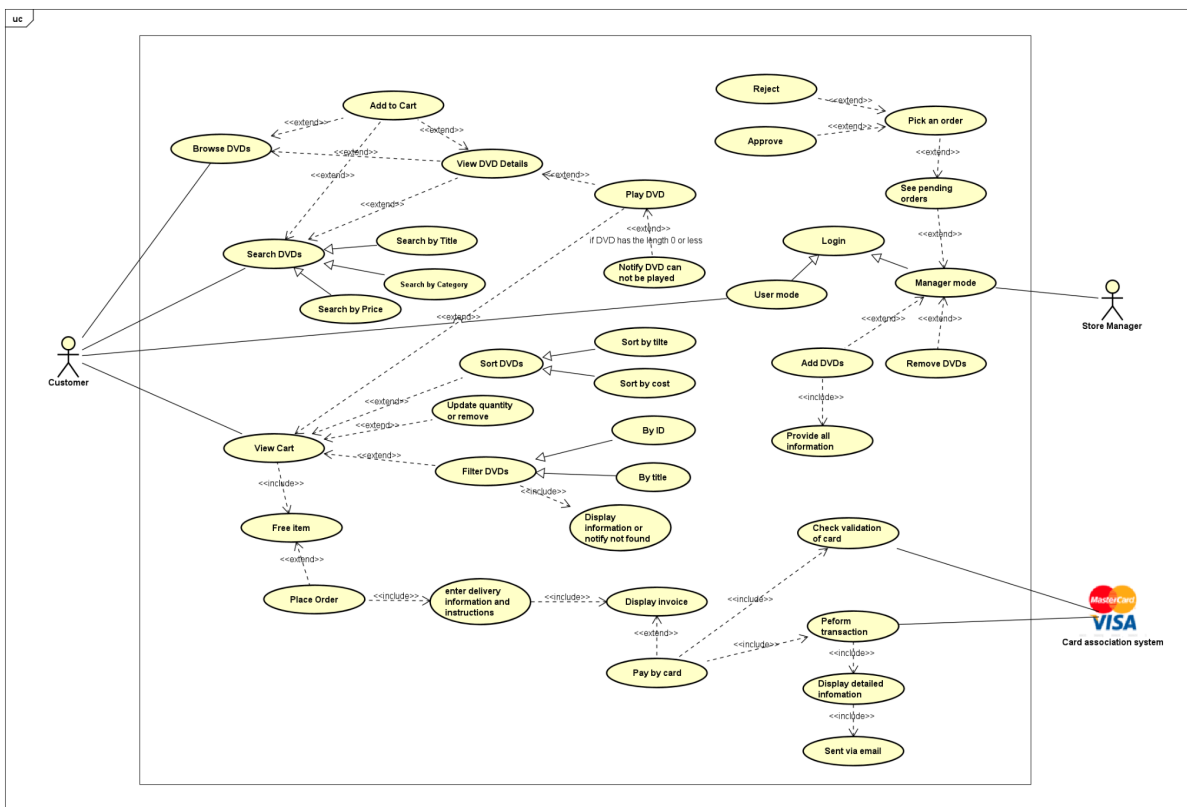
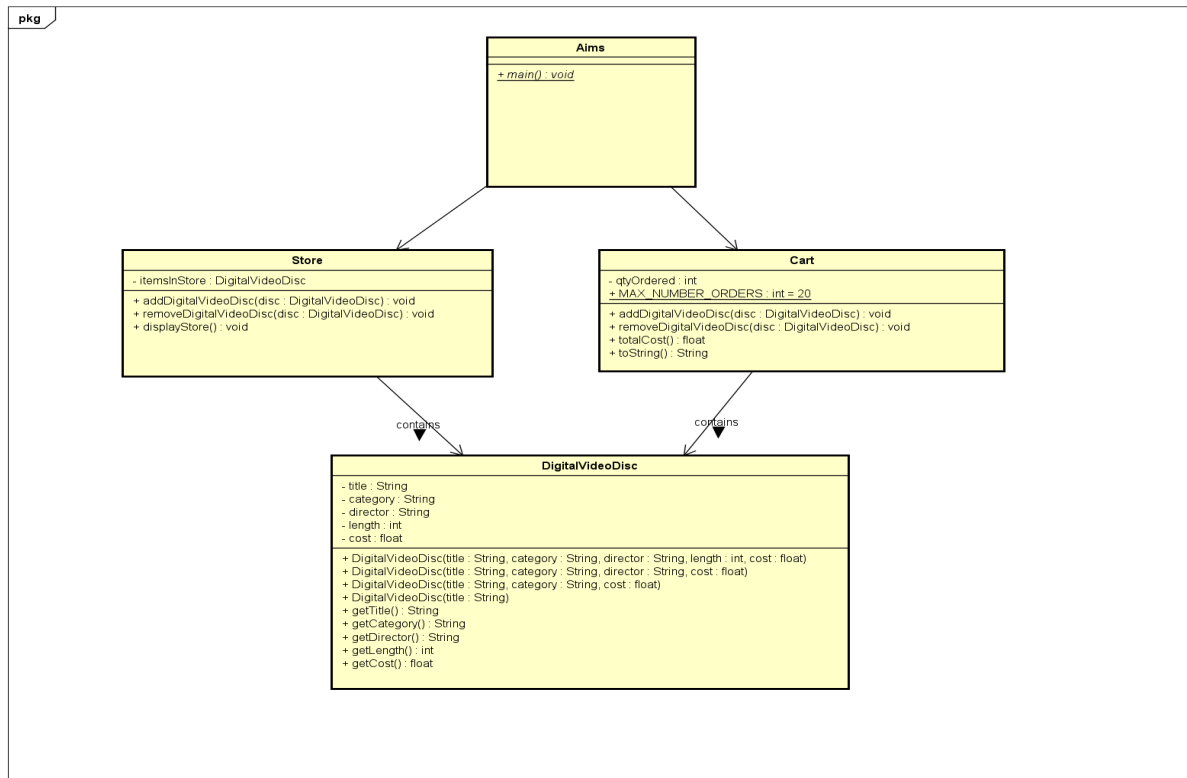


# REPORT LAB 03

## 1. Update use-case diagram and class diagram



## 2. Method overloading

```
public void addDigitalVideoDisc(DigitalVideoDisc... discs) {
    for (DigitalVideoDisc disc : discs) {
        if (qtyOrdered == MAX_ORDERED) {
            System.out.println(x:"The cart is full. Cannot add more discs.");
            break;
        } else {
            itemsOrdered[qtyOrdered] = disc;
            qtyOrdered++;
            System.out.println(x:"The disc has been added");
        }
    }
}
```

- In this case, varargs is the better choice because it provides cleaner and more intuitive usage for adding multiple DVDs to the cart. It avoids the extra step of creating an array.

```
public void addDigitalVideoDisc(DigitalVideoDisc disc1, DigitalVideoDisc disc2) {
    if (qtyOrdered >= MAX_ORDERED){
        System.out.println(x:"The cart is full");
    }
    else {
        itemsOrdered[qtyOrdered] = disc1;
        qtyOrdered++;
        System.out.println(x:"The first cart has been added successfully");
        if (qtyOrdered == MAX_ORDERED){
            System.out.println(x:"The cart is full");
        }else{
            System.out.println(x:"The second cart has been added successfully");
        }
    }
}
```

## 3. Passing parameter

```

1 package hust.soict.dsai.test.disc;
2 import hust.soict.dsai.aims.disc.DigitalVideoDisc;
3
4 public class TestPassingParameter {
5     Run | Debug
6     public static void main(String[] args) {
7         // TODO Auto-generated method stub
8         DigitalVideoDisc jungleDVD = new DigitalVideoDisc(title:"Jungle");
9         DigitalVideoDisc cinderellaDVD = new DigitalVideoDisc(title:"Cinderella");
10
11         swap(jungleDVD, cinderellaDVD);
12         System.out.println("jungle dvd title: " + jungleDVD.getTitle());
13         System.out.println("cinderella dvd title: " + cinderellaDVD.getTitle());
14
15         changeTitle(jungleDVD, cinderellaDVD.getTitle());
16         System.out.println("jungle dvd title: " + jungleDVD.getTitle());
17     }
18 } // public static void swap(DigitalVideoDisc o1, DigitalVideoDisc o2) {...
19
20 public static void swap(Object o1, Object o2) {
21     Object tmp = o1;
22     o1 = o2;
23     o2 = tmp;
24 }
25
26 public static void changeTitle(DigitalVideoDisc dvd, String title) {
27     String oldTitle = dvd.getTitle();
28     dvd.setTitle(title);
29     dvd = new DigitalVideoDisc(oldTitle);
30 }
31
32 }
33
34
35

```

Ln 18, Col 7

```

jungle dvd title: Jungle
cinderella dvd title: Cinderella
cinderella dvd title: Cinderella
jungle dvd title: Cinderella

```

- *Is JAVA a Pass by Value or a Pass by Reference programming language?*
  - + Java always **pass by value**, so the method swap only swap the copy value of object -> the actual title won't change
  - + the reason why method changeTitle works is by using setTitle to modify directly the object
- New code that will work

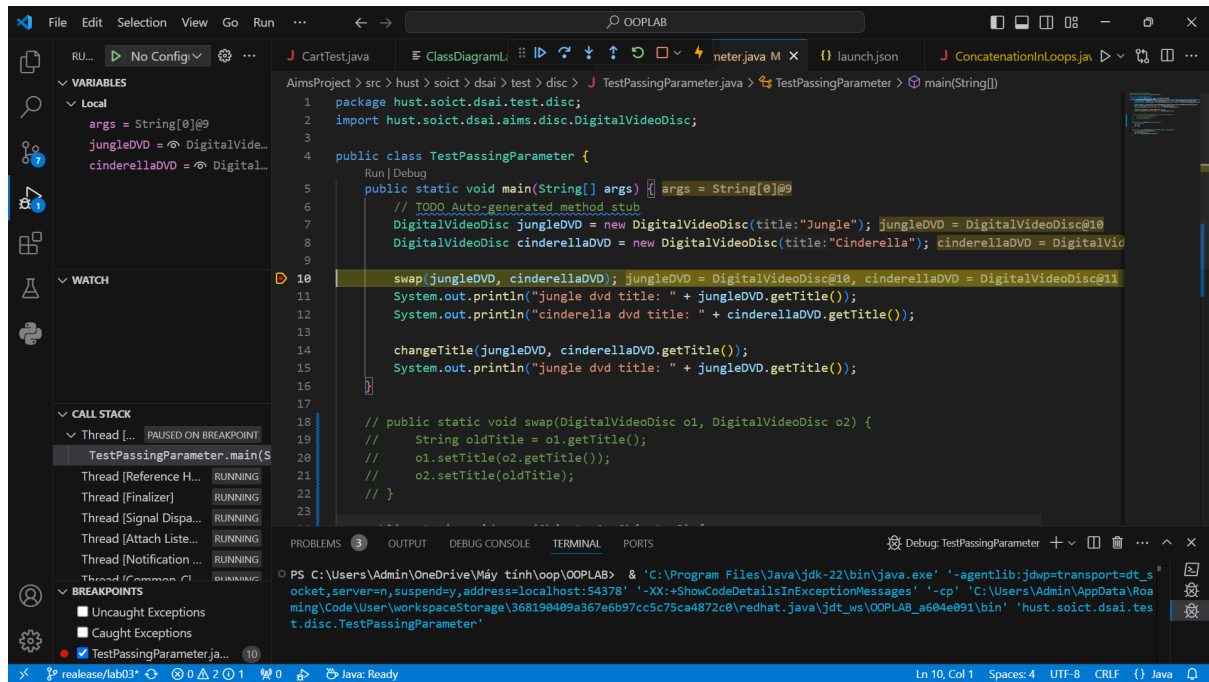
```

public static void swap(DigitalVideoDisc o1, DigitalVideoDisc o2) {
    String oldTitle = o1.getTitle();
    o1.setTitle(o2.getTitle());
    o2.setTitle(oldTitle);
}

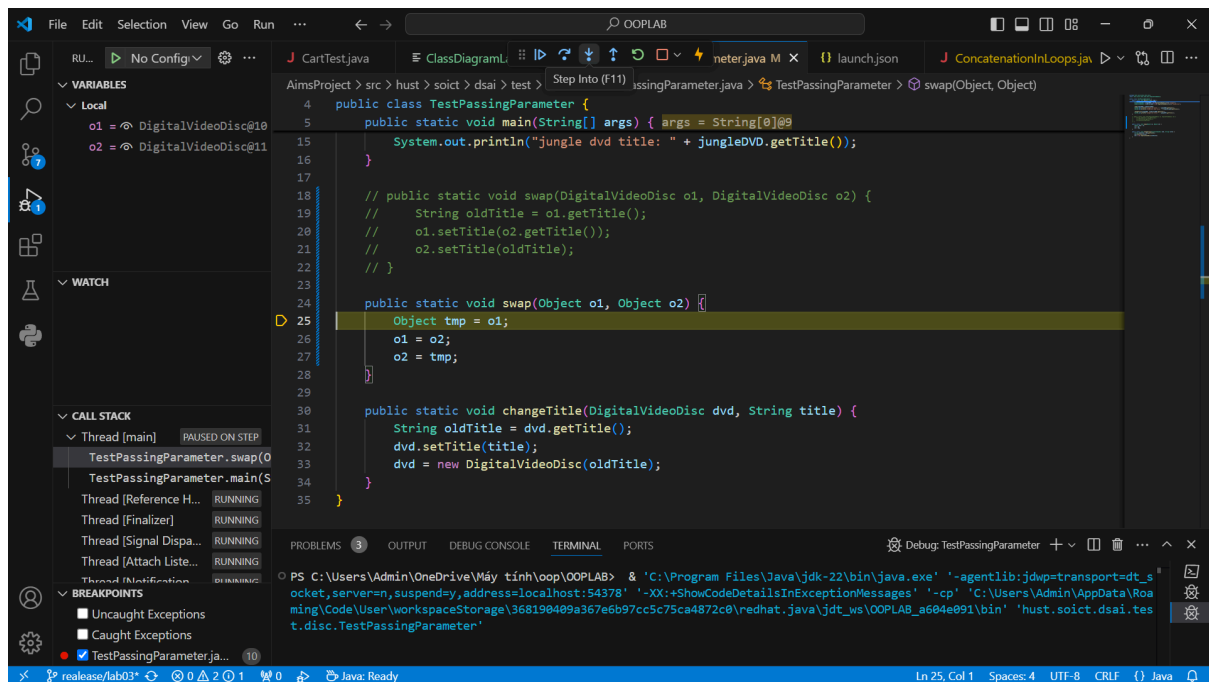
```

## 4. Debugging

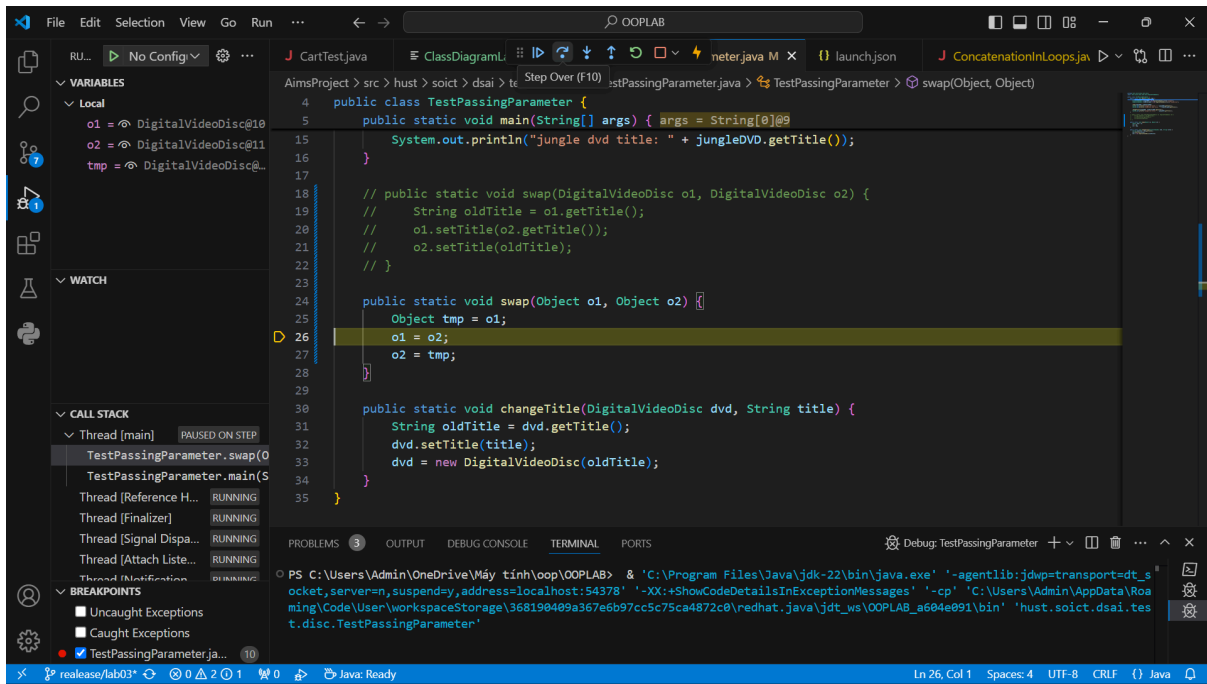
### - break point



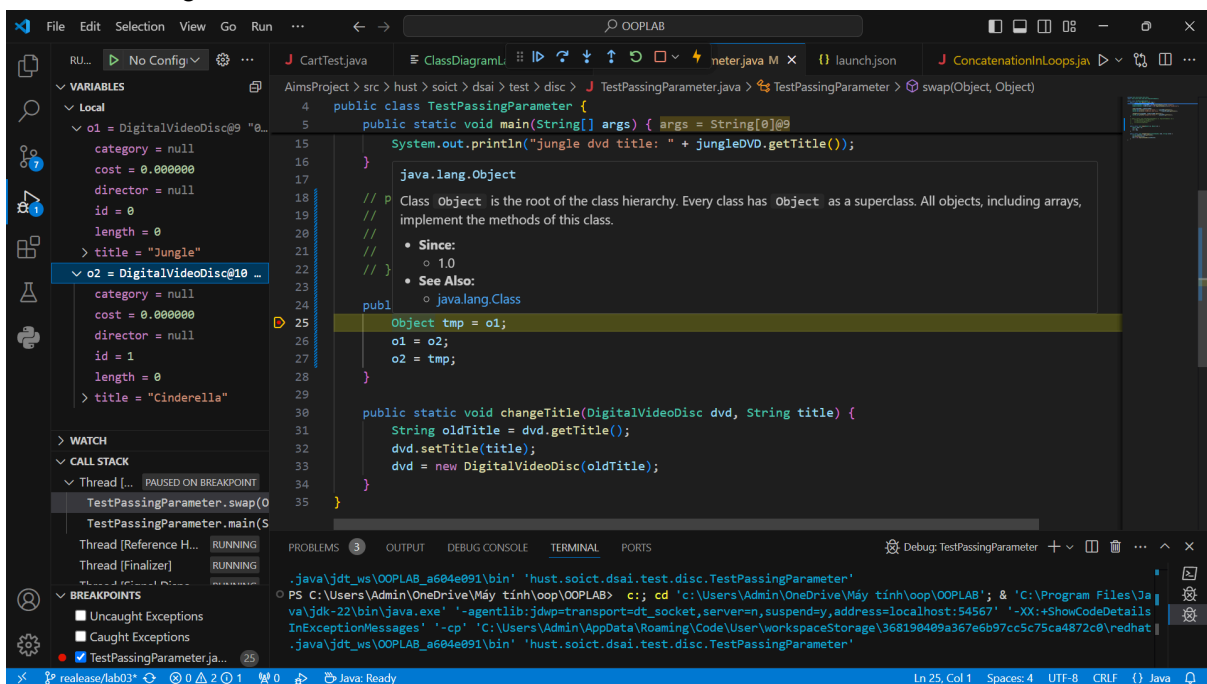
### - step into



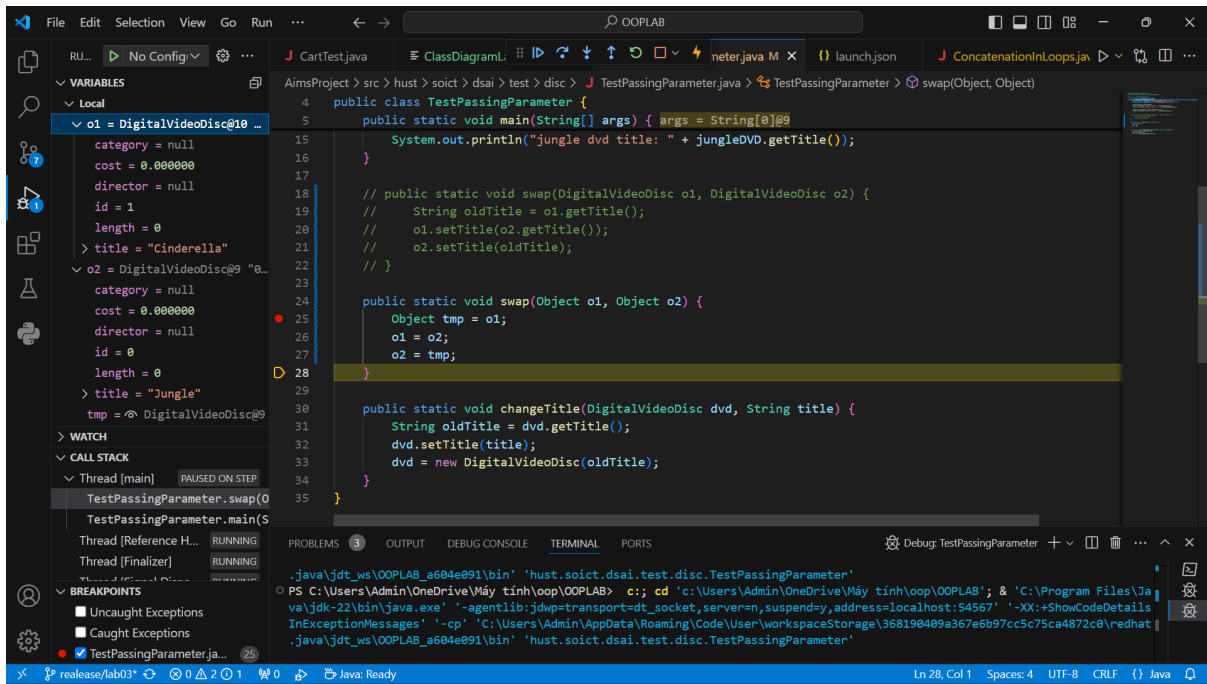
### - step over



- investigate attributes of variables



- step over swap function



- change title attribute value with "abc"

```

InExceptionMessages' '-cp' 'C:
.java\jdt_ws\OOPLAB_a604e091\
jungle dvd title: abc

```

## 5. Classifier Member and Instance Member

```

private int id;
private static int nbDigitalVideoDiscs = 0;

public DigitalVideoDisc(String title) {
    super();
    this.title = title;
    this.id = nbDigitalVideoDiscs;
    nbDigitalVideoDiscs++;
}

public DigitalVideoDisc(String title, String category, float cost) {
    super();
    this.title = title;
    this.category = category;
    this.cost = cost;
    this.id = nbDigitalVideoDiscs;
    nbDigitalVideoDiscs++;
}

public DigitalVideoDisc(String title, String category, String director, float cost) {
    super();
    this.title = title;
    this.category = category;
    this.director = director;
    this.cost = cost;
    this.id = nbDigitalVideoDiscs;
    nbDigitalVideoDiscs++;
}

public DigitalVideoDisc(String title, String category, String director, int length, float cost) {
    super();
    this.title = title;
    this.category = category;
    this.director = director;
    this.length = length;
    this.cost = cost;
    this.id = nbDigitalVideoDiscs;
    nbDigitalVideoDiscs++;
}

```

## 6. Open the Cart class

```

public String toString() {
    return this.id + " - " + this.title + " - " + this.category + " - " + this.director + " - " + this.length + " : " + this.cost + " $";
}

```

- Write a **toString()** method for the **DigitalVideoDisc** class. What should be the return type of this method?
  - + it should return a String
- Write a boolean **isMatch(String title)** method in the **DigitalVideoDisc** which finds out if the corresponding disk is a match given the title.

```

public boolean isMatch (String title){
    return this.title.equals(title);
}

```

- Search by id and title

```
public void searchById(int idNum) {
    int found = 0;
    for(int i = 0; i < qtyOrdered; i++) {
        if (idNum == i){
            System.out.println(itemsOrdered[i].toString());
            found = 1;
            break;
        }
    }
    if(found == 0){
        System.out.println(x:"DVD not found");
    }
}

public void searchByTitle(String title) {
    int found = 0;
    for(int i = 0; i < qtyOrdered; i++){
        if(itemsOrdered[i].isMatch(title)){
            System.out.println(itemsOrdered[i].toString());
            found = 1;
            break;
        }
    }
    if(found == 0){
        System.out.println(x:"DVD not found");
    }
}
```

- CartTest class



```

1 package hust.soict.dsai.test.cart;
2 import hust.soict.dsai.aims.cart.Cart;
3 import hust.soict.dsai.aims.disc.DigitalVideoDisc;
4 public class CartTest {
5     Run | Debug
6     public static void main(String[] args) {
7         //Create a new cart
8         Cart cart = new Cart();
9         //Create new dvd objects and add them to the cart
10        DigitalVideoDisc dvd1 = new DigitalVideoDisc(title:"The Liems King",
11            category:"Animation", director:"Roger Allers", length:87, cost:19.95f);
12        cart.addDigitalVideoDisc(dvd1);
13
14        DigitalVideoDisc dvd2 = new DigitalVideoDisc(title:"Star Wars",
15            category:"Science Fiction", director:"George Lucas", length:87, cost:24.95f);
16        cart.addDigitalVideoDisc(dvd2);
17
18        DigitalVideoDisc dvd3 = new DigitalVideoDisc(title:"Aladin",
19            category:"Animation", cost:18.99f);
20        cart.addDigitalVideoDisc(dvd3);
21
22        //Test the print method
23        cart.printCart();
24
25        //Test the search methods
26        cart.searchById(idNum:5);
27        cart.searchByTitle(title:"Aladin");
28    }
29 }

```

## 7. Implement the Store class

### - Store class

```

package hust.soict.dsai.aims.store;
import hust.soict.dsai.aims.disc.DigitalVideoDisc;

public class Store {
    private DigitalVideoDisc[] itemsInStore; // Array to store DigitalVideoDiscs
    private int currentCount; // Tracks the number of DigitalVideoDiscs in the store

    // Constructor to initialize the store with a given capacity
    public Store(int capacity) {
        itemsInStore = new DigitalVideoDisc[capacity];
        currentCount = 0;
    }

    public void addDigitalVideoDisc(DigitalVideoDisc DigitalVideoDisc) {
        if (currentCount < itemsInStore.length) {
            itemsInStore[currentCount] = DigitalVideoDisc;
            currentCount++;
            System.out.println("Added DigitalVideoDisc: " + DigitalVideoDisc.toString());
        } else {
            System.out.println(x:"Store is full. Cannot add more DigitalVideoDiscs.");
        }
    }

    public void removeDigitalVideoDisc(int id) {

```

```

public void removeDigitalVideoDisc(int id) {
    boolean found = false;
    for (int i = 0; i < currentCount; i++) {
        if (itemsInStore[i].getId() == id) { // Assuming DigitalVideoDisc has a getId() method
            found = true;
            // Shift all elements after the removed DigitalVideoDisc
            for (int j = i; j < currentCount - 1; j++) {
                itemsInStore[j] = itemsInStore[j + 1];
            }
            itemsInStore[currentCount - 1] = null; // Clear the last slot
            currentCount--;
            System.out.println("Removed DigitalVideoDisc with ID: " + id);
            break;
        }
    }
    if (!found) {
        System.out.println("DigitalVideoDisc with ID " + id + " not found in the store.");
    }
}

// Method to display all DigitalVideoDiscs in the store
public void displayStore() {
    System.out.println("DigitalVideoDiscs currently in the store:");
    for (int i = 0; i < currentCount; i++) {
        System.out.println(itemsInStore[i].toString());
    }
}
}

```

#### - StoreTest class

```

1 package hust.soict.dsai.test.store;
2 import hust.soict.dsai.aims.disc.DigitalVideoDisc;
3 import hust.soict.dsai.aims.store.Store;
4
5 public class StoreTest {
6     Run | Debug
7     public static void main(String[] args) {
8         // Create a store with a capacity of 5
9         Store store = new Store(capacity:5);
10
11         // Create some DigitalVideoDiscs
12         DigitalVideoDisc dvd1 = new DigitalVideoDisc(title:"The Liems King", category:"Animation", director:"Roger Al
13         DigitalVideoDisc dvd2 = new DigitalVideoDisc(title:"Star Wars", category:"Science Fiction", director:"George
14         DigitalVideoDisc dvd3 = new DigitalVideoDisc(title:"Aladin",category:"Animation", cost:18.99f);
15
16         // Test addDigitalVideoDiscng DigitalVideoDiscs
17         store.addDigitalVideoDisc(dvd1);
18         store.addDigitalVideoDisc(dvd2);
19         store.addDigitalVideoDisc(dvd3);
20
21         // DigitalVideoDiscsplay the store
22         store.displayStore();
23
24         // Test removing a DigitalVideoDisc
25         store.removeDigitalVideoDisc(id:2);
26
27         // DigitalVideoDiscsplay the store again
28         store.displayStore();
29     }
30 }

```

## 9. String, StringBuilder and StringBuffer

- use String took a lot of time can't even wait for the result

```
OtherProject > hust\soict\globalict\garbage > J GarbageCreator.java > GarbageCreator > main(String[])
3 import java.io.IOException;
4
5 public class GarbageCreator {
6     public static void main(String[] args) {
7         String filename = "largefile.txt"; // Specify a large file path 100 MB
8         String outputString = ""; // Initialize an empty String
9         final int REPORT_INTERVAL = 100000;
10
11         try {
12             byte[] inputBytes = Files.readAllBytes(Paths.get(filename));
13             long startTime = System.currentTimeMillis();
14
15             int count = 0;
16             for (byte b : inputBytes) {
17                 outputString += (char) b; // Using + operator for concatenation
18                 count++;
19
20                 if (count % REPORT_INTERVAL == 0) {
21                     System.out.println("Processed " + count + " bytes...");
22                 }
23             }
24
25             long endTime = System.currentTimeMillis();
26         } catch (IOException e) {
27             e.printStackTrace();
28         }
29     }
30 }
```

onMessages' '-cp' 'C:\Users\Admin\AppData\Roaming\Code\User\workspaceStorage\368198409a367e6b97cc5c75ca4872c0\redhat.java\jdt' ...  
\_ws\OOPLAB\_a604e091\bin' 'GarbageCreator'  
Processed 100000 bytes...  
Processed 200000 bytes...  
Processed 300000 bytes...

- using stringBuffer and stringBuilder is way faster

```
OtherProject > hust\soict\globalict\garbage > J NoGarbage.java > NoGarbage > main(String[])
1 import java.nio.file.Files;
2 import java.nio.file.Paths;
3 import java.io.IOException;
4
5 public class NoGarbage {
6     public static void main(String[] args) {
7         String filename = "largefile.txt"; // Specify a large file path
8         StringBuilder outputString = new StringBuilder(); // Using StringBuilder
9         final int REPORT_INTERVAL = 100000;
10
11         try {
12             byte[] inputBytes = Files.readAllBytes(Paths.get(filename));
13             long startTime = System.currentTimeMillis();
14
15             int count = 0;
16             for (byte b : inputBytes) {
17                 outputString.append((char) b); // Efficiently appending to StringBuilder
18                 count++;
19
20                 if (count % REPORT_INTERVAL == 0) {
21                     System.out.println("Processed " + count + " bytes...");
22                 }
23             }
24
25             long endTime = System.currentTimeMillis();
26             System.out.println("Time taken (NoGarbage): " + (endTime - startTime) + " ms");
27         } catch (IOException e) {
28             e.printStackTrace();
29         }
30     }
31 }
```

Processed 104500000 bytes...  
Processed 104600000 bytes...  
Processed 104700000 bytes...  
Processed 104800000 bytes...  
Time taken (NoGarbage): 1608 ms