1. **Working with method overloading:**

* **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?**

The method which allows to pass an arbitrary number of arguments is called Variable Argument (Varargs) method. The varags allows the method to accept zero or muliple arguments. Before varargs either we use overloaded method or take an array as the method parameter but it was not considered good because it leads to the maintenance problem. If we don't know how many argument we will have to pass in the method, varargs is the better approach.

This varargs method is convenient than passing array parameter because we don’t have to create an array of DigitalVideoDisc in the main function but the method creates an array of parameters of unspecified length in the background and such a parameter can be treated as an array in runtime. Length of the array is limited while varargs is not and it easier to manage the object passed in the method. For example, if you don’t want to pass some elements in the array, you must delete them first before passing in the method but when using varargs, we can choose specified objects to be the argument.

1. **Passing parameter:**

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

Java passes all arguments to a method in form of pass-by-value: Passing value/copy of the real argument. When invoked, the method receives the value of the variable passed in. When the argument is of primitive type, pass-by-value means that the method cannot change its value. Modifying the internal state of object parameters inside a method (Modifying formal arguments) does not affect the real arguments.

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

When the argument is of reference type, pass-by-value means that the method cannot change the object reference, but can invoke the object's methods and modify the accessible variables within the object. Modifying the reference to an object can change the real arguments of the object.

1. **Open the Cart class**

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

getDetail() method return a string which includes information of disc. When calling System.***out***.println(dvd.getDetail()); The program will print out that information.