THỰC HÀNH LẬP TRÌNH HƯỚNG ĐỐI TƯỢNG

ANSWER LAB03

Họ và tên: Phạm Công Sơn

MSSV: 20205220

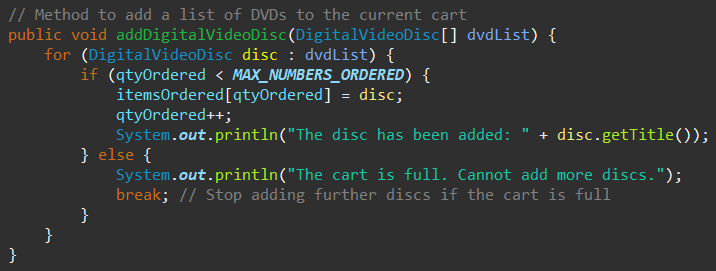
1. Branch your repository
2. Working with method overloading

2.1. Overloading by differing types of parameter

- You will create a new method that has the same name but with different type of parameter.

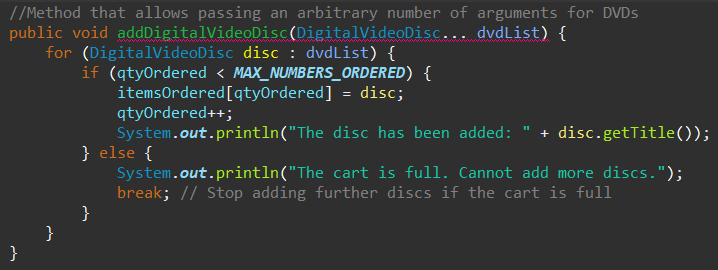
**addDigitalVideoDisc(DigitalVideoDisc [] dvdList)**

This method will add a list of DVDs to the current cart.



- Try to add a method **addDigitalVideoDisc** which allows to pass an arbitrary number of arguments for dvd. **Compare to an array parameter. What do you prefer in this case?**

+ Method addDigitalVideoDisc using varargs in Java to pass an arbitrary number of arguments for dvds:



*+ Comparing Varargs and Array parameter:*

1. *Syntax simplicity:*

* ***Varargs:*** *Simplifies the syntax for method calls. You can pass a comma-separated list of arguments, which is more intuitive and concise.*
* ***Array Parameter:*** *Requires explicitly creating an array before passing it to the method, which is less user-friendly.*

1. *Flexibility:*

* ***Varargs:*** *Supports both single arguments (addDigitalVideoDisc(dvd1)) and multiple arguments (addDigitalVideoDisc(dvd1, dvd2))*
* ***Array Parameter:*** *Only works when you explicitly pass an array, requiring an extra step to declare and populate the array.*

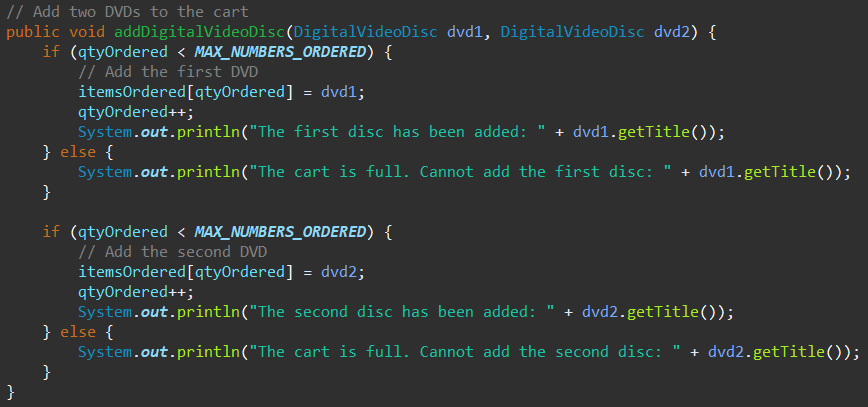
*+ I prefer Varargs to Array because it simplifies syntax for method calls and it supports both single arguments and multiple arguments.*

2.2. Overloading by differing the number of parameters

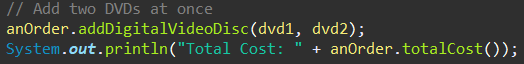
**- Create new method named addDigitalVideoDisc**

**+** The signature of this method has two parameters as following:

**addDigitalVideoDisc(DigitalVideoDisc dvd1,DigitalVideoDisc dvd2)**



* In Aims.java:





1. Passing parameter:

- Question: ***Is JAVA a Pass by Value or a Pass by Reference programming language?***

First of all, we recall what is meant by **pass by value** or **pass by reference**.

* Pass by value: The method parameter values are **copied** to another variable and then the copied object is passed to the method. That's why it's called pass by value
* Pass by reference: An alias or reference to the actual parameter is passed to the method. That's why it's called pass by reference.

*Java is a pass-by-value programming language.*

To test whether a programming language is passing by value or passing by reference, we usually use the **swap** method. This method aims to swap an object to another object.

* After the call of **swap(jungleDVD, cinderellaDVD)** why does the title of these two objects still remain?

*Inside the swap method:*

* *o1 and o2 are local variables that receive copies of the references to jungleDVD and cinderellaDVD.*
* *So swapping o1 and o2 only changes the local references in the swap method, not the original references (jungleDVD and cinderellaDVD) in the main method.*

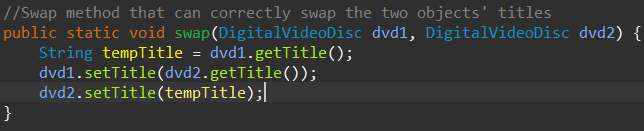
*Since the original references in the main method are unaffected, the titles of the objects referred to by jungleDVD and cinderellaDVD remain unchanged.*

* After the call of **changeTitle(jungleDVD, cinderellaDVD.getTitle())** why is the title of the JungleDVD changed?

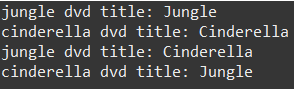
*The changeTitle method modifies the state of the jungleDVD object directly through its reference. Since the method accesses the actual object via its reference, any changes to the object's state are visible in the main method.*

**Please write a swap() method that can correctly swap the two objects.**

***-*** *Swap method to correctly swap two objects’ titles:*



*- Result in Console:*



1. Use debug run
2. Classifier Member and Instance Member
3. Open the Cart class

6.1. Print cart

Create a new method to print the list of ordered items of a cart, the price of each item, and the total price. Format the outline as below:

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*CART\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Ordered Items:

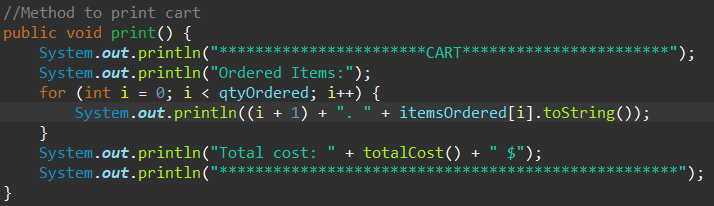
1. DVD - [Title] - [category] - [Director] - [Length]: [Price] $

2. DVD - [Title] - ...

Total cost: [total cost]

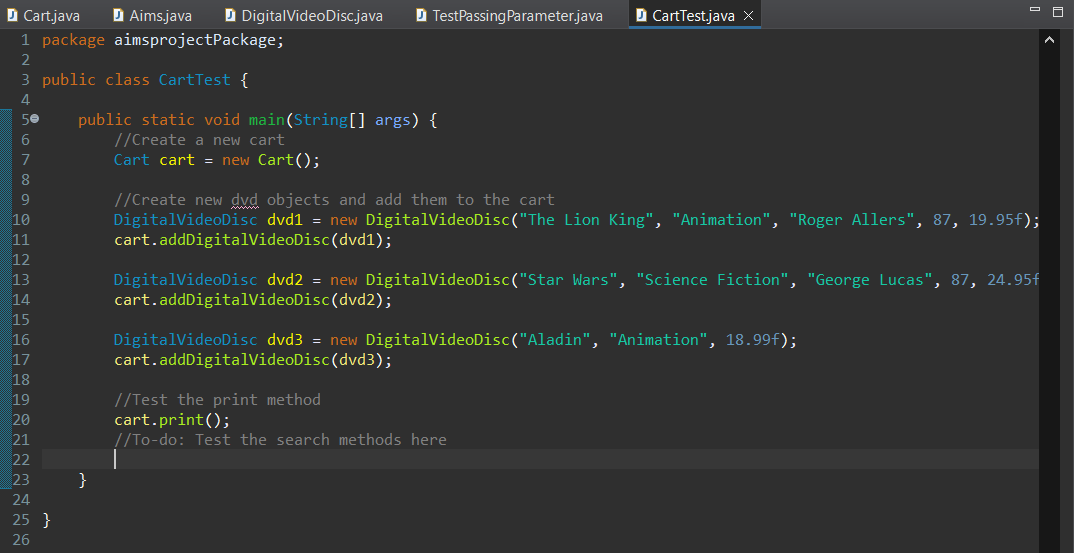
\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

*- Method to print cart:*

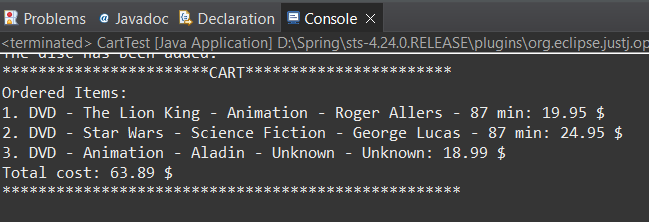


In the **CartTest** class, write codes to test all methods you have written in this exercise.

*- Created a new class named CartTest:*



*- Result in Console of print method:*

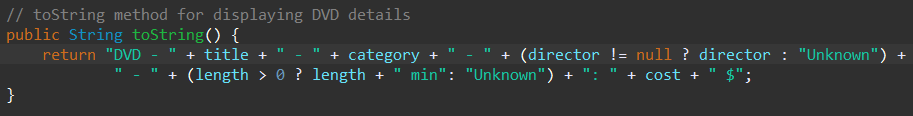
**

6.2. Search cart

**Suggestion**: Write a **toString()** method for the **DigitalVideoDisc** class. What should be the return type of this method?

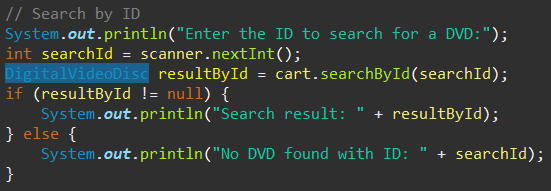
+ The return type of toString() method is String

+ Method toString():



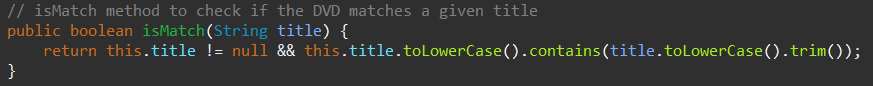
- Search for DVDs in the cart by ID and display the search results. Make sure to notify the user if no match is found.

+ Method searchById():

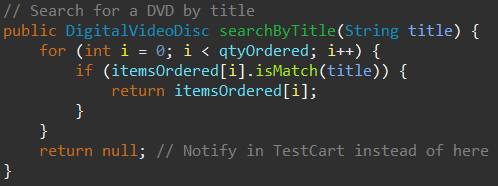


- Search for DVDs in the cart by title and print the results. Make sure to notify the user if no match is found. ***Refer to problem statement in Lab02 for the matching rule.* Suggestion**: write a **boolean isMatch(String title)** method in the **DigitalVideoDisc** which finds out if the corresponding disk is a match given the title.

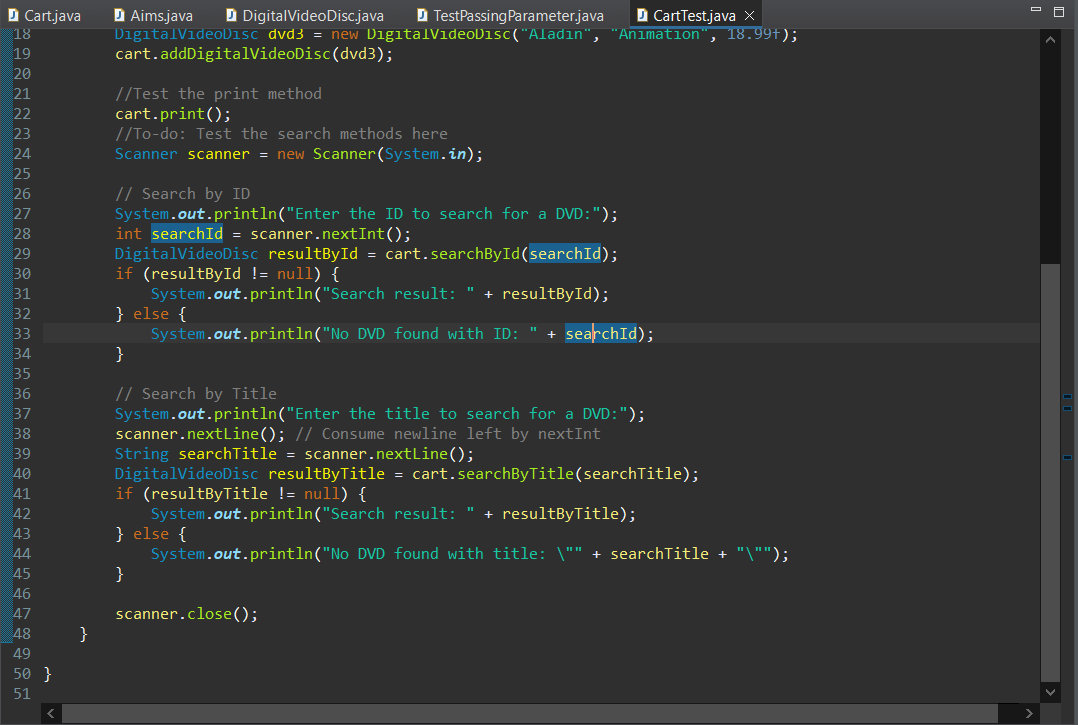
+ Method isMatch():



+ Method searchByTitle():



+ Class CartTest:



+ Result in Console:

