Object Oriented Programming Lab 3
Dinh Dinh Hai Viet – 20225683

1. Screenshot of any new written code

Section 2.1: Method `addDigitalVideoDisc(DigitalVideoDisc[] dvdList)` in `Cart` class:

```
public void addDigitalVideoDisc(DigitalVideoDisc[] dvdList) {
		for(DigitalVideoDisc dvd : dvdList) {
			if(qtyOrdered < MAX_NUMBERS_ORDERED) {
				itemOrdered[qtyOrdered] = dvd;
				qtyOrdered++;
				System.out.println("The DVD has been added.");
			}
			else {
				System.out.println("The cart is almost full!");
			}
		}
		System.out.println();
	}
```

Section 2.2: Method `addDigitalVideoDisc(DigitalVideoDisc dvd1, DigitalVideoDisc dvd2)` in `Cart` class

```
public void addDigitalVideoDisc(DigitalVideoDisc dvd1, DigitalVideoDisc dvd2) {
		if(qtyOrdered < MAX_NUMBERS_ORDERED - 2) {
			itemOrdered[qtyOrdered] = dvd1;
			qtyOrdered++;
			itemOrdered[qtyOrdered] = dvd2;
			qtyOrdered++;
			System.out.println("The two DVDs have been added!");
		}
		else if(qtyOrdered == MAX_NUMBERS_ORDERED - 2) {
			itemOrdered[qtyOrdered] = dvd1;
			qtyOrdered++;
			System.out.println("The first DVD has been added.");
			System.out.println("The cart is almost full! Cannot add the
second DVD to cart.");
			System.out.println();
		}
		else {
			System.out.println("The cart is almost full!");
			System.out.println();
		}
	}
```

Section 3: Method `newSwap(DigitalVideoDisc dvd1, DigitalVideoDisc dvd2)` in `TestPassingParameter` class

```
public static void newSwap(DigitalVideoDisc dvd1, DigitalVideoDisc dvd2) {
            String dvd1OldTitle = dvd1.getTitle();
            String dvd1OldCategory = dvd1.getCategory();
            String dvd1OldDirector = dvd1.getCategory();
            int dvd1OldLength = dvd1.getId();
            double dvd1OldCost = dvd1.getCost();
            int dvd1OldId = dvd1.getId();

            dvd1.setTitle(dvd2.getTitle());
            dvd1.setCategory(dvd2.getCategory());
            dvd1.setDirector(dvd2.getDirector());
            dvd1.setLength(dvd2.getLength());
            dvd1.setCost(dvd2.getCost());
            dvd1.setId(dvd2.getId());

            dvd2.setTitle(dvd1OldTitle);
            dvd2.setCategory(dvd1OldCategory);
            dvd2.setDirector(dvd1OldDirector);
            dvd2.setLength(dvd1OldLength);
            dvd2.setCost(dvd1OldCost);
            dvd2.setId(dvd1OldId);
      }
```

Result in console IO after running the TestPassingParamter class:

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

Section 5: Create new instance attribute `id` and static attributs `nbDigitalVideoDisc` for class `DigitalVideoDisc` to automatically update `id` when new object of class `DigitalVideoDisc` is created.

```
      private int id;
      private static int nbDigitalVideoDiscs = 0;
      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;
      }
```

Section 6:

Method `toString()` in `DigitalVideoDisc` class:

```
public String toString() {
	return "DVD" + this.id + " - " + "Title: " + this.title + " - Category: " +
		this.category + " - Director: " + this.director + " - Length: " +
		this.length + " - Price: " + this.cost + "$";
	}
```

Method `print()` in `Cart` class:

```
//Method to print the cart
	public void print() {
		System.out.println("******************CART***********************");
		System.out.println("Ordered Items");
		for(int i = 0; i < qtyOrdered; i++) {
			System.out.println((i + 1) + ". " + itemOrdered[i].toString());
		}
		System.out.println("Total cost: " + totalCost() + "$");
		System.out.println("**********************************************");
	}
```

Method `isMatch(String keywords)` in `DigitalVideoDisc` class:

```
public boolean isMatch(String keywords) {
		String[] splitKeywords = keywords.toLowerCase().split("\\s+");
		String toLowerTitle = this.title.toLowerCase();

		for(String s : splitKeywords) {
			if(toLowerTitle.contains(s)) {
				return true;
			}
		}
		return false;
	}
```

Method `search(String keywords)` in `Cart` class:

```
//Method to search DVD by title
	public void search(String keywords) {
		int matchItem = 0;
		for(int i = 0; i < qtyOrdered; i++) {
			if(itemOrdered[i].isMatch(keywords)) {
				matchItem++;
				System.out.println(itemOrdered[i].toString());
			}
		}
		if(matchItem == 0) {
			System.out.println("No item found!");
		}
	}
```

Method search(int ID) in Cart class:

```
public void search(int ID) {
            int matchItem = 0;
            for(int i = 0; i < qtyOrdered; i++) {
                  if(itemOrdered[i].getId() == ID) {
                        matchItem++;
                        System.out.println(itemOrdered[i].toString());
                  }
            }
            if(matchItem == 0) {
                  System.out.println("No item found!");
            }
      }
```

Class CartTest to test all the written method:

```
package hust.soict.hedspi.aims.cart;
import hust.soict.hedspi.aims.disc.*;

public class CartTest {
      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("The Lion King",
                        "Animation", "Roger Allers", 87, 19.95);
            cart.addDigitalVideoDisc(dvd1);
            DigitalVideoDisc dvd2 = new DigitalVideoDisc("Star Wars",
                        "Science Fiction", "Geogre Lucas", 87, 24.95);
            DigitalVideoDisc dvd3 = new DigitalVideoDisc("Aladin",
                        "Animation", 18.99);
            cart.addDigitalVideoDisc(dvd2, dvd3);
            DigitalVideoDisc dvd4 = new DigitalVideoDisc("Frozen",
                        "Animation", 24.99);
            DigitalVideoDisc dvd5 = new DigitalVideoDisc("The Lion King III",
                        "Animation", 30.55);
            DigitalVideoDisc[] list = {dvd4, dvd5};
            cart.addDigitalVideoDisc(list);

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

            //Test the search method (by title)
            cart.search("lion"); //Two items found
            cart.search("snow"); //No item found

            //Test the search method (by ID)
            cart.search(1); //One item found
            cart.search(9); //No item found
      }
}
```

Result in console IO after running the CartTest class:

```
The disc has been added.
The two DVDs have been added!
The DVD has been added.
The DVD has been added.

**********CART**********
Ordered Items
1. DVD1 - Title: The Lion King - Category: Animation - Director: Roger Allers - Length: 87 - Price: 19.95$
2. DVD2 - Title: Star Wars - Category: Science Fiction - Director: Geogre Lucas - Length: 87 - Price: 24.95$
3. DVD3 - Title: Aladin - Category: Animation - Director: null - Length: 0 - Price: 18.99$
4. DVD4 - Title: Frozen - Category: Animation - Director: null - Length: 0 - Price: 24.99$
5. DVD5 - Title: The Lion King III - Category: Animation - Director: null - Length: 0 - Price: 30.55$
Total cost: 119.42999999999999$
**************************************************
DVD1 - Title: The Lion King - Category: Animation - Director: Roger Allers - Length: 87 - Price: 19.95$
DVD5 - Title: The Lion King III - Category: Animation - Director: null - Length: 0 - Price: 30.55$
No item found!
DVD1 - Title: The Lion King - Category: Animation - Director: Roger Allers - Length: 87 - Price: 19.95$
No item found!
```

Section 7:

Class Store:

```
package hust.soict.hedspi.aims.store;
import hust.soict.hedspi.aims.disc.*;

public class Store {
	private DigitalVideoDisc itemInStore[] = new DigitalVideoDisc[20];
	private int qtyItem;

	public void addDVD(DigitalVideoDisc dvd) {
		itemInStore[qtyItem] = dvd;
		qtyItem++;
		System.out.println("The dvd has been added.");
		System.out.println();
	}

	public void removeDVD(DigitalVideoDisc dvd) {
		int index = -1;
		if(qtyItem < 0) {
			System.out.println("The store is empty.");
			System.out.println();
		}
		else {
			for(int i = 0; i < qtyItem; i++) {
				if(itemInStore[i].equals(dvd)) {
					index = i;
					break;
				}
			}
			if(index == -1) {
				System.out.println("The dvd hasn't in the store yet!");
				System.out.println();
			}
			else {
				for(int i = 0; i < qtyItem - 1; i++) {
					itemInStore[i] = itemInStore[i + 1];
					itemInStore[qtyItem - 1] = null;
					qtyItem--;
				}
				System.out.println("Remove successfully!");
				System.out.println();
			}
		}
	}
}
```

Class StoreTest:

```
package hust.soict.hedspi.aims.store;
import hust.soict.hedspi.aims.disc.*;

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

        DigitalVideoDisc dvd1 = new DigitalVideoDisc("The Lion King");
        DigitalVideoDisc dvd2 = new DigitalVideoDisc("Star Wars");
        DigitalVideoDisc dvd3 = new DigitalVideoDisc("Aladdin");

        // Thêm DVD vào store
        store.addDVD(dvd1);
        store.addDVD(dvd2);
        store.addDVD(dvd3);

        // Xóa DVD khỏi store
        store.removeDVD(dvd2); // Xóa "Star Wars"
        store.removeDVD(new DigitalVideoDisc("Avatar")); // Xóa DVD không tồn tại
      }
}
```

Result in console IO after running the StoreTest class:

```
The dvd has been added.

The dvd has been added.

The dvd has been added.

Remove successfully!

The dvd hasn't in the store yet!
```

Section 9:

Class GarbageCreator:

```
package hust.soict.hedspi.garbage;
import java.nio.file.Files;
import java.nio.file.Paths;
import java.io.IOException;

public class GarbageCreator {
    public static void main(String[] args) {
        String filename = "D:/TTUD/Laboratory/Lab1/Exercise1.exe";
        byte[] inputBytes = {0};
        long startTime, endTime;

        try {
            inputBytes = Files.readAllBytes(Paths.get(filename));
            startTime = System.currentTimeMillis();
            String outputString = "";
            for(byte b : inputBytes) {
                outputString += (char) b;
            }
            endTime = System.currentTimeMillis();
            System.out.println((endTime - startTime) + " ms");
        } catch (IOException e) {
            System.out.println("Lỗi khi đọc file: " + e.getMessage());
        }
    }
}
```

Result: This code can not run successfully because it should let the program hangs or even stop working when we create too much "garbage".

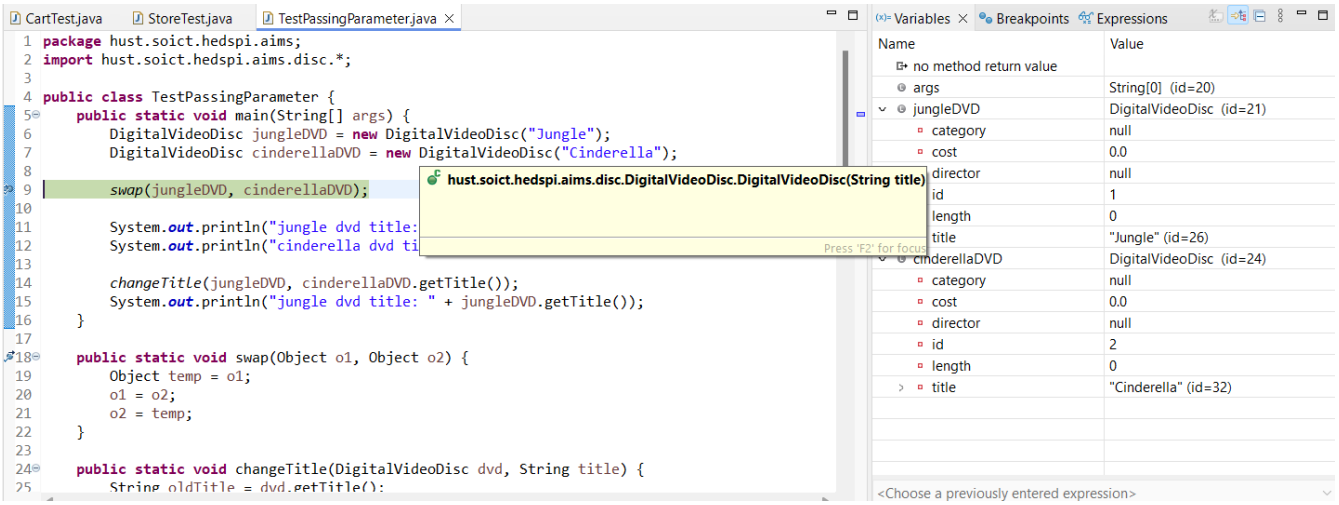
Class NoGarbage:

```
package hust.soict.hedspi.garbage;
import java.io.IOException;
import java.nio.file.Files;
import java.nio.file.Paths;

public class NoGarbage {
      public static void main(String[] args) {
        String filename = "D:/TTUD/Laboratory/Lab1/Exercise1.exe";
        byte[] inputBytes = {0};
        long startTime, endTime;

        try {
            inputBytes = Files.readAllBytes(Paths.get(filename));
            startTime = System.currentTimeMillis();
            StringBuffer outputStringBuffer = new StringBuffer();
            for(byte b : inputBytes) {
                outputStringBuffer.append((char)b);
            }
            endTime = System.currentTimeMillis();
            System.out.println((endTime - startTime) + " ms");
        } catch (IOException e) {
            System.out.println("Lỗi khi đọc file: " + e.getMessage());
        }
    }
}
```

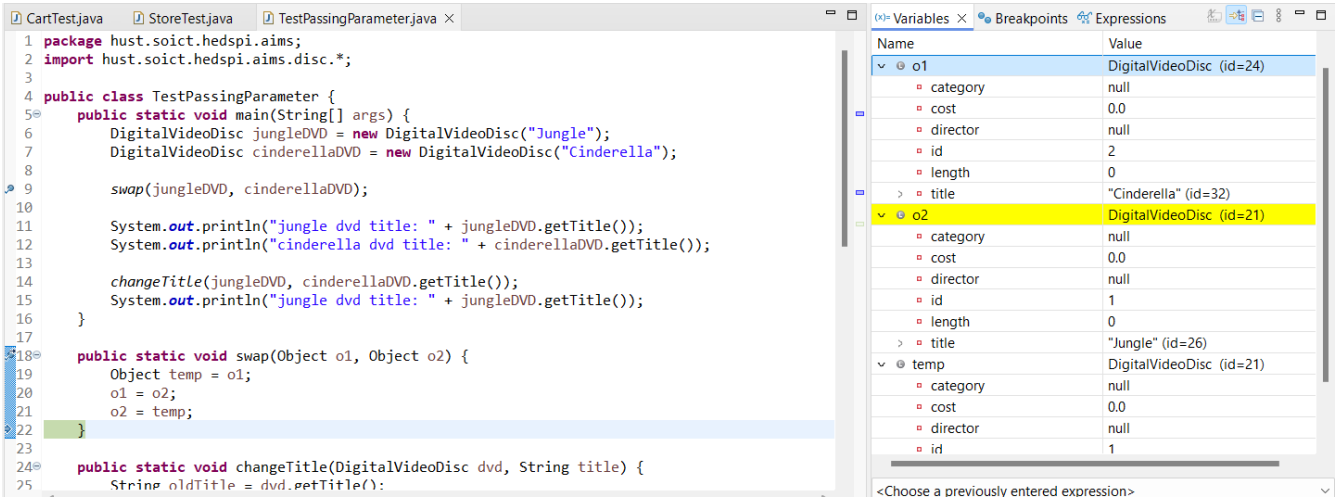
Result: The console will print out "63ms", which is the limited time to read the file Exercise1.exe

2. Code debugging and result

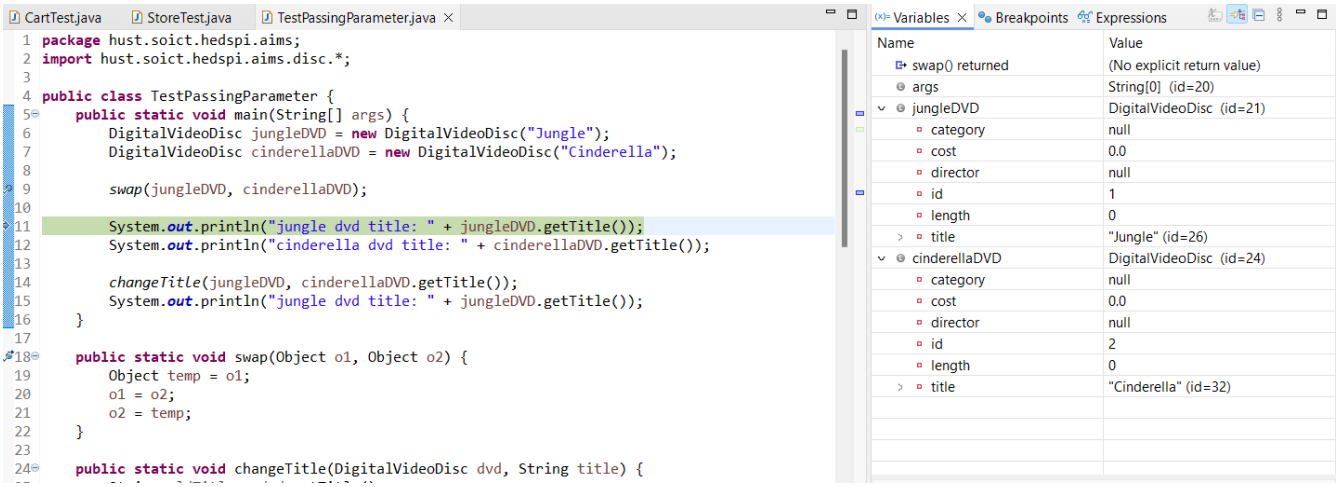
Before pressing "Step Into":



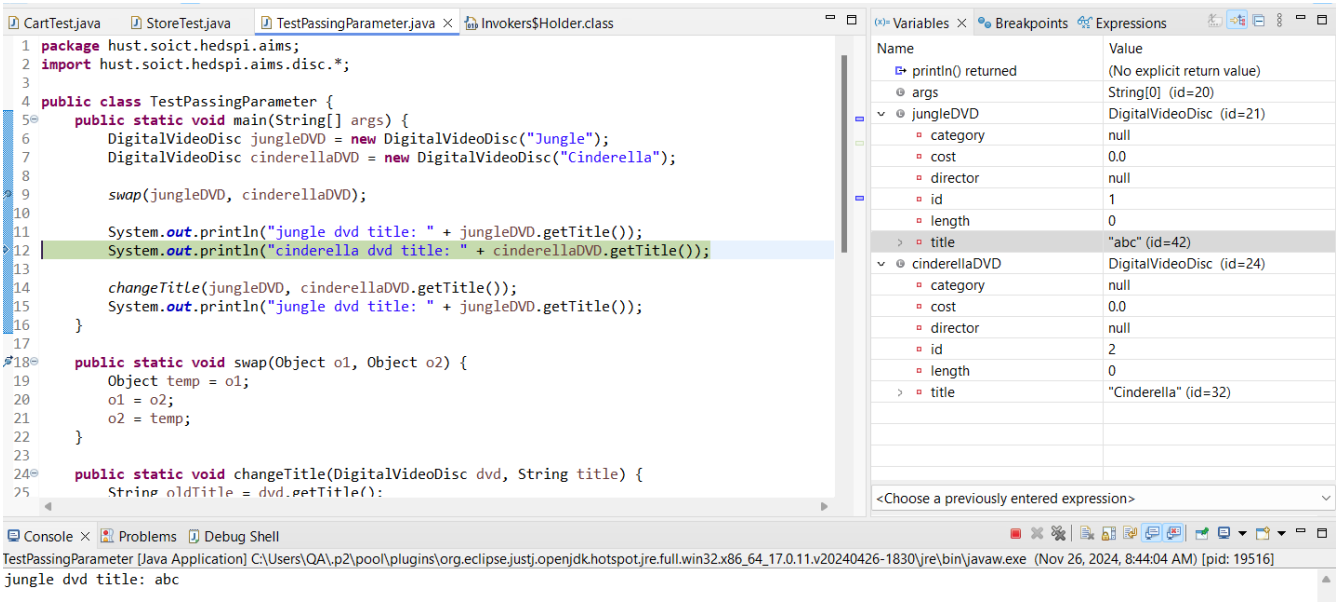
After pressing "Step Into" till the end of method `Swap()`:



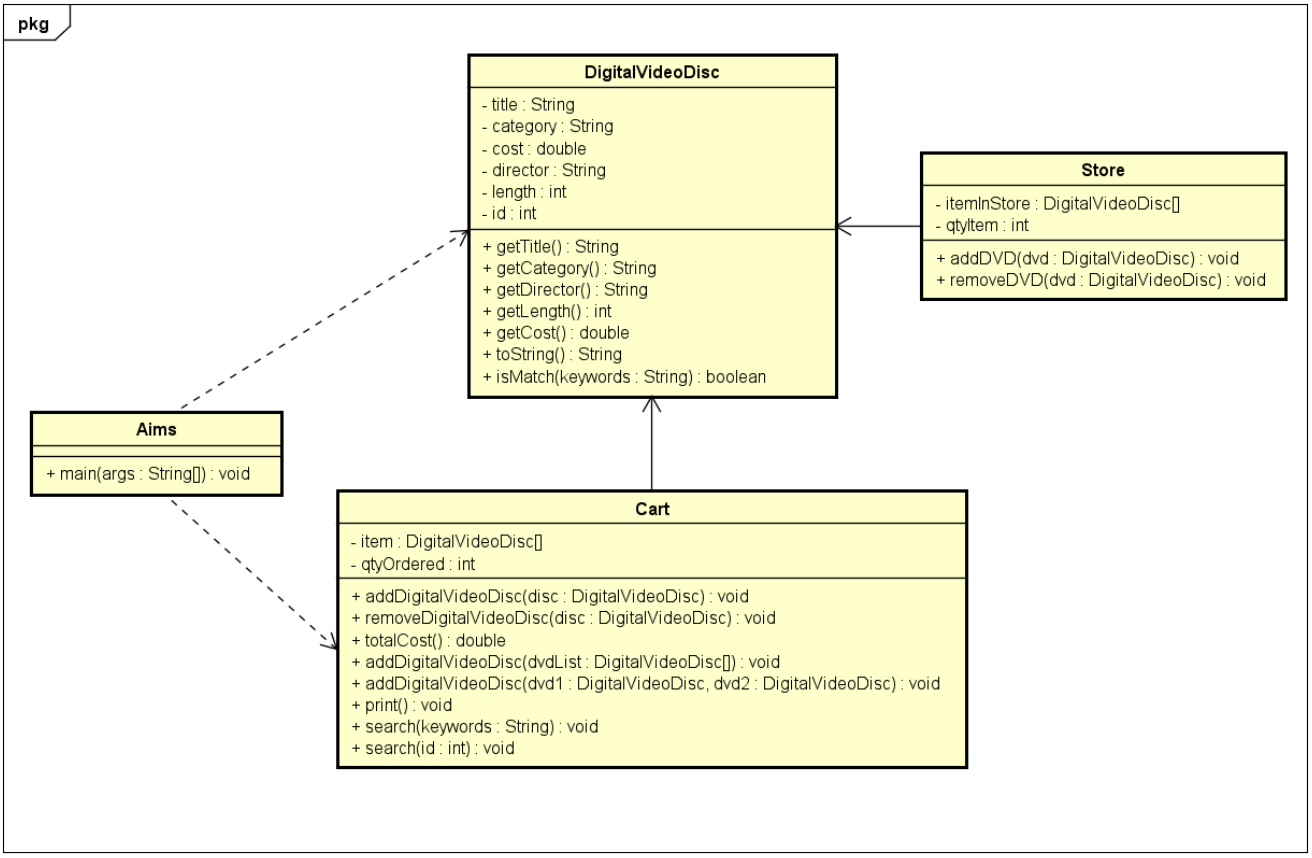
After pressing "Step Into" and comming back to the next command. The value of `jungleDVD` and `cinderellaDVD` remain unchanged:

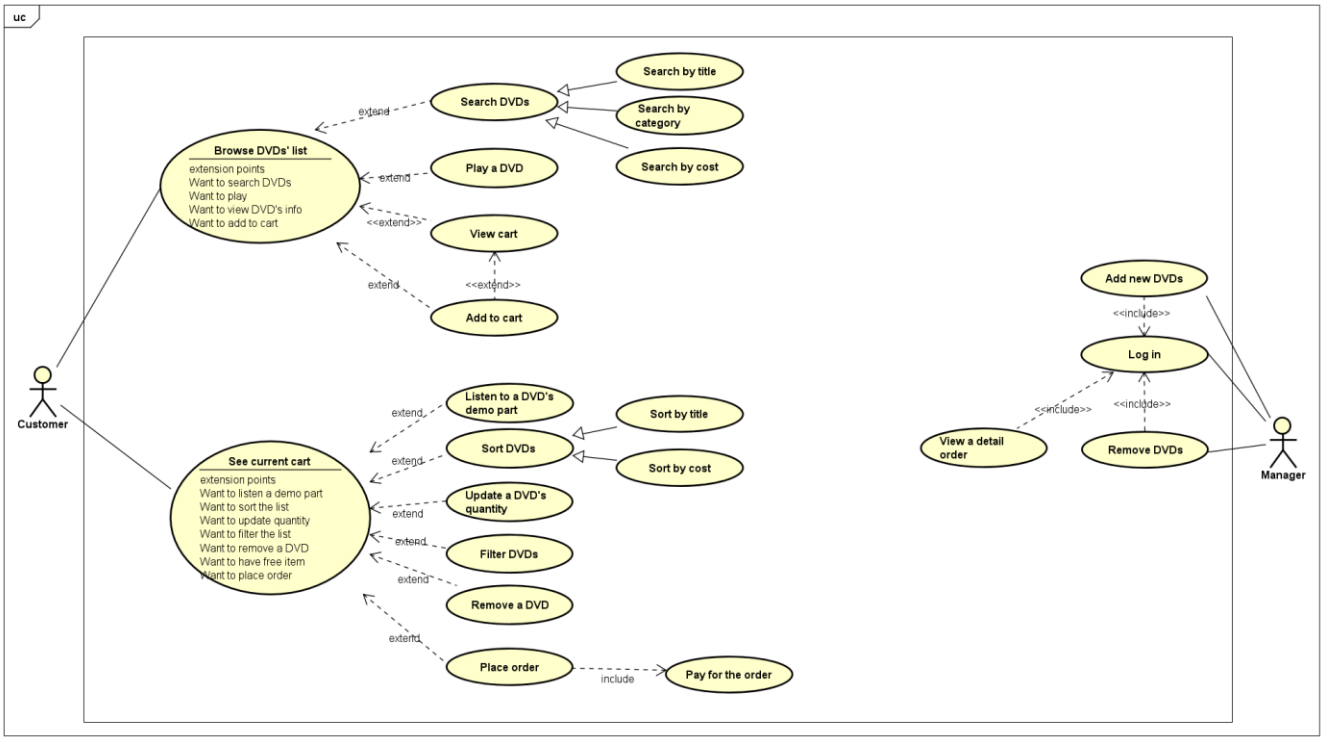


Change value



3. Images of the updated use-case diagram and class diagram





4. Answer the questions

Question 1: Is JAVA a Pass by Value or a Pass by Reference programming language?
Java is strictly **pass-by-value**

Question 2: After the call of `swap(jungleDVD, cinderellaDVD)` why does the title of these two objects still remain?
`jungleDVD --> Object A ("Jungle Book")`
`cinderellaDVD --> Object B ("Cinderella")`
Before the swap method:

- `o1` (local copy) points to `Object A`.
- `o2` (local copy) points to `Object B`.

Inside the swap method:

- temp points to `Object A`.
- `o1` is updated to point to `Object B`.
- `o2` is updated to point to `Object A`.

After the swap method ends:

- The original references `jungleDVD` and `cinderellaDVD` remain unchanged:
  - `jungleDVD` still points to `Object A`.
  - `cinderellaDVD` still points to `Object B`.

Question 3: After the call of `changeTitle(jungleDVD, cinderellaDVD.getTitle())` why is the title of the `JungleDVD` changed?
When `changeTitle` is called, the **value of the reference to `jungleDVD`** is passed into the method. This means the method's parameter `dvd` is a new variable that holds a copy of the reference pointing to the same object as `jungleDVD`.
Inside the method, the statement `String oldTitle = dvd.getTitle();` retrieves the current title of the `jungleDVD` object.

The next line, `dvd.setTitle(title);`, updates the title field of the object that dvd refers to. Since dvd and `jungleDVD` both point to the same object, the title change is reflected in the `jungleDVD` object.
The line `dvd = new DigitalVideoDisc(oldTitle);` creates a new `DigitalVideoDisc` object with the original title stored in `oldTitle`.
However, this reassignment only affects the **local variable dvd** inside the method. It no longer points to the same object as `jungleDVD`. The original `jungleDVD` reference in the calling method is not affected by this reassignment.