Lab02 report: problem modeling and encapsulation

Student: Lê Văn Pháp

StudentID: 20226118

Contents

1. Problem

Aims project

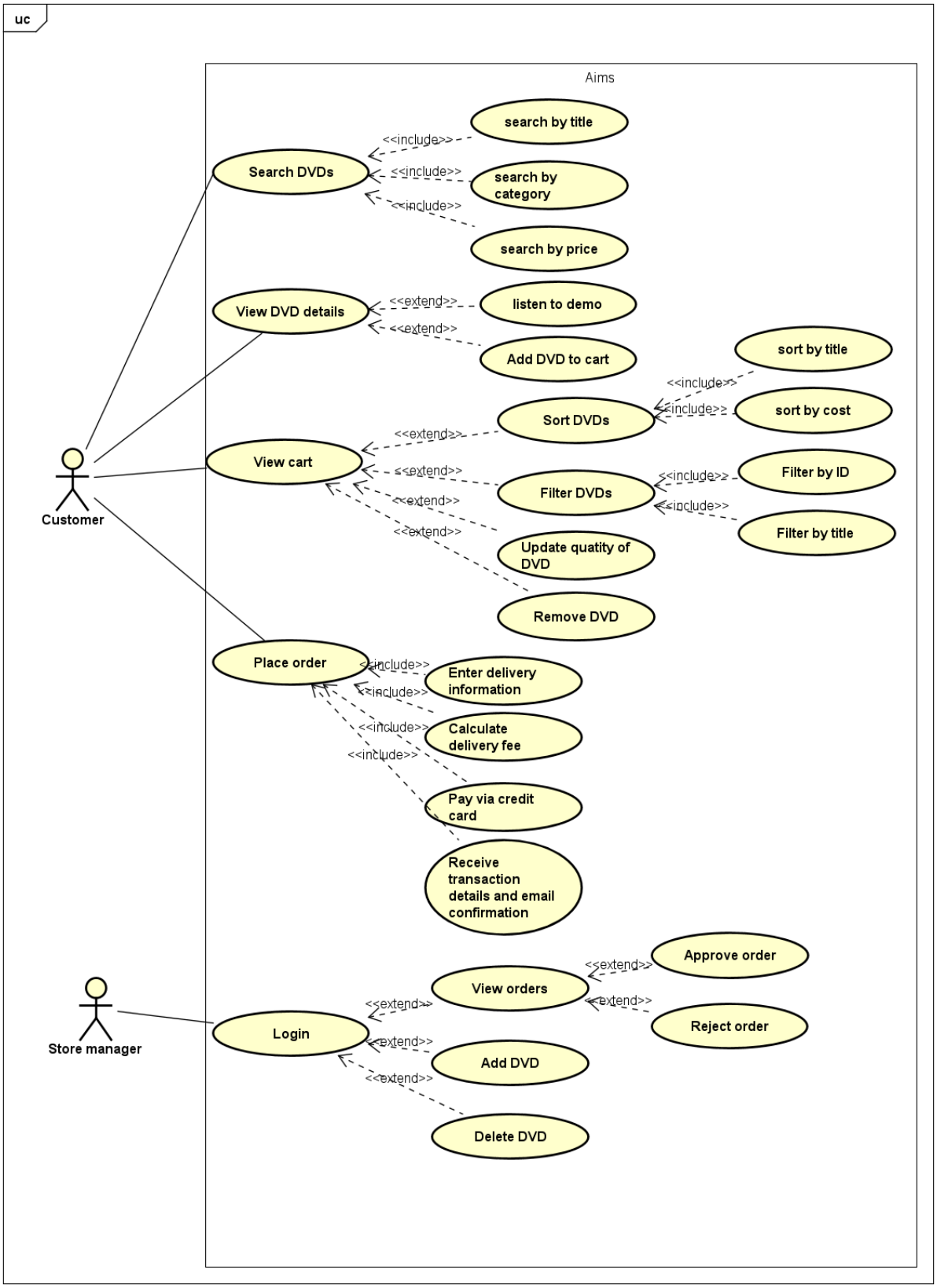
a For customers:

- They can browse or search for DVDs
- View detail of DVDs or listen to demo parts
- Access and edit their cart
- Place order

b For store manager:

- they have to log in first
- Check the list of pending orders and approve/reject them
- Delete or add new DVDs

2. Use-case diagram



3. Class diagram

3.1Classes

3.1.1 Aims

- Attributes: none
- Methods:
+ main() : void

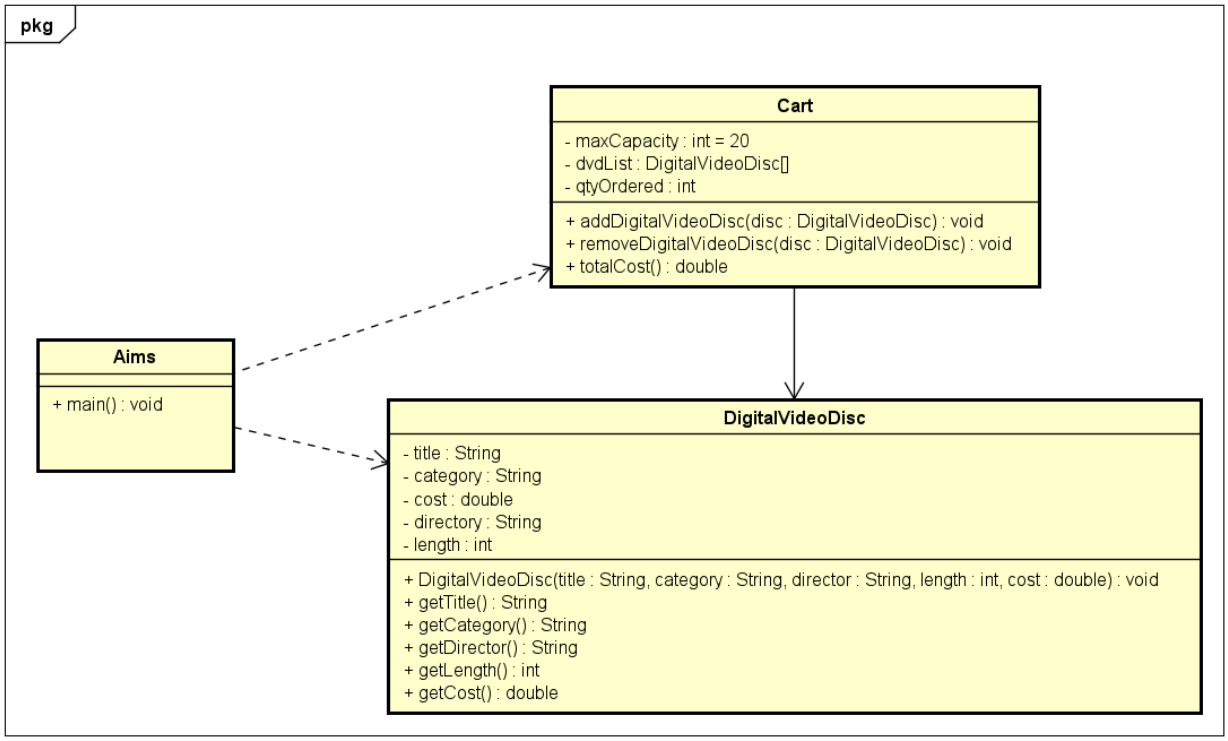
3.1.2 Cart

- Attributes:
+ maxCapacity: int =20
+ dvdList: DigitalVideoDisc[]
+ qtyOrdered: int
- Methods:
+ addDigitalVideoDisc(disc : DigitalVideoDisc)
+ removeDigitalVideoDisc(disc : DigitalVideoDisc)
+ totalCost(): double

3.1.3 DigitalVideoDisc

- Attributes:
+ title: String
+ category: String
+ cost: double
+ director: String
+ length: int
- Methods:
+ DigitalVideoDisc(title : String, category : String, director : String, length : int, cost : double) : void
+ getTitle() : String
+ getCategory() : String
+ getDirector() : String
+ getLength() : int
+ getCost() : double

3.2 Class diagram



4. Source code

4.1 DigitalVideoDisc.java

```
package aims;

public class DigitalVideoDisc {

	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;
	}

	public DigitalVideoDisc(String title, String category, double cost) {
	    this.title = title;
	    this.category = category;
	    this.cost = cost;
	}

	public DigitalVideoDisc(String title, String category, String director, double
cost) {
```

```
            this.title = title;
            this.category = category;
            this.director = director;
            this.cost = cost;
        }

        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;
        }

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

        public String getCategory() {
            return category;
        }

        public String getDirector() {
            return director;
        }

        public int getLength() {
            return length;
        }

        public double getCost() {
            return cost;
        }
}
```

4.2 Cart.java

```
package aims;

public class Cart {
    private static final int maxCapacity = 20;
    private DigitalVideoDisc[] dvdList = new DigitalVideoDisc[maxCapacity];

    private int qtyOrdered = 0;

    public void addDigitalVideoDisc(DigitalVideoDisc disc) {
        if (qtyOrdered < maxCapacity) {
            dvdList[qtyOrdered] = disc;
            qtyOrdered++;
```

```
            System.out.println("The disc has been added.");
        } else {
            System.out.println("The cart is almost full.");
        }
    }

    public void removeDigitalVideoDisc(DigitalVideoDisc disc) {
        boolean found = false;
        for (int i = 0; i < qtyOrdered; i++) {
            if (dvdList[i] == disc) {
                dvdList[i] = dvdList[qtyOrdered - 1];
                dvdList[qtyOrdered - 1] = null;
                qtyOrdered--;
                found = true;
                System.out.println("The disc has been removed.");
                break;
            }
        }
        if (!found) {
            System.out.println("The disc is not found in the cart.");
        }
    }

    public float totalCost() {
        float total = 0.00f;
        for (int i = 0; i < qtyOrdered; i++) {
            total += dvdList[i].getCost();
        }
        return total;
    }
}
```

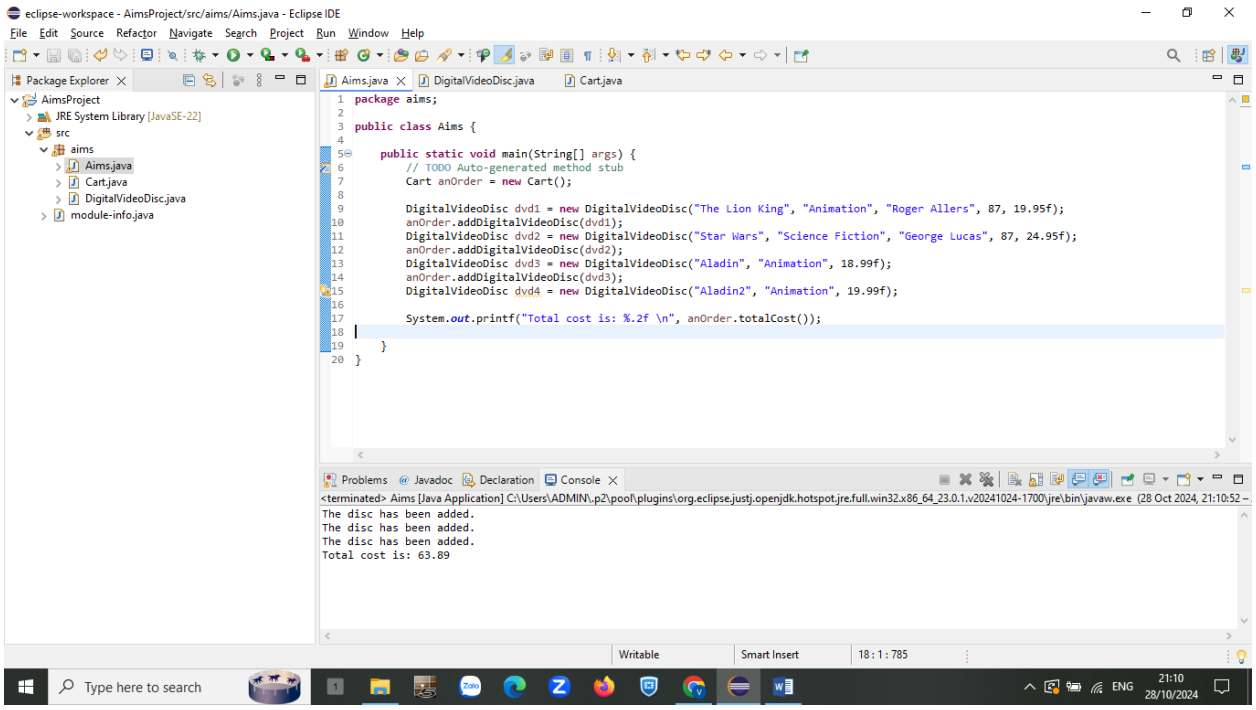
4.3 Aims.java

```
package aims;

public class Aims {

       public static void main(String[] args) {
             // TODO Auto-generated method stub
        Cart anOrder = new Cart();

        DigitalVideoDisc dvd1 = new DigitalVideoDisc("The Lion King", "Animation",
"Roger Allers", 87, 19.95f);
        anOrder.addDigitalVideoDisc(dvd1);
        DigitalVideoDisc dvd2 = new DigitalVideoDisc("Star Wars", "Science Fiction",
"George Lucas", 87, 24.95f);
        anOrder.addDigitalVideoDisc(dvd2);
        DigitalVideoDisc dvd3 = new DigitalVideoDisc("Aladin", "Animation", 18.99f);
        anOrder.addDigitalVideoDisc(dvd3);
        DigitalVideoDisc dvd4 = new DigitalVideoDisc("Aladin2", "Animation", 19.99f);

        System.out.printf("Total cost is: %.2f \n", anOrder.totalCost());
```

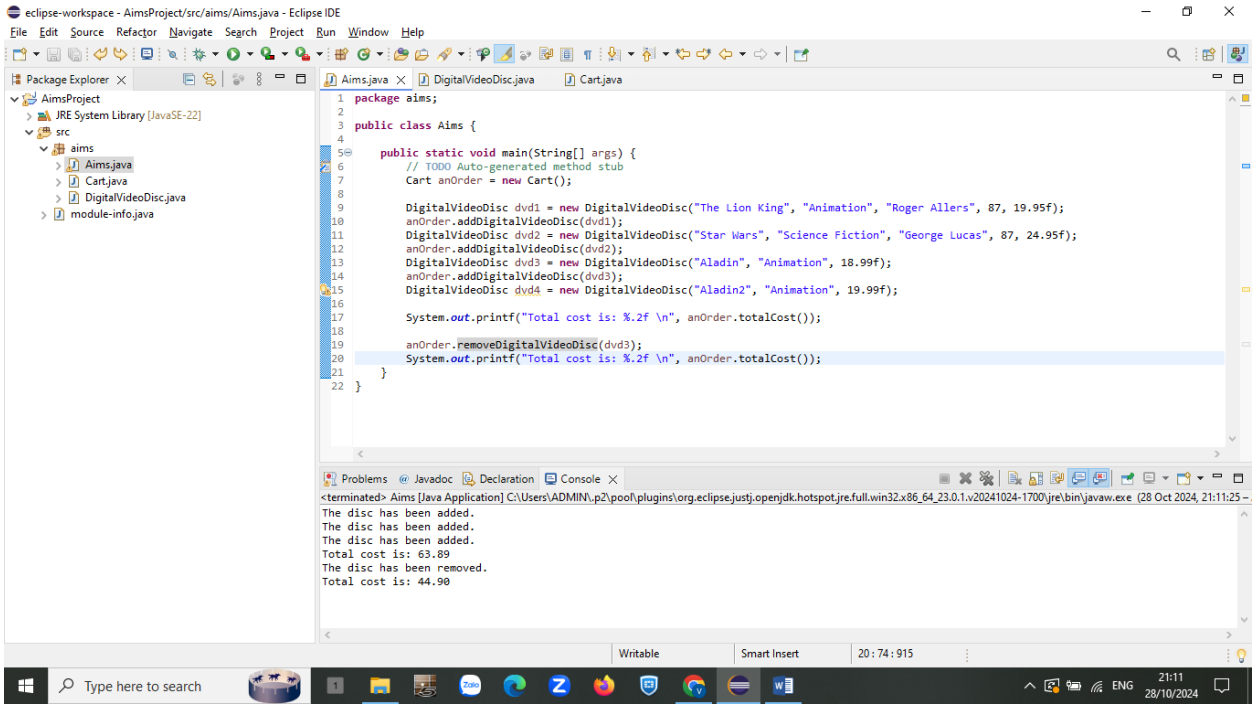
```
        anOrder.removeDigitalVideoDisc(dvd4);
        System.out.printf("Total cost is: %.2f \n", anOrder.totalCost());
    }
}
```

5. Demo

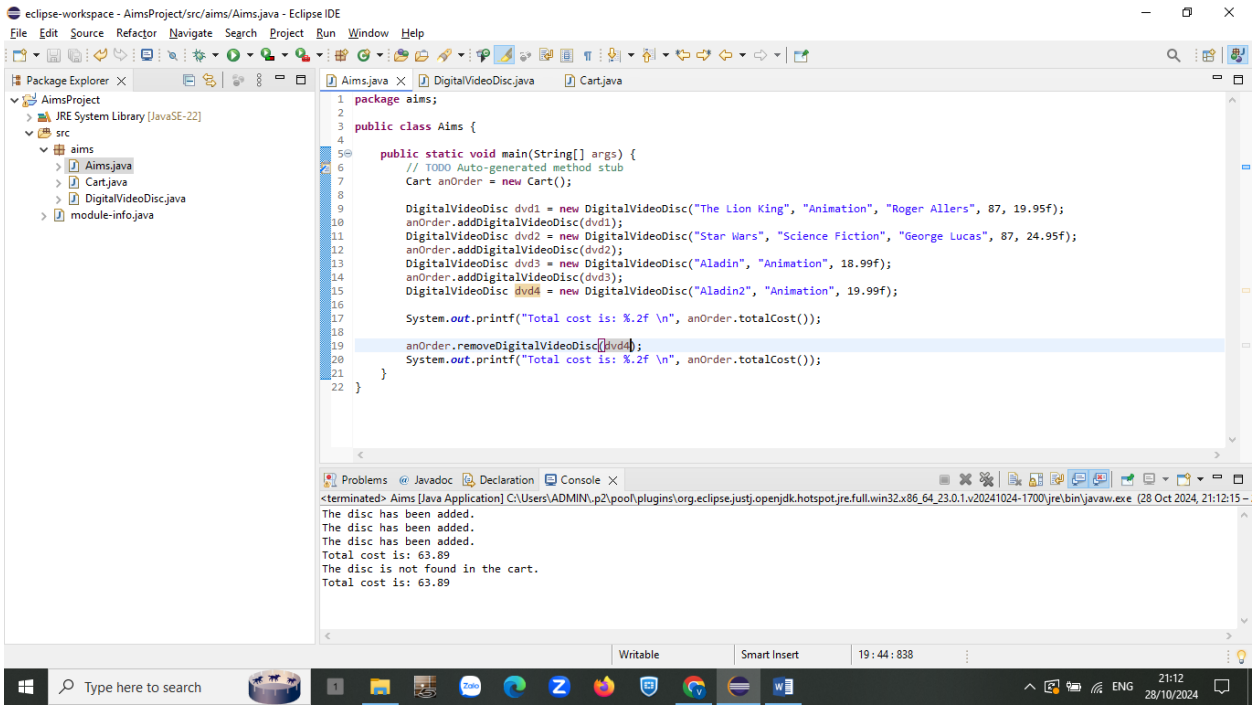
5.1 Add 3 dics and calculate the total cost



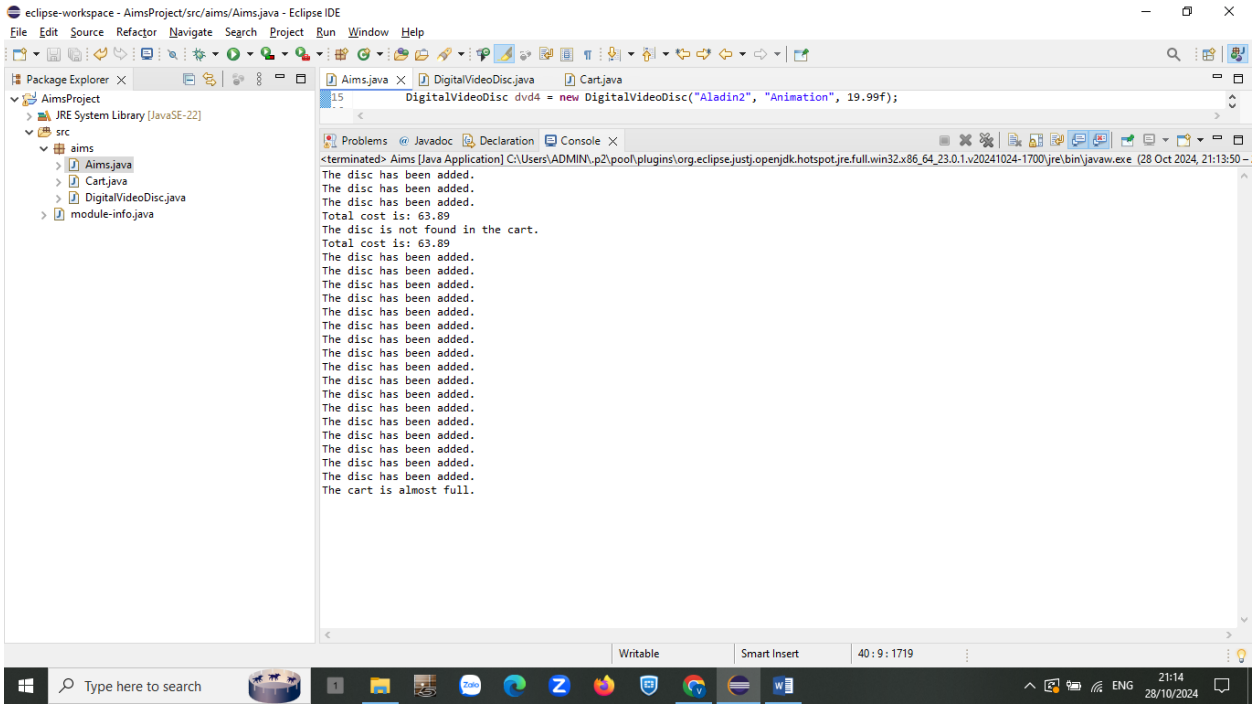
5.2 Add 3 dics then remove dics 3



5.3 Add 3 dics: 1,2,3 and remove dics 4:



5.4 Add 20 dics and at the 21st, there is an announcement

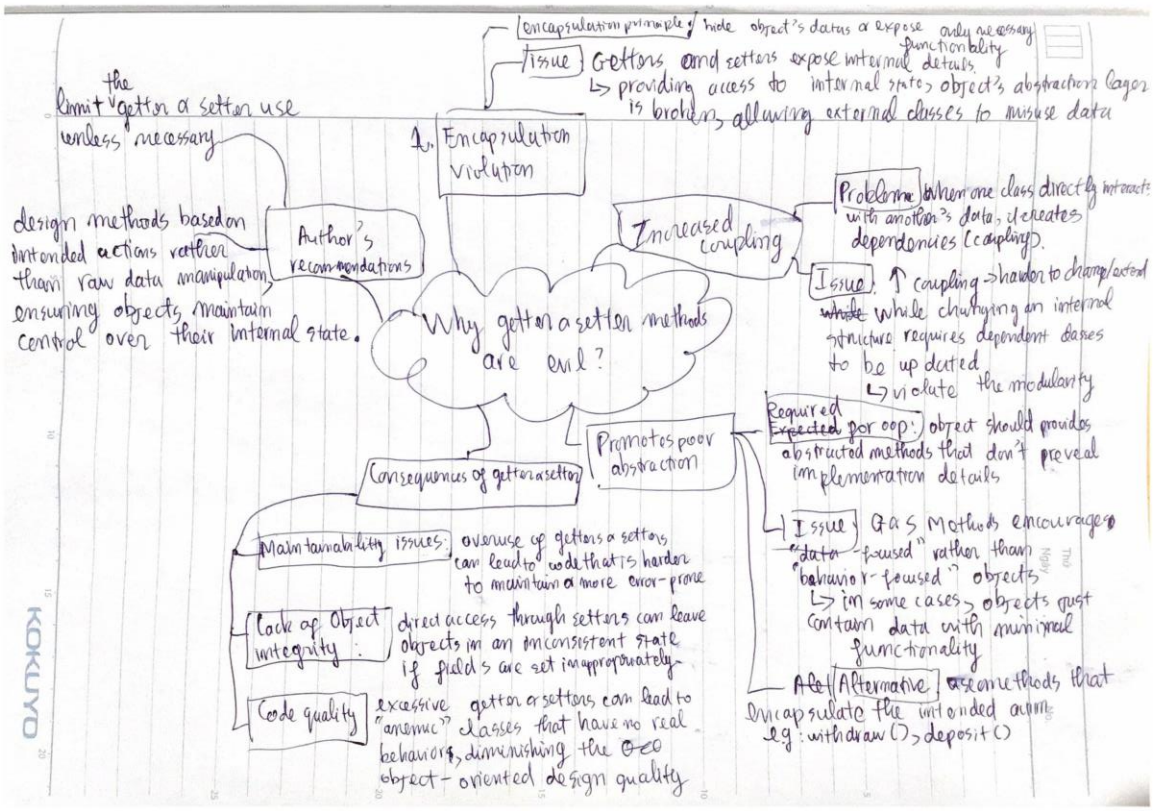


6. Reading assignment

When should accessor methods be used?

Read the following article and find the best possible answer to the above question: Holub, Allen. "Why getter and setter methods are evil" *JavaWorld*, 5 Sep. 2003, https://www.infoworld.com/article/2073723/why-getter-and-setter-methods-are-evil.html

You should expand your research to other sources as well. For the response, give a summary of your findings in the form of a mindmap. You can draw this mind map by hand and take a picture of your work or use any online tools. In both cases, the accepted format for the image file is one of the following: .png, .jpg, .jpeg and .pdf.



7. Answer the question

- If you create a constructor method to build a DVD by title then create a constructor method to build a DVD by category. Does JAVA allow you to do this?

Answer:

JAVA allows us to create multiple constructor methods for a class with differents parameters, as known as method overloading. Each constructor must have different parameter lists (unique signature)

Example:

```
public class DigitalVideoDics{
	private String title;
	private String category;

	// constructor by title
	public DigitalVideoDics(String title) {
```

```
		this.title = title;
	}

	//constructor by category
	public DigitalVideoDics(String category, String title) {
		this.category = category;
		this.title = title;
	}
}
```