        "Bill Bates", "O'Reilly","NON-FICTION");
myInventory[4] = new Book("Grid Computing", 79.95,2,
        "Bobby Beowold", "Trouser Press","NON-FICTION");
```

18. Using the same technique as above, create a constructor for the **Artist** class that only assigns the name attribute for now. Remember, **Artist** does NOT inherit form **Item**.

19. Create a constructor in the **CD** class similar to what you did in the **Book** class. This constructor should accept 5 parameters which are `title`, `price`, `quantity`, `artist` and `releaseDate`.

20. Go back to **InventoryDriver** and create a **CD** object. When creating a **CD**, you will need to pass the constructor an instance of **Artist** and a **Date**. Before creating the **CD**, create an instance of **Artist** for that particular CD**.** Also, we are using a deprecated technique for creating dates for now. Since we have not discussed exceptions yet, it would be too confusing to use the **DateFormat** class which is a more sophisticated way of managing dates. Here is an example of what you code might look like. Add 2 more **CDs**.

```
myInventory[5] = new CD("Going For The One", 12.95, 4,
                    new Artist("YES"), new Date("07/07/1977"));
```

21. We will finish up the **ClassicalCD** class and write a report to make sure everything is working correctly in the next lab. However, at this point you should not have any fatal compilation errors.