Report OOP Lab 03

Student: Lê Văn Pháp

Mssv: 20226118

Contents

1.Overloading method

1.1 Overloading by differing types of parameter

1.1.1 different type of parameter

```
public int addDigitalVideoDisc(DigitalVideoDisc [] dvdList) {
        int addedCount = 0;
        for (DigitalVideoDisc disc : dvdList) {
                if (qtyOrdered == maxCapacity) {
                        System.out.println("Full");
                        break;
                }else {
                        pickedDVD[qtyOrdered] = disc;
                        qtyOrdered++;
                        System.out.println("Added successfully");
                        addedCount++;
                }
        }
        return addedCount;
    }
```

1.1.2 allows to pass an arbitrary number of arguments for dvd

```
public int addDigitalVideoDisc(DigitalVideoDisc...dvdArray) {
        int addcount=0;
        for(DigitalVideoDisc disc : dvdArray) {
                if (qtyOrdered == maxCapacity) {
                        System.out.println("Full");
                        break;
                }else {
                        pickedDVD[qtyOrdered] = disc;
                        qtyOrdered++;
                        System.out.println("Added successfully");
                        addcount++;
                }
        }
        return addcount;
    }
```

1.1.3 My point of view

I prefer the second way because using varargs is more flexible, it allows adding any number of DVDs without specifying the array explicitly. This makes the source code cleaner and more convenient when calling the method.

1.2 Overloading by differing the number of parameters

```
public int addDigitalVideoDisc(DigitalVideoDisc dvd1,DigitalVideoDisc dvd2) {
        if (qtyOrdered + 1 >= maxCapacity) {
                System.out.println("Full");
                return 0;
        } else {
```

```
            pickedDVD[qtyOrdered] = dvd1;
            qtyOrdered++;
            System.out.println("Added successfully");

            pickedDVD[qtyOrdered] = dvd2;
            qtyOrdered++;
            System.out.println("Added successfully");

            return 2;
        }
    }
```

2. Passing parameter

2.1. QA

Java is a "Pass by Value" programming language. In Java, when you pass a parameter to a method, the value of the parameter is copied and passed into the method, which means that if you change the value of the parameter inside the method, the value of the variable calling the method is not affected.

2.2 class TestPassingParameter

```
package aims;
public class TestPassingParameter {
        public static void main(String[] args) {
            // TODO Auto-generated method stub
            DigitalVideoDisc jungleDVD = new DigitalVideoDisc("Jungle");
            DigitalVideoDisc cinderellaDVD = new DigitalVideoDisc("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) {
            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);
        }
}
```

Result:

```
Problems  @ Javadoc  Declaration  Console
<terminated> TestPassingParameter [Java Application] C:\Users\ADMIN\.p2\pool\plu
jungle dvd title: Jungle
cinderella dvd title: Cinderella
jungle dvd title: Cinderella
Writable
```

2.3 QA

a. After the call of swap(jungleDVD, cinderellaDVD) why does the title ofthese two objects still remain?

After executing the swap(jungleDVD, cinderellaDVD) method, the titles of the two objects remain the same because in Java, the parameter passed to the method is the value of the object, not the reference to the object. When we change the value of the parameter inside the method (such as swapping o1 and o2), this change does not affect the value of the original objects.

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

After calling changeTitle(jungleDVD, cinderellaDVD.getTitle()), the title of jungleDVD is changed because

in the changeTitle method, we make a direct change on the dvd object (passed into the method) by calling dvd.setTitle(title). This directly affects the original object passed into the method.

2.4 New swap code

```java
public static void swap(DigitalVideoDisc dvd1, DigitalVideoDisc dvd2) {
            DigitalVideoDisc tmp = new DigitalVideoDisc(dvd1.getTitle());

            dvd1.setTitle(dvd2.getTitle());
            dvd2.setTitle(tmp.getTitle());
        }
```

New result

```
Problems  @ Javadoc  Declaration  Console
<terminated> TestPassingParameter [Java Application] C:\Users\ADMIN\.p2\pool\plug
jungle dvd title: Cinderella
cinderella dvd title: Jungle
jungle dvd title: Jungle
Writable
```

3.classifier

```
package aims;

public class DigitalVideoDisc {
	private static int nbDigitalVideoDiscs = 0; //class attribute
	private int id;//instance attribute

	private String title;
	private String category;
	private String director;
	private double cost;
	private int length;

	//constructor method
	public DigitalVideoDisc(String title) {
		super();
	    this.title = title;
	    this.id =  ++nbDigitalVideoDiscs; //update c.a and assign id
	}

	public DigitalVideoDisc(String title, String category, double cost) {
	    this.title = title;
	    this.category = category;
	    this.cost = cost;
	    this.id =  ++nbDigitalVideoDiscs; //update c.a and assign id
	}

	public DigitalVideoDisc(String title, String category, String director,
double cost) {
	    this.title = title;
	    this.category = category;
	    this.director = director;
	    this.cost = cost;
	    this.id =  ++nbDigitalVideoDiscs; //update c.a and assign id
	}

	public DigitalVideoDisc(String title, String category, String director, int
length, double cost) {
	    this.title = title;
	    this.category = category;
	    this.director = director;
	    this.length = length;
	    this.cost = cost;
	    this.id =  ++nbDigitalVideoDiscs; //update c.a and assign id
	}

	// getter
	public String getTitle() {
	    return title;
	}

	public String getCategory() {
	    return category;
	}
```

```java
        public String getDirector() {
            return director;
        }

        public int getLength() {
            return length;
        }

        public double getCost() {
            return cost;
        }
        //setter
        public void setTitle(String title) {
              this.title = title;
        }

        public void setCategory(String category) {
              this.category = category;
        }

        public void setDirector(String director) {
              this.director = director;
        }

        public void setLength(int length) {
              this.length = length;
        }

        public void setCost(double cost) {
              this.cost = cost;
        }
}
```

4.Open the cart class

4.1.code:

```java
public void print() {
        StringBuilder output = new
StringBuilder("***********************CART***********************\r\n"
                    + "Ordered Items:\r\n"
                    + "");
        for (int i=0; i< qtyOrdered;i++) {
              output.append(i+1 +
                          "DVD - [" + pickedDVD[i].getTitle() + "] - " +
                          "[" + pickedDVD[i].getCategory() + "] - " +
                          "[" + pickedDVD[i].getDirector() + "] - " +
                          "[" + pickedDVD[i].getLength() + "]: " +
                          "[" + pickedDVD[i].getCost() + "] $\n");
        }
        output.append("Total cost: ").append(totalCost()).append(" $\n");
        output.append("**************************************************\n");
```

```
        System.out.println(output);
    }
```

4.2.run the code

```
**********************CART**********************
Ordered Items:
1. DVD - [The Lion King] - [Animation] - [Roger Allers] - [87]: [19.950000762939453] $
2. DVD - [Star Wars] - [Science Fiction] - [George Lucas] - [87]: [24.950000762939453] $
3. DVD - [Aladin] - [Animation] - [null] - [0]: [18.989999771118164] $
Total cost: 63.89 $
*************************************************
```

5.Store class:

5.1Store class:

```
package aims;

import java.util.LinkedList;

public class Store {
    private LinkedList<DigitalVideoDisc> itemsInStore = new LinkedList<>();

    private boolean checkDVD(DigitalVideoDisc disc) {
        for (DigitalVideoDisc digitalVideoDisc : itemsInStore) {
            if (digitalVideoDisc.equals(disc)) {
                return true;
            }
        }
        return false;
    }

    public void addDVD(DigitalVideoDisc disc) {
        if (!checkDVD(disc)) {
            itemsInStore.add(disc);
            System.out.println(disc.getTitle() + " has been added to the store!");
        } else {
            System.out.println(disc.getTitle() + " already exists in the store!");
        }
    }

    public void removeDVD(DigitalVideoDisc disc) {
        if (checkDVD(disc)) {
            itemsInStore.remove(disc);
            System.out.println(disc.getTitle() + " has been deleted from the
store!");
        } else {
            System.out.println("There is no " + disc.getTitle() + " in the
store!");
        }
    }
```

```
    public String toString() {
        StringBuilder string = new
StringBuilder("****************STORE****************\nItems in the store:\n");
        if (itemsInStore.isEmpty()) {
            string.append("There is no DVD in the store!\n");
        } else {
            for (DigitalVideoDisc dvd : itemsInStore) {
                string.append(dvd.getTitle())
                      .append(" - ")
                      .append(dvd.getCost())
                      .append(" $\n");
            }
        }
        string.append("****************************************");
        return string.toString();
    }
}
```

5.2 StoreTest:

```
package aims;
public class StoreTest {
    public static void main(String[] args) {
        Store store = new Store();

        DigitalVideoDisc dvd1 = new DigitalVideoDisc("The Lion King",
"Animation", "Roger Allers", 87, 19.95f);
        DigitalVideoDisc dvd2 = new DigitalVideoDisc("Star Wars", "Science
Fiction", "George Lucas", 87, 24.95f);
        DigitalVideoDisc dvd3 = new DigitalVideoDisc("Aladin", "Animation",
18.99f);
        //addDVD
        store.addDVD(dvd1);
        store.addDVD(dvd2);
        store.addDVD(dvd3);

        //addexistedDVD
        store.addDVD(dvd2);

        //print
        System.out.println(store);

        //delete
        store.removeDVD(dvd2);

        //delete_inexistent_DVD
        store.removeDVD(dvd2);

        //print1moretime
        System.out.println(store);
    }
}
```

5.3.result

```
The Lion King has been added to the store!
Star Wars has been added to the store!
Aladin has been added to the store!
Star Wars already exists in the store!
****************STORE****************
Items in the store:
The Lion King - 19.950000762939453 $
Star Wars - 24.950000762939453 $
Aladin - 18.989999771118164 $
*************************************
Star Wars has been deleted from the store!
There is no Star Wars in the store!
****************STORE****************
Items in the store:
The Lion King - 19.950000762939453 $
Aladin - 18.989999771118164 $
*************************************
```

6. String, StringBuilder, StringBuffer

6.1 Class ConcatenationInLoops

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

public class ConcatenationInLoops {
    private static final int ITERATIONS = 100_000;

    public static void main(String[] args) {
        long startTime = System.currentTimeMillis();
        String str = "";
        for (int i = 0; i < ITERATIONS; i++) {
            str += i;
        }
        long endTime = System.currentTimeMillis();
        System.out.println("Time taken by String: " + (endTime - startTime) + "
ms");

        startTime = System.currentTimeMillis();
        StringBuffer stringBuffer = new StringBuffer();
        for (int i = 0; i < ITERATIONS; i++) {
            stringBuffer.append(i);
        }
        endTime = System.currentTimeMillis();
        System.out.println("Time taken by StringBuffer: " + (endTime - startTime) +
" ms");

        startTime = System.currentTimeMillis();
        StringBuilder stringBuilder = new StringBuilder();
        for (int i = 0; i < ITERATIONS; i++) {
            stringBuilder.append(i);
        }
        endTime = System.currentTimeMillis();
        System.out.println("Time taken by StringBuilder: " + (endTime - startTime)
+ " ms");
```

```
    }
}
```

6.2 Class NoGarbage

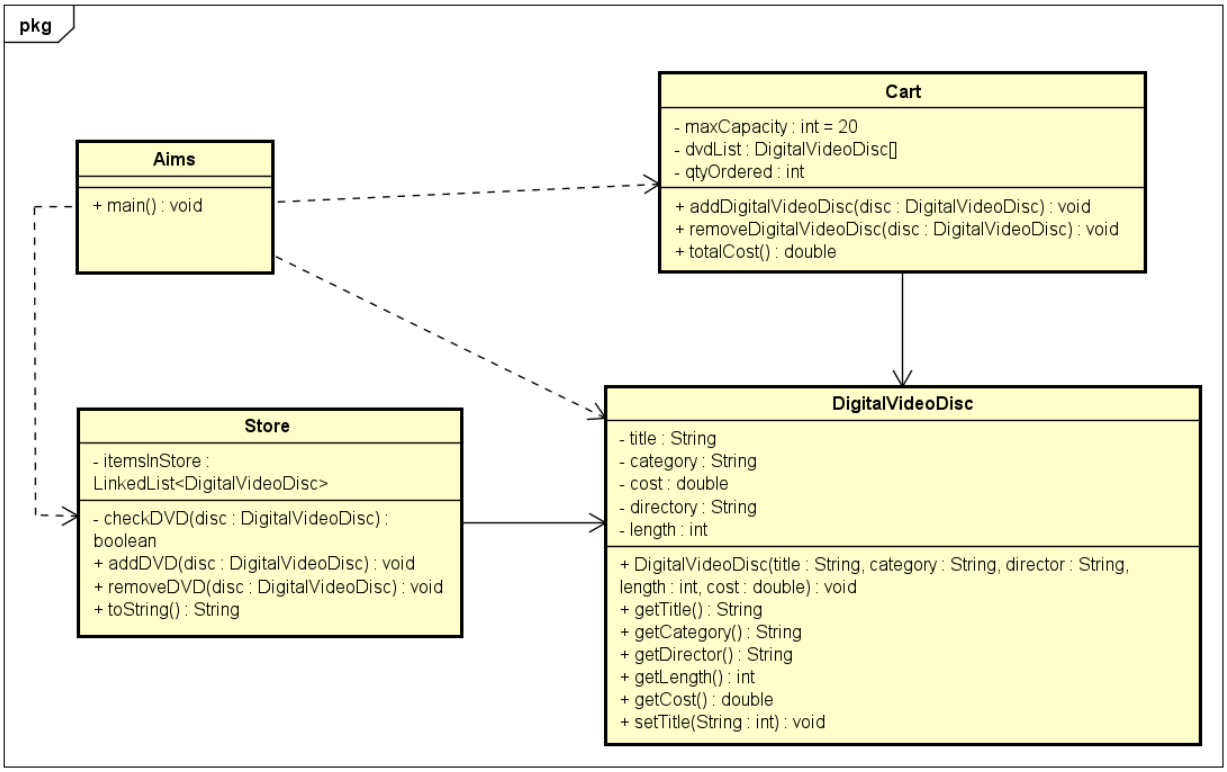
```
package hust.soict.globalict.garbage;

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

public class NoGarbage {
    public static void main(String[] args) {
        try {
            String filePath = "test.txt"; // Đường dẫn đến tệp lớn
            byte[] bytes = Files.readAllBytes(Paths.get(filePath));
            String content = new String(bytes);

            StringBuilder result = new StringBuilder();
            long startTime = System.currentTimeMillis();
            for (int i = 0; i < 1000; i++) {
                result.append(content);
            }
            long endTime = System.currentTimeMillis();
            System.out.println("Time taken with StringBuilder: " + (endTime -
startTime) + " ms");

        } catch (Exception e) {
            System.out.println("Error: " + e.getMessage());
        }
    }
}
```

6.3 Class GarbageCreator

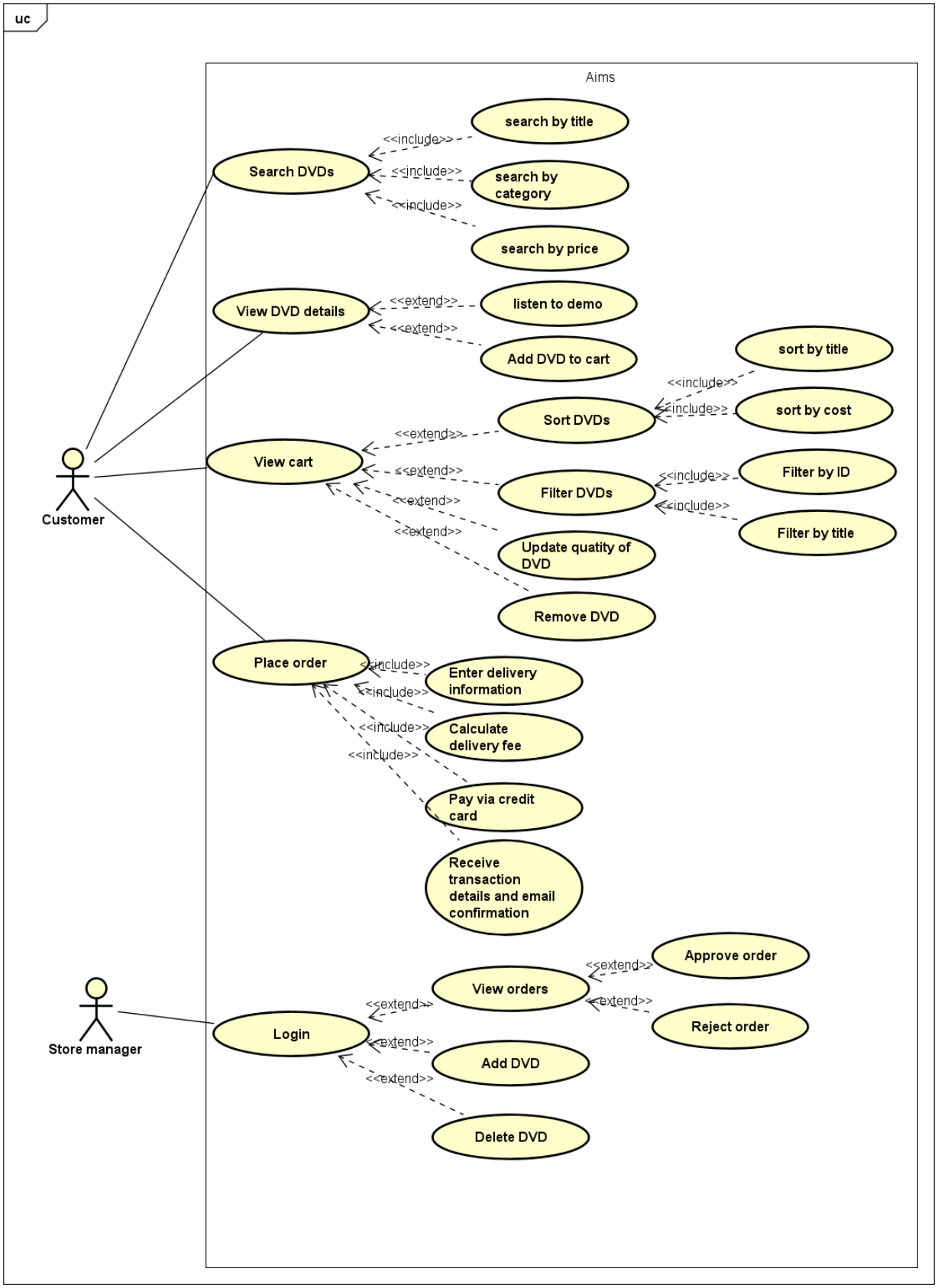
```
package hust.soict.globalict.garbage;

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

public class GarbageCreator {
    public static void main(String[] args) {
        try {
            String filePath = "test.txt"; // Đường dẫn đến tệp lớn
            byte[] bytes = Files.readAllBytes(Paths.get(filePath));
            String content = new String(bytes);

            String result = "";
            long startTime = System.currentTimeMillis();
            for (int i = 0; i < 1000; i++) {
                result += content;
            }
            long endTime = System.currentTimeMillis();
            System.out.println("Time taken with String: " + (endTime - startTime) +
" ms");

        } catch (Exception e) {
```

```
            System.out.println("Error: " + e.getMessage());
        }
    }
}
```

7. Class Diagram



8. Usecase diagram


uc
Aims
Customer
Store manager
Search DVDs
search by title
search by category
search by price
<<include>>
View DVD details
listen to demo
Add DVD to cart
<<extend>>
View cart
Sort DVDs
sort by title
sort by cost
Filter DVDs
Filter by ID
Filter by title
Update quatity of DVD
Remove DVD
Place order
Enter delivery information
Calculate delivery fee
Pay via credit card
Receive transaction details and email confirmation
Login
View orders
Approve order
Reject order
Add DVD
Delete DVD

9. Debugging

I have successfully applied the debugging method.