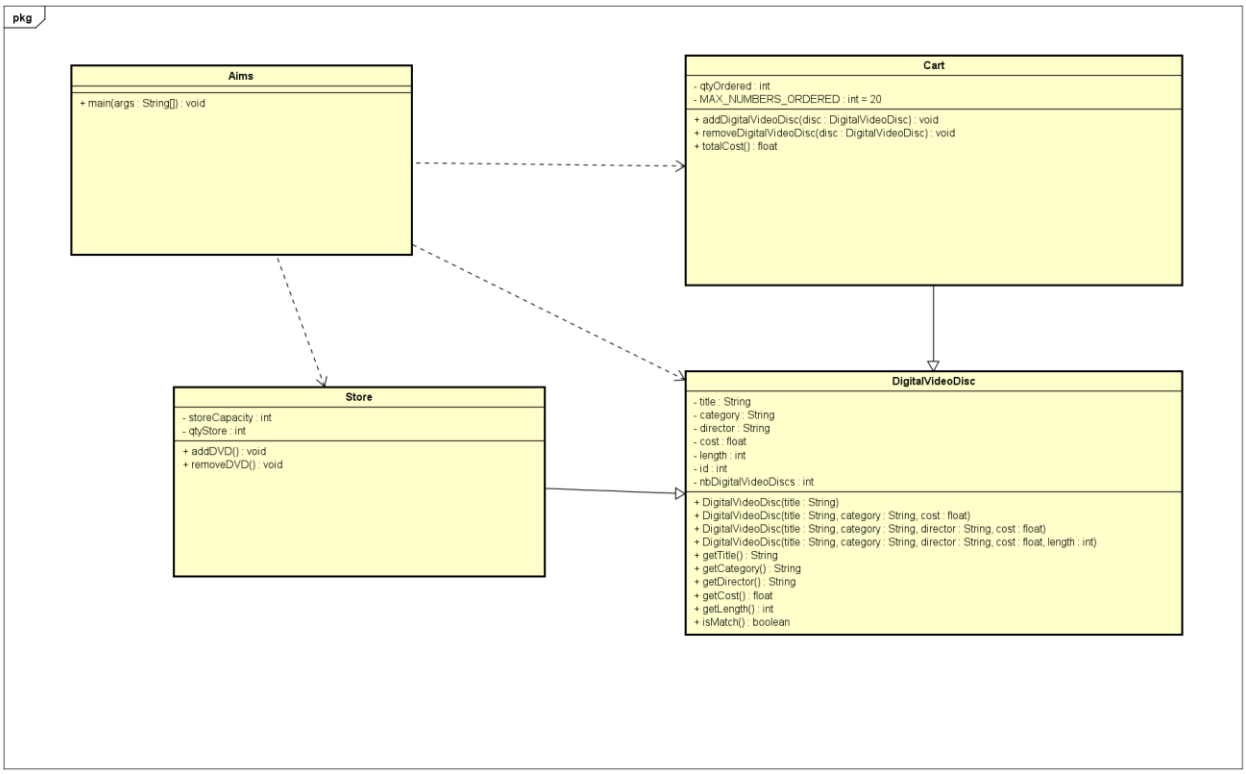
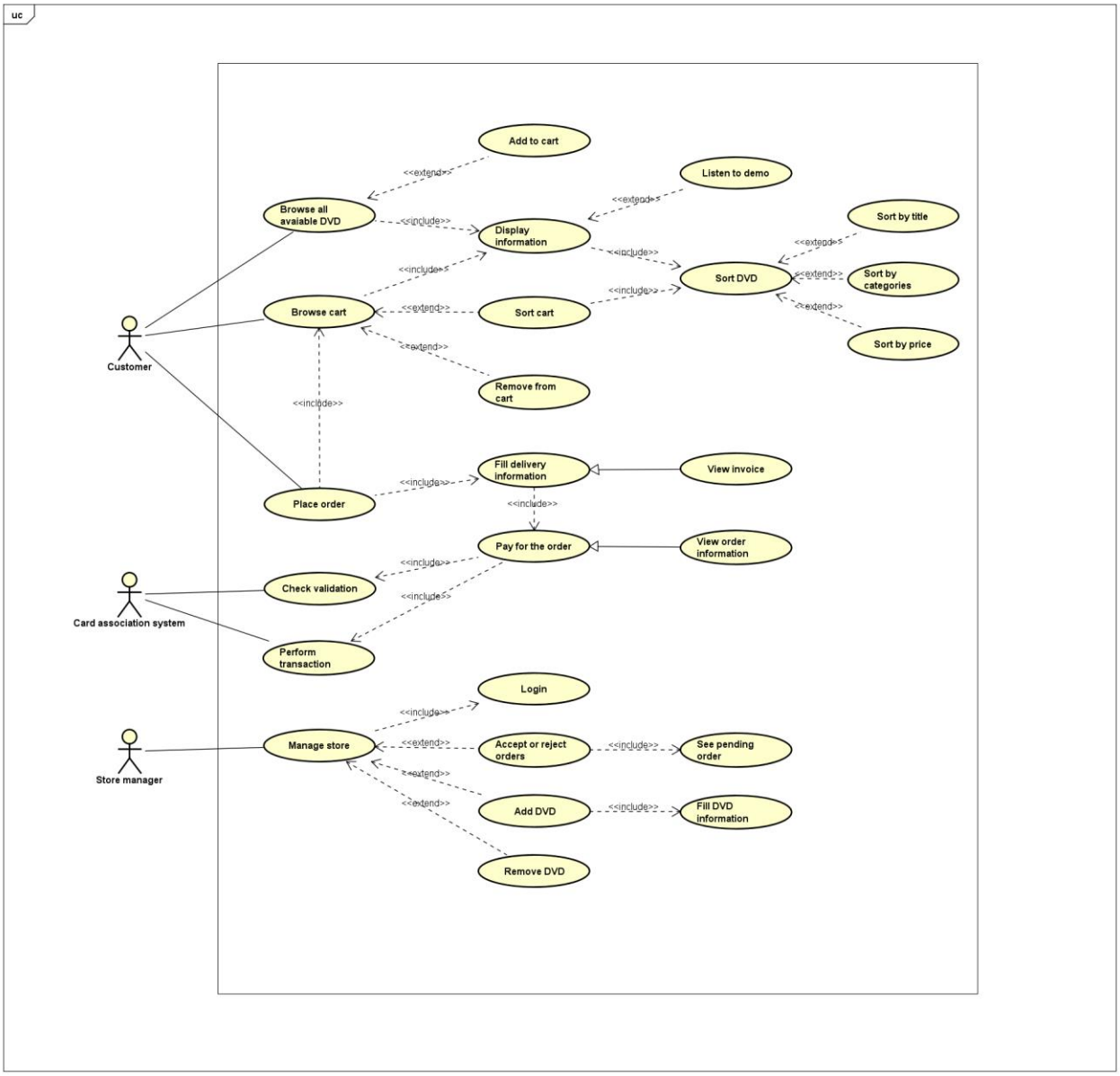
Name: Đặng Thịnh Tường Minh

Student ID: 20235528

Use-case diagram and class diagram





Working with method overloading

```java
public void addDigitalVideoDisc(DigitalVideoDisc [] dvdList) {
    for (DigitalVideoDisc disc : dvdList) {
        if (qtyOrdered < MAX_NUMBERS_ORDERED) {
            itemsOrdered[qtyOrdered] = disc;
            qtyOrdered += 1;
            System.out.println(x:"DVD list added");
        } else {
            System.out.println(x:"Maximum capacity reached");
        }
    }
}

public void addDigitalVideoDisc(DigitalVideoDisc dvd1,DigitalVideoDisc dvd2) {

    if (qtyOrdered + 1 < MAX_NUMBERS_ORDERED) {
        itemsOrdered[qtyOrdered] = dvd1;
        itemsOrdered[qtyOrdered + 1] = dvd2;
        qtyOrdered += 2;
        System.out.println(x:"2 DVD added");
    } else {
        System.out.println(x:"Maximum capacity reached");
    }
}
```

- 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?

I would prefer array parameter as it is easier to add a certain amount of DVD

Passing parameter

True swap method:

```java
public class TestPassingParameter {

    public static void main(String[] args) {
        // TODO Auto-generated method stub
        DigitalVideoDisc jungleDVD = new DigitalVideoDisc(title:"Jungle");
        DigitalVideoDisc cinderellaDVD = new DigitalVideoDisc(title:"Cinderella");

        swap(jungleDVD, cinderellaDVD);
        System.out.println("jungle dvd title: " + jungleDVD.getTitle());
        System.out.println("cinderella dvd title: " + cinderellaDVD.getTitle());

        changeTitle(jungleDVD, cinderellaDVD.getTitle());
        System.out.println("jungle dvd title: " + jungleDVD.getTitle());
    }

    public static void swap(DigitalVideoDisc dvd1, DigitalVideoDisc dvd2) {
        String tempTitle = dvd1.getTitle();
        dvd1.setTitle(dvd2.getTitle());
        dvd2.setTitle(tempTitle);
    }


    public static void changeTitle(DigitalVideoDisc dvd, String title) {
        String oldTitle = dvd.getTitle();
        dvd.setTitle(title);
        dvd = new DigitalVideoDisc(oldTitle);
    }
}
```

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

JAVA is a pass by value programming language. The dvd1 and dvd2 in the method are copies of the reference to the original dvd1 and dvd2

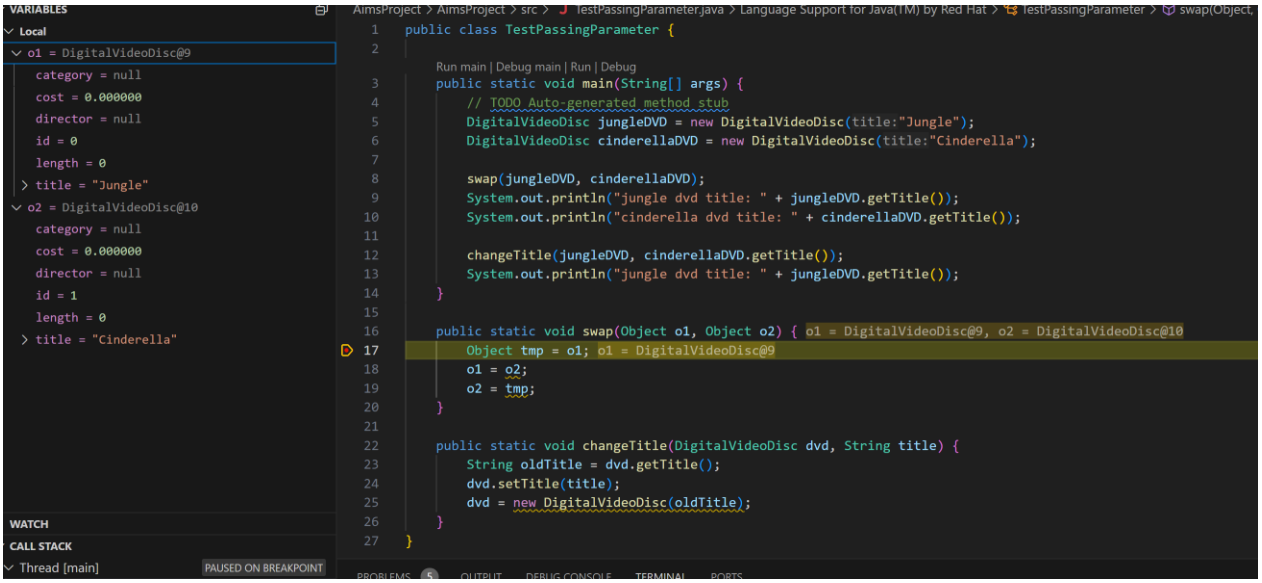
- After the call of **swap(jungleDVD, cinderellaDVD)** why does the title of these two objects still remain?

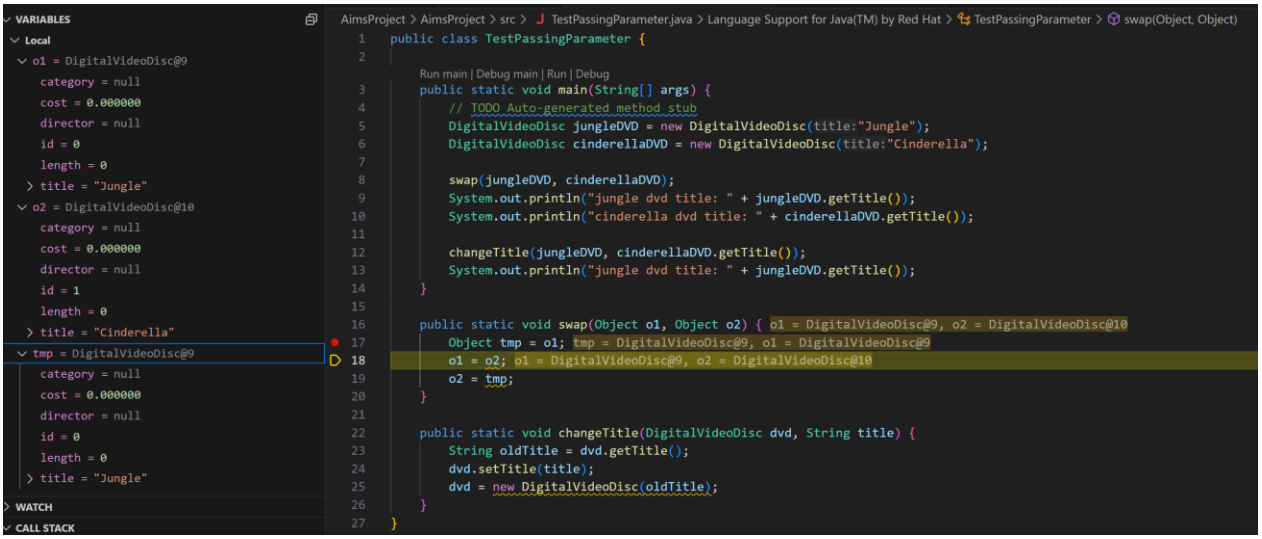
The dvd1 and dvd2 in the method are copies of the reference to the original dvd1 and dvd2, therefore swap only swap the copies, not change the original object

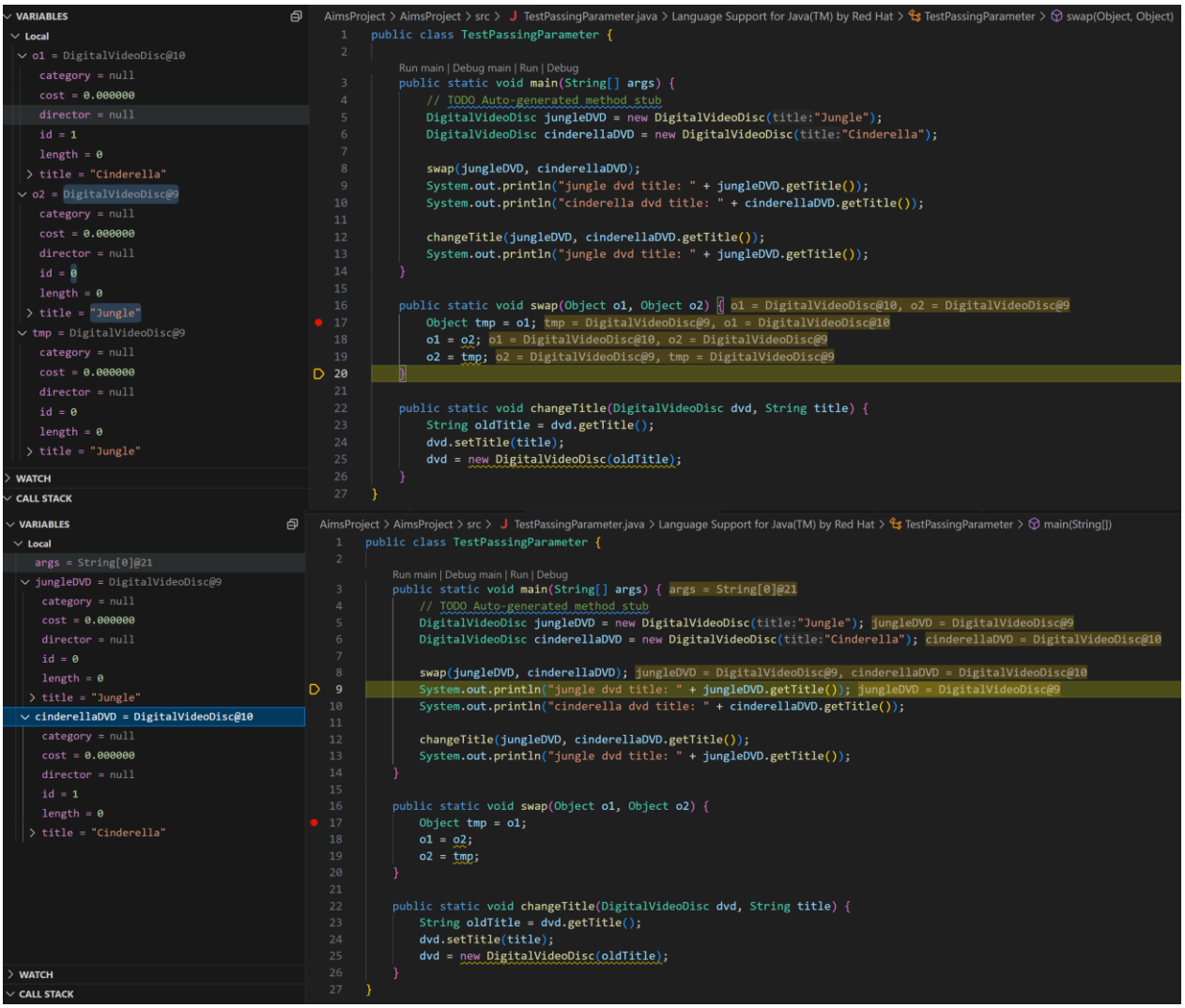
- After the call of **changeTitle(jungleDVD, cinderellaDVD.getTitle())** why is the title of the JungleDVD changed?

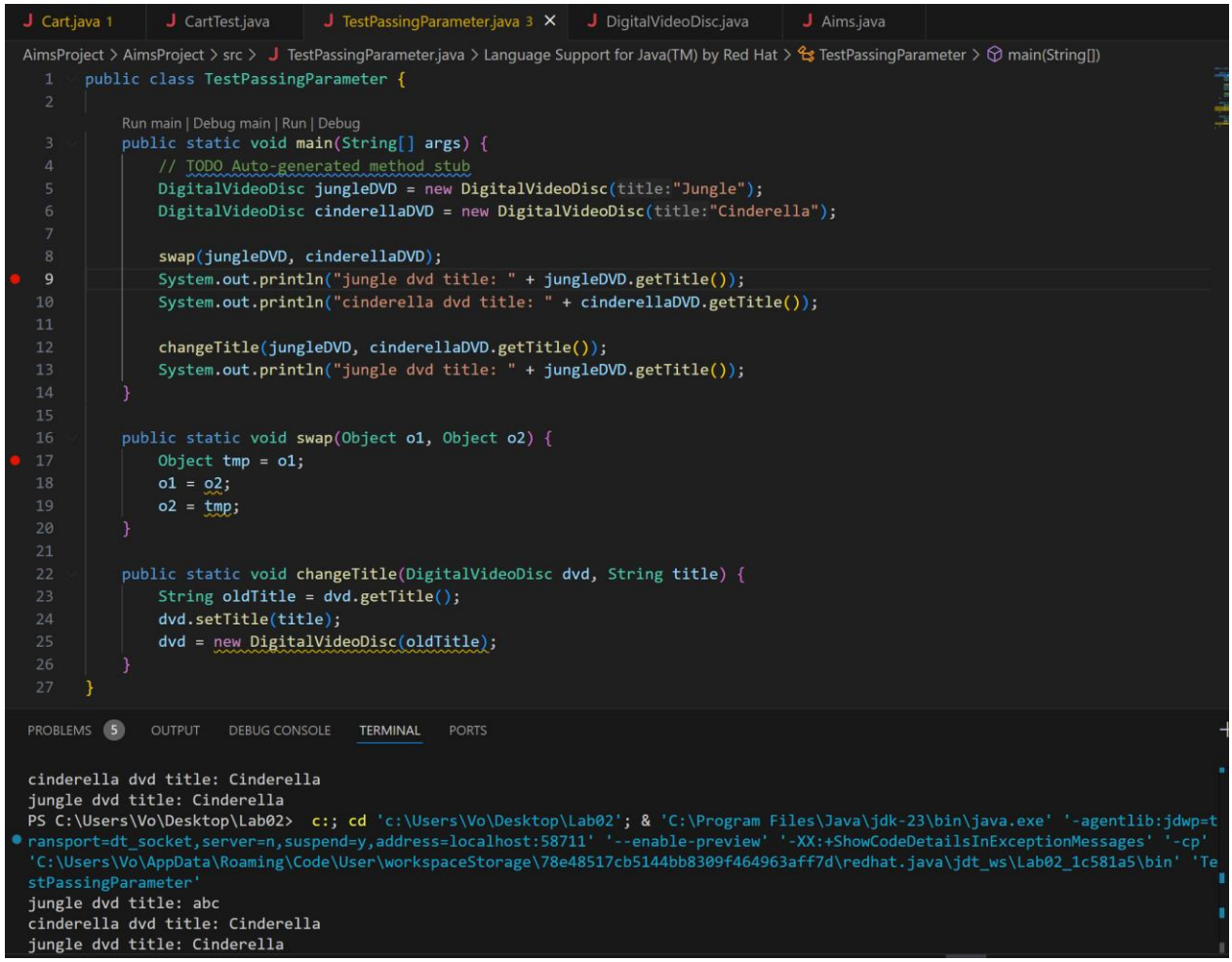
changeTitle take the address and modify title at that address, therefore it also change the original dvd

Debug screenshot






VARIABLES
Local
o1 = DigitalVideoDisc@10
category = null
cost = 0.000000
director = null
id = 1
length = 0
title = "Cinderella"
o2 = DigitalVideoDisc@9
category = null
cost = 0.000000
director = null
id = 0
length = 0
title = "Jungle"
tmp = DigitalVideoDisc@9
category = null
cost = 0.000000
director = null
id = 0
length = 0
title = "Jungle"
WATCH
CALL STACK
AimsProject > AimsProject > src > TestPassingParameter.java > Language Support for Java(TM) by Red Hat > TestPassingParameter > swap(Object, Object)
public class TestPassingParameter {
Run main | Debug main | Run | Debug
public static void main(String[] args) {
// TODO Auto-generated method stub
DigitalVideoDisc jungleDVD = new DigitalVideoDisc(title:"Jungle");
DigitalVideoDisc cinderellaDVD = new DigitalVideoDisc(title:"Cinderella");
swap(jungleDVD, cinderellaDVD);
System.out.println("jungle dvd title: " + jungleDVD.getTitle());
System.out.println("cinderella dvd title: " + cinderellaDVD.getTitle());
changeTitle(jungleDVD, cinderellaDVD.getTitle());
System.out.println("jungle dvd title: " + jungleDVD.getTitle());
}
public static void swap(Object o1, Object o2) { o1 = DigitalVideoDisc@10, o2 = DigitalVideoDisc@9
Object tmp = o1; tmp = DigitalVideoDisc@9, o1 = DigitalVideoDisc@10
o1 = o2; o1 = DigitalVideoDisc@10, o2 = DigitalVideoDisc@9
o2 = tmp; o2 = DigitalVideoDisc@9, tmp = DigitalVideoDisc@9
}
public static void changeTitle(DigitalVideoDisc dvd, String title) {
String oldTitle = dvd.getTitle();
dvd.setTitle(title);
dvd = new DigitalVideoDisc(oldTitle);
}
}
VARIABLES
Local
args = String[0]@21
jungleDVD = DigitalVideoDisc@9
category = null
cost = 0.000000
director = null
id = 0
length = 0
title = "Jungle"
cinderellaDVD = DigitalVideoDisc@10
category = null
cost = 0.000000
director = null
id = 1
length = 0
title = "Cinderella"
WATCH
CALL STACK
AimsProject > AimsProject > src > TestPassingParameter.java > Language Support for Java(TM) by Red Hat > TestPassingParameter > main(String[])
public class TestPassingParameter {
Run main | Debug main | Run | Debug
public static void main(String[] args) { args = String[0]@21
// TODO Auto-generated method stub
DigitalVideoDisc jungleDVD = new DigitalVideoDisc(title:"Jungle"); jungleDVD = DigitalVideoDisc@9
DigitalVideoDisc cinderellaDVD = new DigitalVideoDisc(title:"Cinderella"); cinderellaDVD = DigitalVideoDisc@10
swap(jungleDVD, cinderellaDVD); jungleDVD = DigitalVideoDisc@9, cinderellaDVD = DigitalVideoDisc@10
System.out.println("jungle dvd title: " + jungleDVD.getTitle()); jungleDVD = DigitalVideoDisc@9
System.out.println("cinderella dvd title: " + cinderellaDVD.getTitle());
changeTitle(jungleDVD, cinderellaDVD.getTitle());
System.out.println("jungle dvd title: " + jungleDVD.getTitle());
}
public static void swap(Object o1, Object o2) {
Object tmp = o1;
o1 = o2;
o2 = tmp;
}
public static void changeTitle(DigitalVideoDisc dvd, String title) {
String oldTitle = dvd.getTitle();
dvd.setTitle(title);
dvd = new DigitalVideoDisc(oldTitle);
}
}



Classifier Member and Instance Member

```java
public class DigitalVideoDisc {
    private String title;
    private String category;
    private String director;
    private int length;
    private float cost;
    private int id;

    private static int nbDigitalVideoDiscs = 0;
```

```java
public DigitalVideoDisc(String title) {
    super();
    this.title = title;
    this.id = nbDigitalVideoDiscs;
    nbDigitalVideoDiscs += 1;
}

public DigitalVideoDisc(String title, String category, float cost) {
    super();
    this.title = title;
    this.category = category;
    this.cost = cost;
    this.id = nbDigitalVideoDiscs;
    nbDigitalVideoDiscs += 1;
}

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 += 1;
}

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 += 1;
}
```

Open the Cart class

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

It should be String

```java
public void print() {
    System.out.println(x:"**********CART**********");
    System.out.println(x:"Ordered Items:");

    for (int i = 0; i < qtyOrdered; i++) {

        System.out.print((i + 1) + ". DVD - ");

        if (itemsOrdered[i].getTitle() != null) {
            System.out.print(itemsOrdered[i].getTitle() + " - ");
        }
        if (itemsOrdered[i].getCategory() != null) {
            System.out.print(itemsOrdered[i].getCategory() + " - ");
        }
        if (itemsOrdered[i].getDirector() != null) {
            System.out.print(itemsOrdered[i].getDirector() + " - ");
        }
        if (itemsOrdered[i].getLength() != 0) {
            System.out.print(itemsOrdered[i].getLength() + ": ");
        }
        if (itemsOrdered[i].getCost() != 0) {
            System.out.println(itemsOrdered[i].getCost() + " $");
        }
    }

    System.out.println("Total cost: " + this.totalCost() + " $");

    System.out.println(x:"**********");
}
```

```java
public void search(int id) {

    boolean found = false;

    for (int i = 0; i < qtyOrdered; i++) {

        if (itemsOrdered[i].isMatch(id)) {

            System.out.print("Found: " + (i + 1) + ". DVD - ");

            if (itemsOrdered[i].getTitle() != null) {
                System.out.print(itemsOrdered[i].getTitle() + " - ");
            }
            if (itemsOrdered[i].getCategory() != null) {
                System.out.print(itemsOrdered[i].getCategory() + " - ");
            }
            if (itemsOrdered[i].getDirector() != null) {
                System.out.print(itemsOrdered[i].getDirector() + " - ");
            }
            if (itemsOrdered[i].getLength() != 0) {
                System.out.print(itemsOrdered[i].getLength() + ": ");
            }
            if (itemsOrdered[i].getCost() != 0) {
                System.out.println(itemsOrdered[i].getCost() + " $");
            }

            found = true;
        }
    }
    if (found == false) {
        System.out.println(x:"No match is found");
    }
}
```

```java
public void search(String title) {

    boolean found = false;

    for (int i = 0; i < qtyOrdered; i++) {

        if (itemsOrdered[i].isMatch(title)) {

            System.out.print("Found: " + (i + 1) + ". DVD - ");

            if (itemsOrdered[i].getTitle() != null) {
                System.out.print(itemsOrdered[i].getTitle() + " - ");
            }
            if (itemsOrdered[i].getCategory() != null) {
                System.out.print(itemsOrdered[i].getCategory() + " - ");
            }
            if (itemsOrdered[i].getDirector() != null) {
                System.out.print(itemsOrdered[i].getDirector() + " - ");
            }
            if (itemsOrdered[i].getLength() != 0) {
                System.out.print(itemsOrdered[i].getLength() + ": ");
            }
            if (itemsOrdered[i].getCost() != 0) {
                System.out.println(itemsOrdered[i].getCost() + " $");
            }

            found = true;
        }
    }
    if (found == false) {
        System.out.println(x:"No match is found");
    }
}
```

```java
package hust.soict.dsai.test.cart;
import hust.soict.dsai.aims.cart.Cart;
import hust.soict.dsai.aims.disc.DigitalVideoDisc;

public class CartTest {

    Run main | Debug main | Run | Debug
    public static void main(String[] args) {
        // Create a new cart
        Cart cart = new Cart();

        // Create new dvd objects and add them to the cart
        DigitalVideoDisc dvd1 = new DigitalVideoDisc(title:"The Lion King",
                category:"Animation", director:"Roger Allers", length:87, cost:19.95f);
        cart.addDigitalVideoDisc(dvd1);

        DigitalVideoDisc dvd2 = new DigitalVideoDisc(title:"Star Wars",
                category:"Science Fiction", director:"George Lucas", length:87, cost:24.95f);
        cart.addDigitalVideoDisc(dvd2);

        DigitalVideoDisc dvd3 = new DigitalVideoDisc(title:"Aladdin",
                category:"Animation", cost:18.99f);
        cart.addDigitalVideoDisc(dvd3);

        // Test the print method
        cart.print();

        // To-do: Test the search methods here
        cart.search(id:2);
        cart.search(title:"Star Wars");
    }
}
```

```
DVD added
DVD added
**********CART**********
Ordered Items:
1. DVD - The Lion King - Animation - Roger Allers - 87: 19.95 $
2. DVD - Star Wars - Science Fiction - George Lucas - 87: 24.95 $
3. DVD - Aladdin - Animation - 18.99 $
Total cost: 63.89 $
**************************************************
Found: 3. DVD - Aladdin - Animation - 18.99 $
Found: 2. DVD - Star Wars - Science Fiction - George Lucas - 87: 24.95 $
PS C:\Users\Vo\Desktop\Lab03>
```

Implement Store class

```java
public class Store {

    private final int storeCapacity = 1000;
    private DigitalVideoDisc itemsInStore[] = new DigitalVideoDisc[1000];

    private int qtyStore = 0;

    public void addDVD(DigitalVideoDisc disc) {

        if (qtyStore < storeCapacity) {
            itemsInStore[qtyStore] = disc;
            qtyStore += 1;
            System.out.println(x:"DVD added");
        } else {
            System.out.println(x:"Maximum capacity reached");
        }
    }

    public void removeDVD(DigitalVideoDisc disc) {

        boolean found = false;

        for (int i = 0; i < qtyStore; i++) {

            if (itemsInStore[i].equals(disc)) {
                itemsInStore[i] = null;
                found = true;
                for (int j = i; j < qtyStore; j++) {
                    itemsInStore[j] = itemsInStore[j + 1];
                }
                qtyStore -= 1;
                System.out.println(x:"DVD deleted");
            }
        }

        if (found == false) {
            System.out.println(x:"No DVD founded");
        }
    }
```

```java
package hust.soict.dsai.test.store;
import hust.soict.dsai.aims.disc.DigitalVideoDisc;
import hust.soict.dsai.aims.store.Store;

public class StoreTest {

    Run main | Debug main | Run | Debug
    public static void main(String[] args) {
        // Create a new cart
        Store store = new Store();

        // Create new dvd objects and add them to the store
        DigitalVideoDisc dvd1 = new DigitalVideoDisc(title:"The Lion King",
                category:"Animation", director:"Roger Allers", length:87, cost:19.95f);
        store.addDVD(dvd1);

        DigitalVideoDisc dvd2 = new DigitalVideoDisc(title:"Star Wars",
                category:"Science Fiction", director:"George Lucas", length:87, cost:24.95f);
        store.addDVD(dvd2);

        DigitalVideoDisc dvd3 = new DigitalVideoDisc(title:"Aladdin",
                category:"Animation", cost:18.99f);
        store.addDVD(dvd3);

        store.removeDVD(dvd2);
    }
}
```

String, StringBuilder and StringBuffer

```java
package hust.soict.dsai.garbage;

import java.util.Random;

public class ConcatenationInLoops {
    Run main | Debug main | Run | Debug
    public static void main(String[] args) {
        Random r = new Random(seed:123);
        long start = System.currentTimeMillis();

        String s = "";
        for (int i = 0; i < 65536; i++) {
            s += r.nextInt(bound:2);
        }
        System.out.println(System.currentTimeMillis() - start); // This prints roughly 4500.

        r = new Random(seed:123);
        start = System.currentTimeMillis();

        StringBuilder sb = new StringBuilder();
        for (int i = 0; i < 65536; i++) {
            sb.append(r.nextInt(bound:2));
        }
        s = sb.toString();
        System.out.println(System.currentTimeMillis() - start); // This prints 5.
    }
}
```

```java
package hust.soict.dsai.garbage;

import java.io.IOException;
import java.nio.file.Files;
import java.nio.file.Paths;

public class GarbageCreator {
    Run main | Debug main | Run | Debug
    public static void main(String[] args) throws IOException {
        String filepath = "C:/Users/Vo/Desktop/large_input.txt";
        byte[] inputBytes = {0};
        long startTime, endTime;

        inputBytes = Files.readAllBytes(Paths.get(filepath));
        startTime = System.currentTimeMillis();
        String outputString = "";

        for (byte b : inputBytes) {
            outputString += (char)b;
        }

        endTime = System.currentTimeMillis();
        System.out.println(endTime - startTime);
    }
}
```

```java
package hust.soict.dsai.garbage;

import java.io.IOException;
import java.nio.file.Files;
import java.nio.file.Paths;

public class NoGarbage {
    Run main | Debug main | Run | Debug
    public static void main(String[] args) throws IOException {
        String filepath = "C:/Users/Vo/Desktop/large_input.txt";
        byte[] inputBytes = {0};
        long startTime, endTime;

        inputBytes = Files.readAllBytes(Paths.get(filepath));
        startTime = System.currentTimeMillis();
        StringBuilder outputString = new StringBuilder();

        for (byte b : inputBytes) {
            outputString.append(b);
        }

        endTime = System.currentTimeMillis();
        System.out.println(endTime - startTime);
    }
}
```