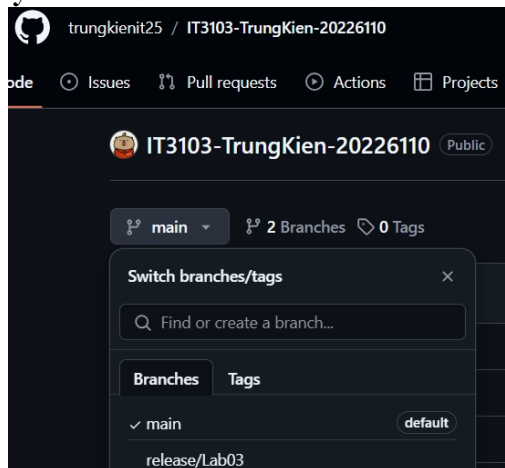


# Object-Oriented Programming

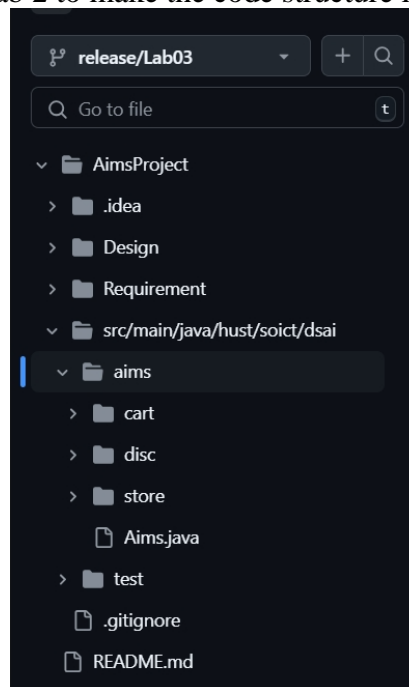
## Lab 03: Basic Object-Oriented Techniques

Họ và tên: Nguyễn Trung Kiên  
MSSV: 20226110

### 1. Branch your repository



- Restructure the folders from Lab 2 to make the code structure look like this:



## 2. Working with method overloading

### 2.1 Overloading by differing types of parameter

**Question:** 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?

#### Cách 1: Sử dụng varargs ():

- **Ưu điểm:**

- Linh hoạt, dễ sử dụng khi số lượng tham số truyền vào không cố định.
- Không cần tạo mảng trước khi gọi phương thức.
- Cú pháp gọn gàng, trực quan hơn.

#### Cách 2: Sử dụng mảng (DigitalVideoDisc[]):

- **Ưu điểm:**

- Thích hợp nếu dữ liệu đã có sẵn dưới dạng mảng.
- Rõ ràng hơn khi xử lý danh sách cố định các phần tử.

```
public void addDigitalVideoDisc(DigitalVideoDisc disc) {
    itemsOrdered[qtyOrdered] = disc;
    qtyOrdered++;
    System.out.println(x:"Awesome, the disc has been added!");
    if (MAX_NUMBERS_ORDERED == qtyOrdered) {
        System.out.println(x:"Warning! The cart is almost full!");
    }
}

// Add a list of DVDs to the current cart.
public void addDigitalVideoDisc(DigitalVideoDisc[] dvdList) {
    if (qtyOrdered + dvdList.length <= MAX_NUMBERS_ORDERED){
        for (DigitalVideoDisc dvd: dvdList) {
            itemsOrdered[qtyOrdered] = dvd;
            qtyOrdered++;
        }
        System.out.println(x:"The discs have been added.");
    } else {
        System.out.println(x:"Maximum number of orders exceeded.");
    }
}
```

Figure 1. Method addDigitalVideoDisc(DigitalVideoDisc [] dvdList)

```
//Nguyen Trung Kien 20226110
package hust.soict.dsai.aims;

import hust.soict.dsai.aims.cart.Cart;
import hust.soict.dsai.aims.disc.DigitalVideoDisc;

public class Aims {
    Run main | Debug main | Run | Debug
    public static void main(String[] args) {
        Cart cart = new Cart();
        DigitalVideoDisc d1 = new DigitalVideoDisc(title:"The Lion King",
            category:"Animation", director:"Roger Allers", length:87, cost:19.95f);
        DigitalVideoDisc d2 = new DigitalVideoDisc(title:"Star Wars", category:"Science fiction", director:"George Lucas", length:87, cost:24.95f);
        DigitalVideoDisc d3 = new DigitalVideoDisc(title:"Aladin",
            category:"Animation", cost:18.99f);
        DigitalVideoDisc[] dvdList = {d1, d2, d3};
        cart.addDigitalVideoDisc(dvdList);
        System.out.println("Total cost is: " + cart.totalCost());
        cart.removeDigitalVideoDisc(d3);
        System.out.println("Total cost after removal is: " + cart.totalCost());
    }
}
```

Figure 2. Aims Class

## 2.2. Overloading by differing the number of parameters

- Continuing focus on the **Cart** class

- Create new method named **addDigitalVideoDisc**

+ The signature of this method has two parameters as following:

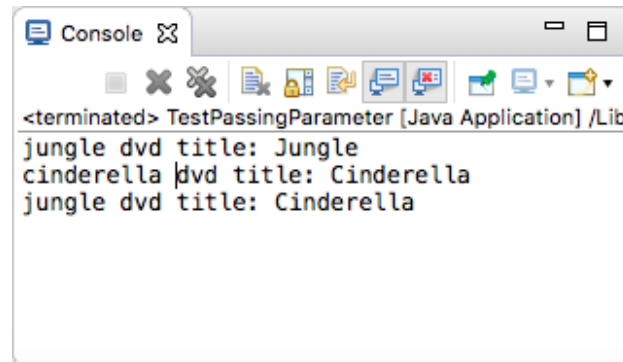
**addDigitalVideoDisc(DigitalVideoDisc dvd1, DigitalVideoDisc dvd2)**

```
public void addDigitalVideoDisc(DigitalVideoDisc d1, DigitalVideoDisc d2) {  
    if (qtyOrdered <= (MAX_NUMBERS_ORDERED - 2)){  
        addDigitalVideoDisc(d1);  
        addDigitalVideoDisc(d2);  
        System.out.println(x:"The discs have been added.");  
    } else {  
        System.out.println(x:"Maximum number of orders exceeded.");  
    }  
}
```

## 3. Passing parameter

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

The result in console is below:



```
<terminated> TestPassingParameter [Java Application] /Lib  
jungle dvd title: Jungle  
cinderella dvd title: Cinderella  
jungle dvd title: Cinderella
```

Figure 1. Results(1)

### Questions 1:

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

Trả lời: Đây là do Java **luôn luôn** truyền tham số **theo giá trị** (pass-by-value).

- Khi truyền o1 và o2 vào hàm swap, Java sẽ tạo ra **bản sao** của hai đối tượng này.
- Bên trong hàm swap, mọi thao tác thực hiện trên các bản sao này sẽ **không ảnh hưởng** đến các đối tượng o1 và o2 ban đầu.
- Mặc dù ta có đổi chỗ disc của hai bản sao, nhưng khi hàm swap kết thúc, các bản sao này bị hủy, và o1, o2 ban đầu vẫn giữ nguyên giá trị.

### Questions 2:

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

Trả lời:

- Trong trường hợp này, khi truyền `jungleDVD` vào hàm `changeTitle`, Java vẫn tạo ra một bản sao của đối tượng `DigitalVideoDisc`.
- Lớp `DigitalVideoDisc` có phương thức `setTitle()`. Phương thức này **thay đổi trực tiếp** thuộc tính `title` của đối tượng.
- Do đó, khi gọi `dvd.setTitle(title)` bên trong hàm `changeTitle`, bản sao của `jungleDVD` được thay đổi tiêu đề.
- Quan trọng là, bản sao này vẫn **trở đến cùng vùng nhớ** với `jungleDVD` ban đầu. Vì vậy, việc thay đổi tiêu đề của bản sao cũng đồng thời thay đổi tiêu đề của `jungleDVD` ban đầu.

Kiên will write a `swap()` method that can correctly swap the two objects.

```

1 public class TestPassingParameter {
2     public static void main(String[] args) {
3         DigitalVideoDisc jungleDVD = new DigitalVideoDisc("Jungle");
4         DigitalVideoDisc cinderellaDVD = new DigitalVideoDisc("Cinderella");
5         Wrap o1 = new Wrap(jungleDVD);
6         Wrap o2 = new Wrap(cinderellaDVD);
7         swap(o1, o2);
8         System.out.println("jungle dvd title: " + o1.disc.getTitle());
9         System.out.println("cinderella dvd title: " + o2.disc.getTitle());
10
11         changeTitle(jungleDVD, cinderellaDVD.getTitle());
12         System.out.println("jungle dvd title: " + jungleDVD.getTitle());
13     }
14
15     public static void swap(Wrap o1, Wrap o2) {
16         DigitalVideoDisc tmp = o1.disc;
17         o1.disc = o2.disc;
18         o2.disc = tmp;
19     }
20
21     public static void changeTitle(DigitalVideoDisc dvd, String title){
22         String oldTitle = dvd.getTitle();
23         dvd.setTitle(title);
24         dvd = new DigitalVideoDisc(oldTitle);
25     }
26 }
27 class Wrap {
28     DigitalVideoDisc disc;
29     Wrap(DigitalVideoDisc d) {
30         this.disc = d;}
31 }

```

< Problems Javadoc Declaration Console ×

<terminated> TestPassingParameter [Java Application] C:\Program Files\Java\jdk-22\bin\javaw.exe (Nov 23, 2023)

```

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

```

## 4. Use debug run:

### 4.1. Debugging Java in Eclipse

Video: <https://www.youtube.com/watch?v=9gAjlQc4bPU&t=8s>

### 4.2. Example of debug run for the *swap* method of *TestPassingParameter*

#### 4.2.1. Setting, deleting & deactivate breakpoints:

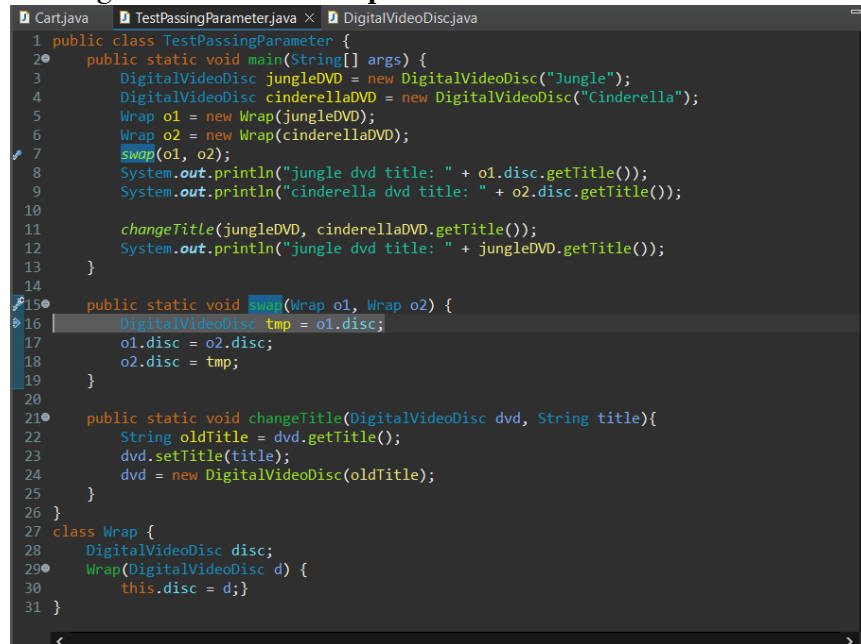


Figure 2. A breakpoint is set

To deactivate the breakpoint, navigate to the Breakpoints View and uncheck the tick mark next to the breakpoint you want to deactivate (Figure 3). **The program will only stop at activated breakpoints.**

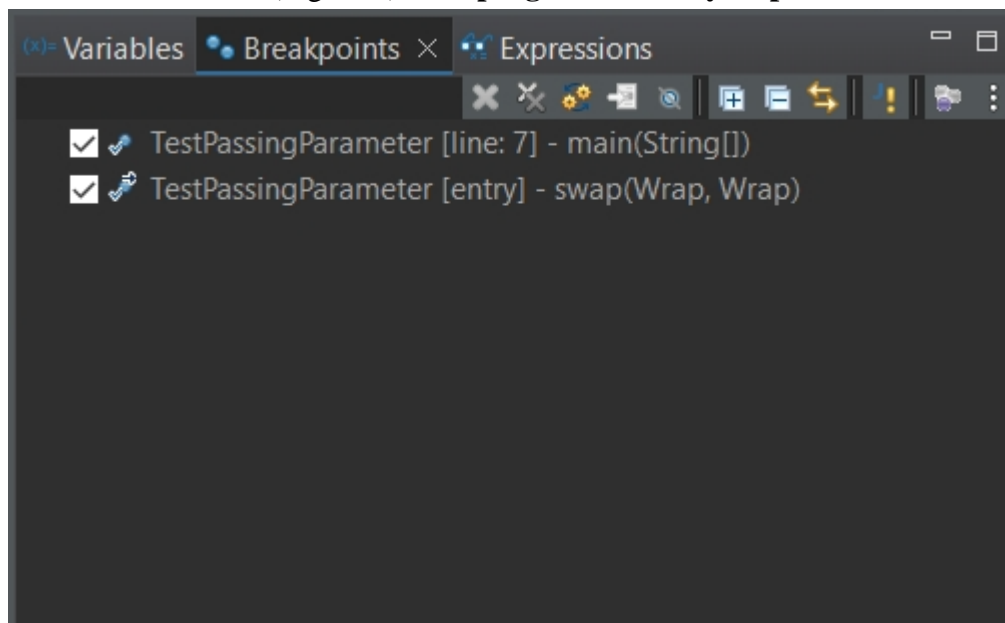


Figure 3. Deactivated breakpoint in Breakpoints View



#### 4.2.2. Run in Debug mode:

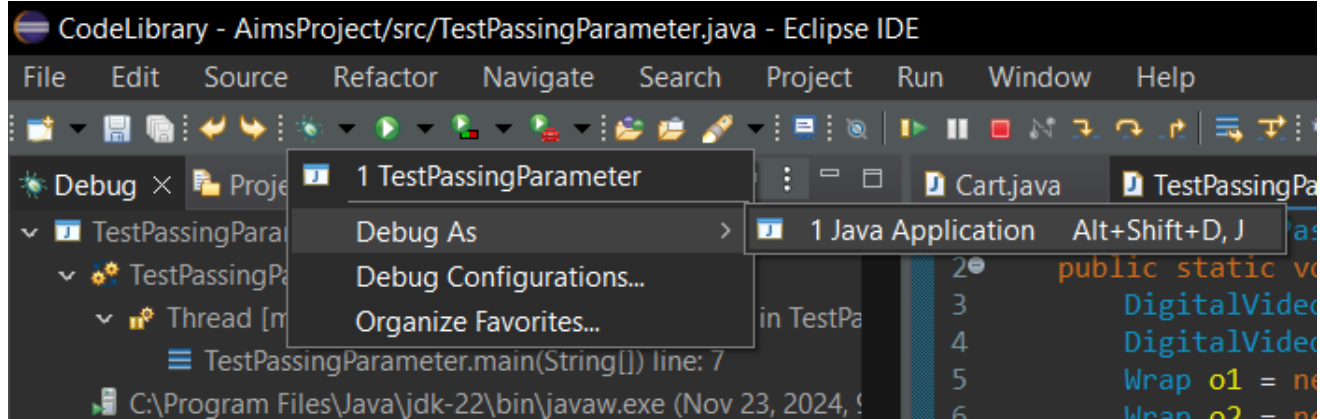


Figure 4. Run Debug from a class

#### 4.2.3. Step Into, Step Over, Step Return, Resume:

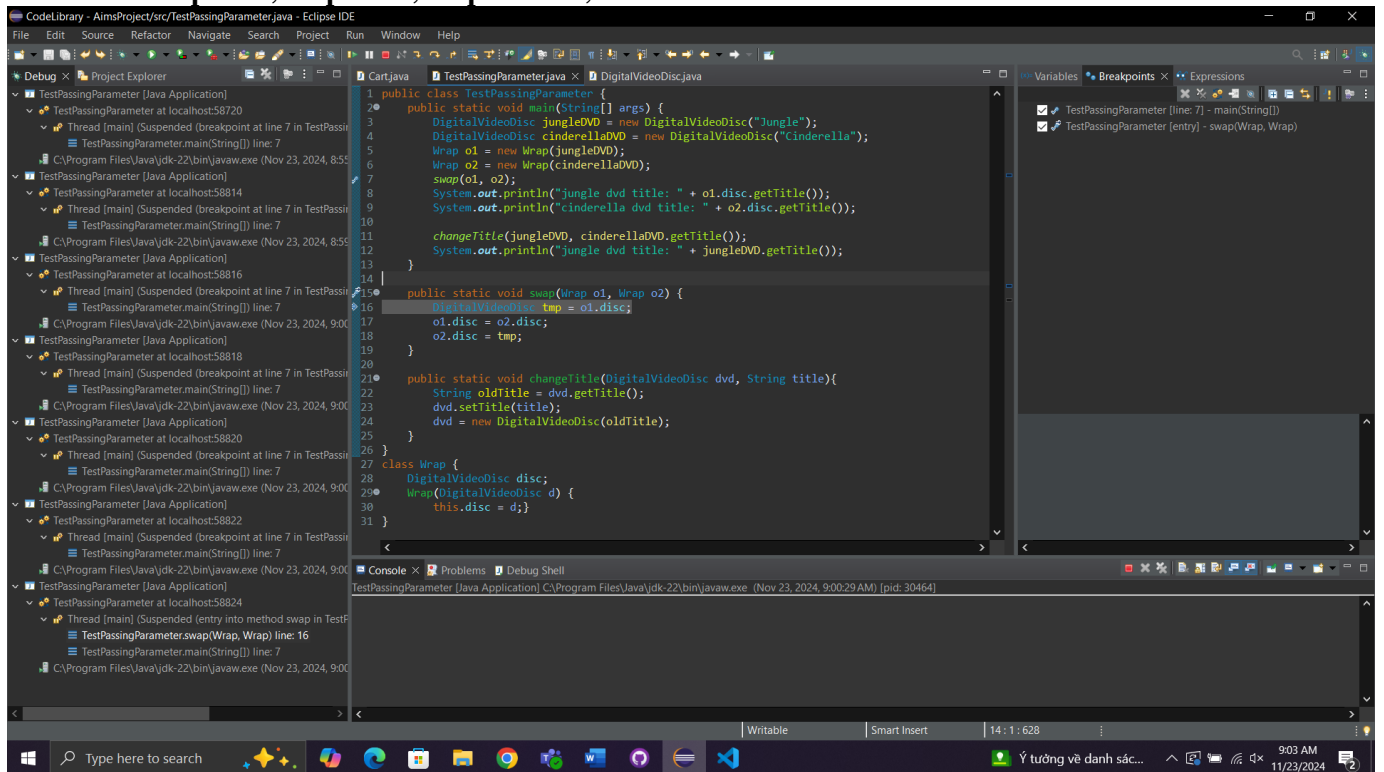


Figure 5. Step into swap function

#### 4.2.4. Investigate value of variables:

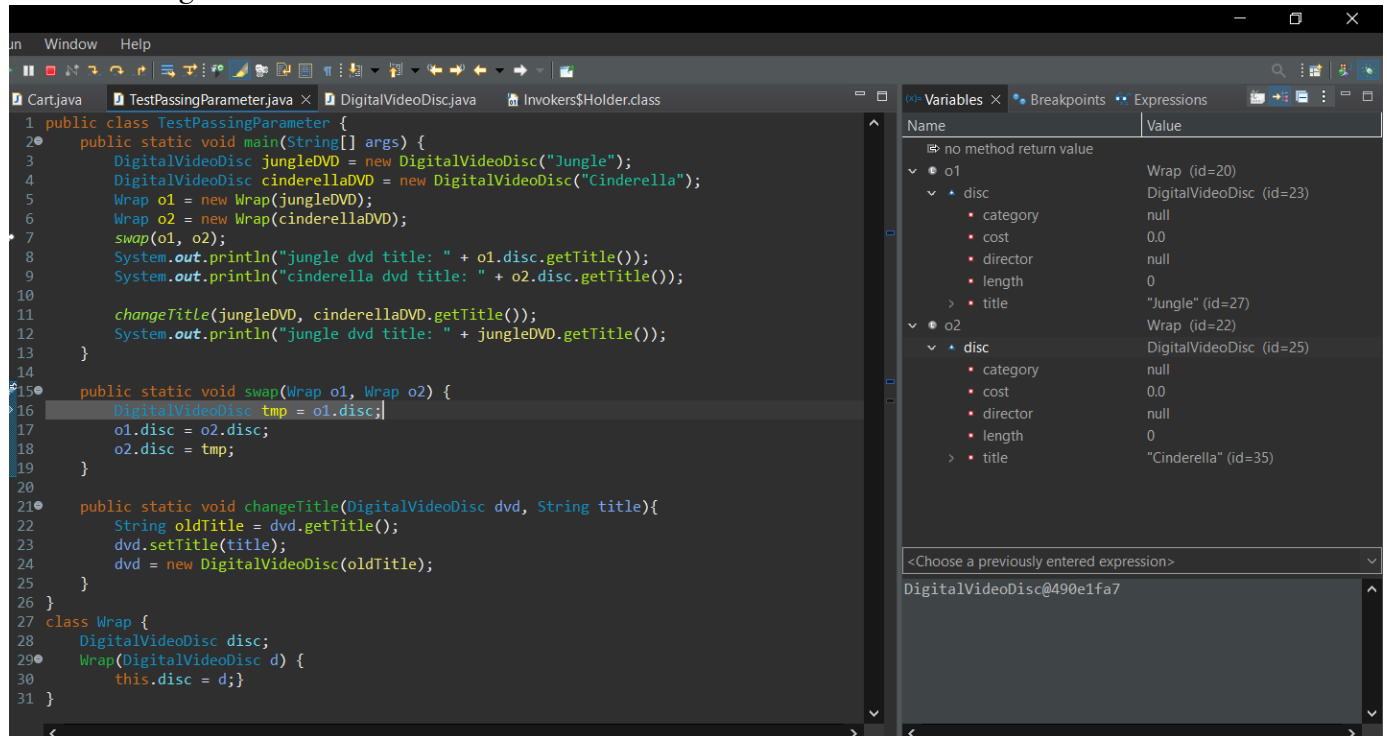


Figure 6. Variables shown in Variable View

Click Step Over and watch the change in the value of variables `o1`, `o2` & `tmp`. Repeat this until the end of the `swap` function (Figure 7, Figure 8, Figure 9).

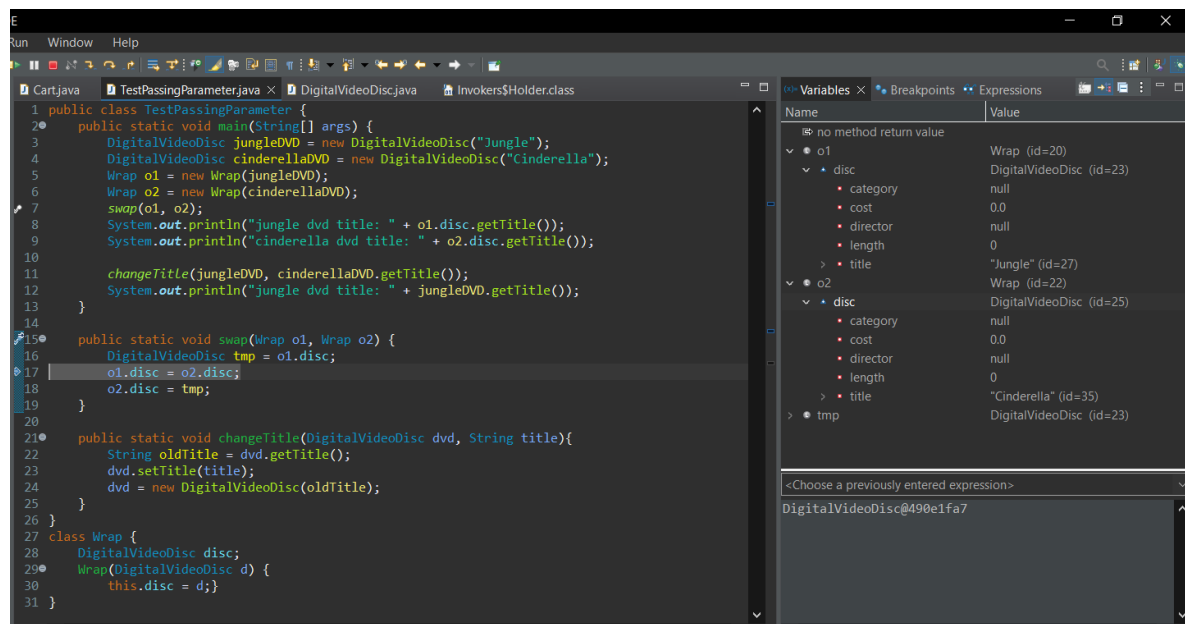


Figure 7. Step over line 18 of swap function

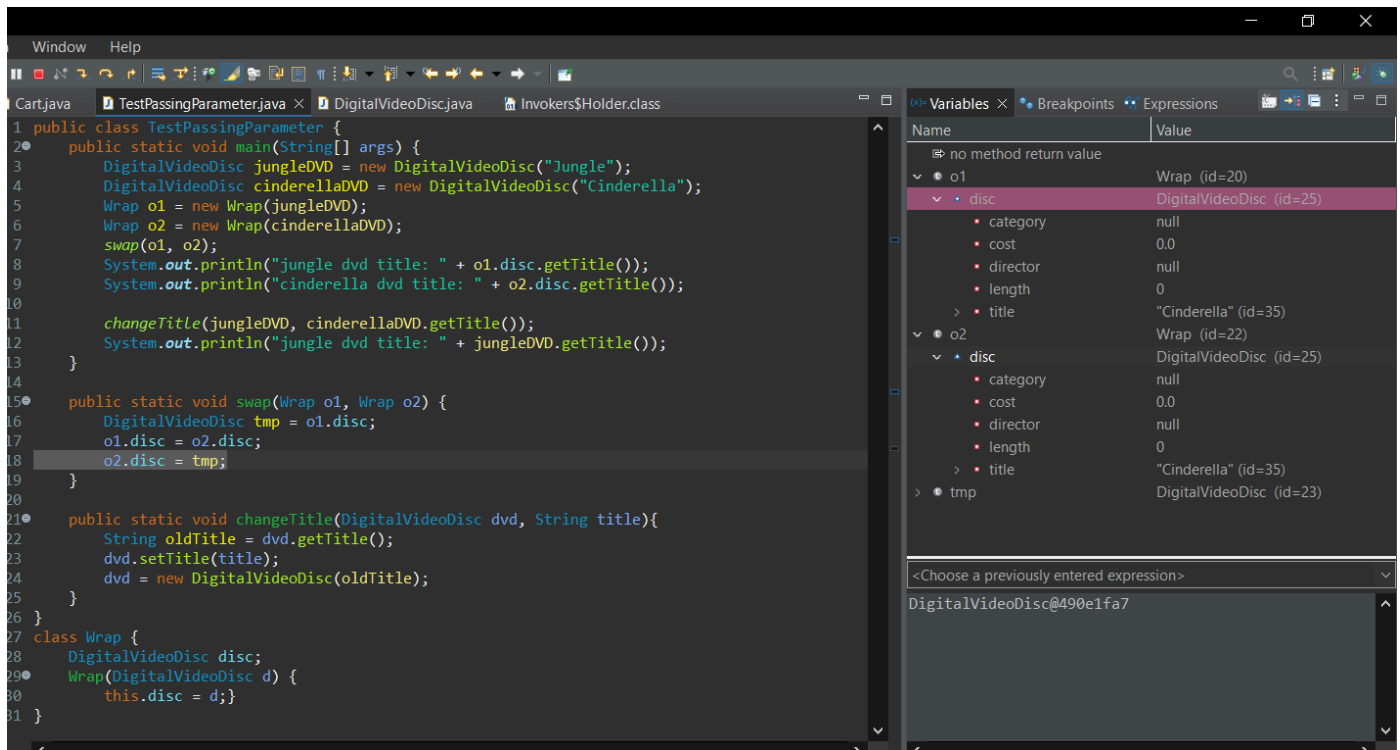


Figure 8. Step over line 19 of swap function

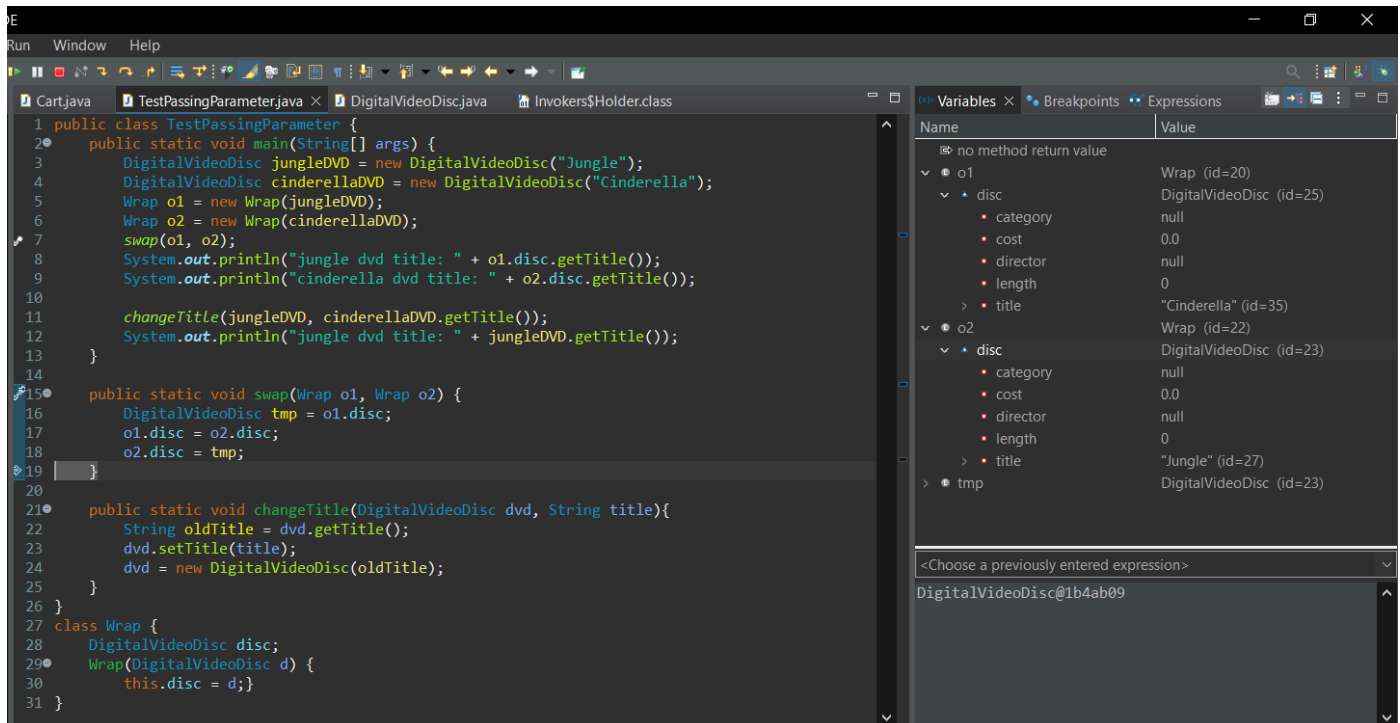


Figure 9. Step over line 20 of swap function



#### 4.2.4. Change value of variables:

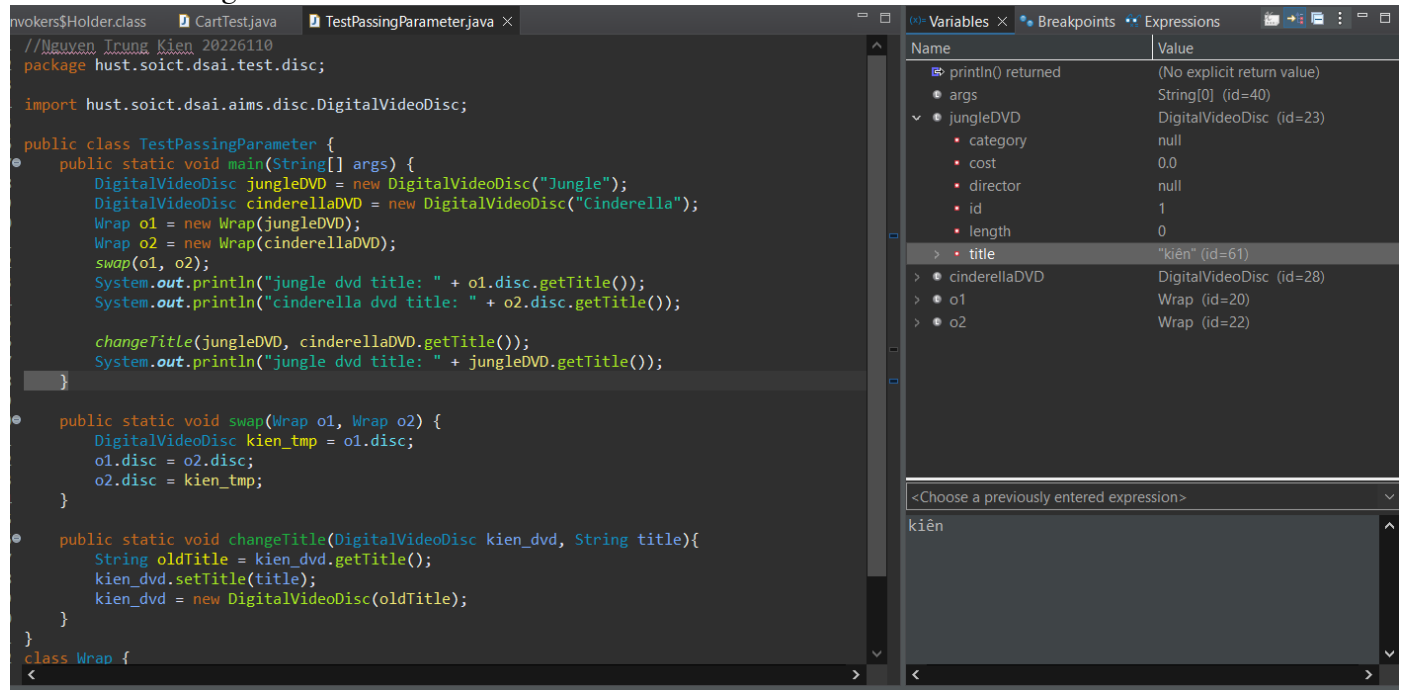


Figure 10. Step return to main function

The variable `jungleDVD` still has a title attribute with value “Jungle”. Change this value by clicking on it and change it to “kiên”.

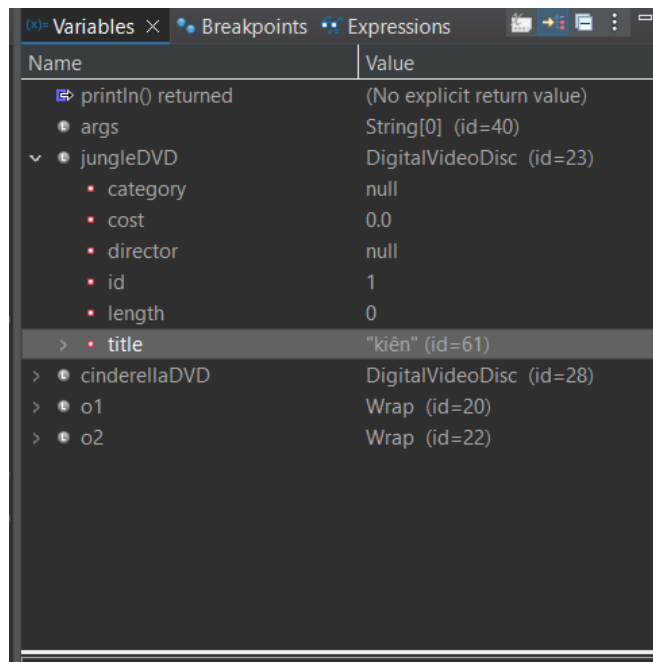


Figure 11. Change title of jungleDVD

Click Step Over and see the result in the output in the Console (Figure 12)

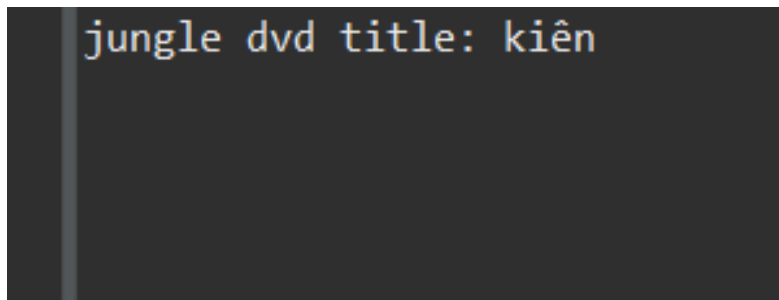


Figure 12. Results(2)

## 5. Classifier Member and Instance Member

```
//Nguyen Trung Kien 20226110
package hust.soict.dsai.aims.disc;

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;

    public DigitalVideoDisc(String title) {

        this.title = title;
        nbDigitalVideoDiscs ++;
        this.id = nbDigitalVideoDiscs;
    }

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

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

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

```

        this.id = nbDigitalVideoDiscs;
    }

    public String getTitle() {
        return title;
    }

    public String getCategory() {
        return category;
    }

    public String getDirector() {
        return director;
    }

    public int getLength() {
        return length;
    }

    public float getCost() {
        return cost;
    }

    public int getId() {
        return id;
    }

    public void setTitle(String title) {
        this.title = title;
    }

    @Override
    public String toString() {
        return "DVD - " + title + " - " + category + " - " +
            director + " - " + length + ": " + cost + " $";
    }

    public boolean isMatch(String searchStr) {
        boolean matched = false;
        String[] searchArr = searchStr.split(" ", 0);
        for (String word: searchArr) {
            String lowerCaseTitle = title.toLowerCase();
            int index = lowerCaseTitle.indexOf(word.toLowerCase());
            if (index != -1) {
                matched = true;
                break;
            }
        }
        return matched;
    }
}

```

## 6. Open the **Cart** class

```
public void print() {
    System.out.println(x:"*****OUR*CART*****");
    System.out.println(x:"Ordered Items:");
    for (int i = 1; i <= qtyOrdered; i++ ) {
        System.out.println(i + ". " + itemsOrdered[i-1].toString());
    }
    System.out.println("Total cost is: " + totalCost());
    System.out.println(x:"*****");
}
```

Figure 13. Content in Cart

```
public void searchById(int id) {
    boolean found = false;
    System.out.println("Search: " + id);
    System.out.println(x:"Search result:");
    for (int i = 0; i < qtyOrdered; i++ ) {
        DigitalVideoDisc dvd = itemsOrdered[i];
        if (dvd.getId() == id) {
            System.out.println(dvd.toString());
            found = true;
            break;
        }
    }
    if (!found) {
        System.out.println(x:"No match is found.");
    }
}
```

Figure 20. Search by id

```
public void searchByTitle(String title) {
    System.out.println("Search: " + title);
    System.out.println(x:"Search result:");
    boolean found = false;
    for (int i = 0; i < qtyOrdered; i++) {
        DigitalVideoDisc dvd = itemsOrdered[i];
        if (dvd.isMatch(title)) {
            System.out.println(dvd.toString());
            found = true;
        }
    }
    if (!found) {
        System.out.println(x:"No match is found.");
    }
}
```

Figure 21. Search by title

```

@Override
public String toString() {
    return "DVD - " + title + " - " + category + " - " +
        director + " - " + length + ": " + cost + " $";
}

public boolean isMatch(String searchStr) {
    boolean matched = false;
    String[] searchArr = searchStr.split(regex:" ", limit:0);
    for (String word: searchArr) {
        String lowerCaseTitle = title.toLowerCase();
        int index = lowerCaseTitle.indexOf(word.toLowerCase());
        if (index != -1) {
            matched = true;
            break;
        }
    }
    return matched;
}

```

Figure 21. isMatch()

```

//Nguyen Trung Kien 20226110
package hust.soict.dsai.test.cart;

import hust.soict.dsai.aims.cart.Cart;
import hust.soict.dsai.aims.disc.DigitalVideoDisc;

public class CartTest {
    public static void main(String[] args) {
        Cart cart = new Cart();
        DigitalVideoDisc d1 = new DigitalVideoDisc(title:"The Lion King",
            category:"Animation", director:"Roger Allers", length:87, cost:19.95f);
        DigitalVideoDisc d2 = new DigitalVideoDisc(title:"Star Wars", category:"Science Fiction", director:"George Lucas", length:87,
            cost:18.99f);
        DigitalVideoDisc d3 = new DigitalVideoDisc(title:"Atadin",
            category:"Animation", cost:18.99f);
        DigitalVideoDisc[] dvdList = {d1, d2, d3};
        cart.addDigitalVideoDisc(dvdList);
        cart.print();
        cart.searchById(id:1);
        cart.searchByTitle(title:"The scorpio has no heart");
    }
}

```

Figure 21. CartTest Class

## 7. Implement the **Store** class

```

package hust.soict.dsai.aims.store;

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

public class Store {
    public static int MAX_QUANTITY = 1000000;
    private DigitalVideoDisc[] itemsInStore = new DigitalVideoDisc[MAX_QUANTITY];
    private int quantity = 0;
    public void addDVD(DigitalVideoDisc disc){
        if (quantity < MAX_QUANTITY) {

```

```

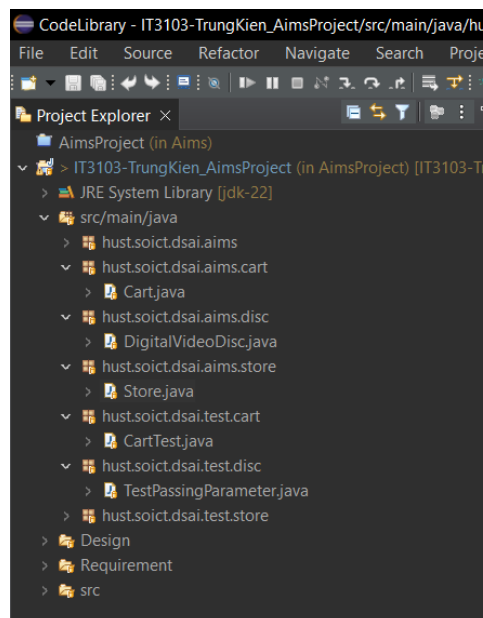
        itemsInStore[quantity] = disc;
        quantity++;
        System.out.println("The disc has been added.");
    } else {
        System.out.println("Maximum number of DVDs exceeded.");
    }
}

public void removeDVD(DigitalVideoDisc disc){
    boolean removed = false;
    for (int i = 0; i < quantity; i++) {
        if (itemsInStore[i] == disc) {
            for (int j = i; j < quantity - 1; j++) {
                itemsInStore[j] = itemsInStore[j + 1];
            }
            quantity--;
            System.out.println("The disc has been removed.");
            removed = true;
            break;
        }
    }
    if (!removed) {
        System.out.println("The disc is not found.");
    }
}

public void print(){
    System.out.println("*****STORE*****");
    System.out.println("In-store items:");
    for (int i = 0; i < quantity; i++){
        System.out.println(itemsInStore[i].toString());
    }
    System.out.println("*****");
}
}

```

## 8. Re-organize your projects





## 9. String, StringBuilder and StringBuffer

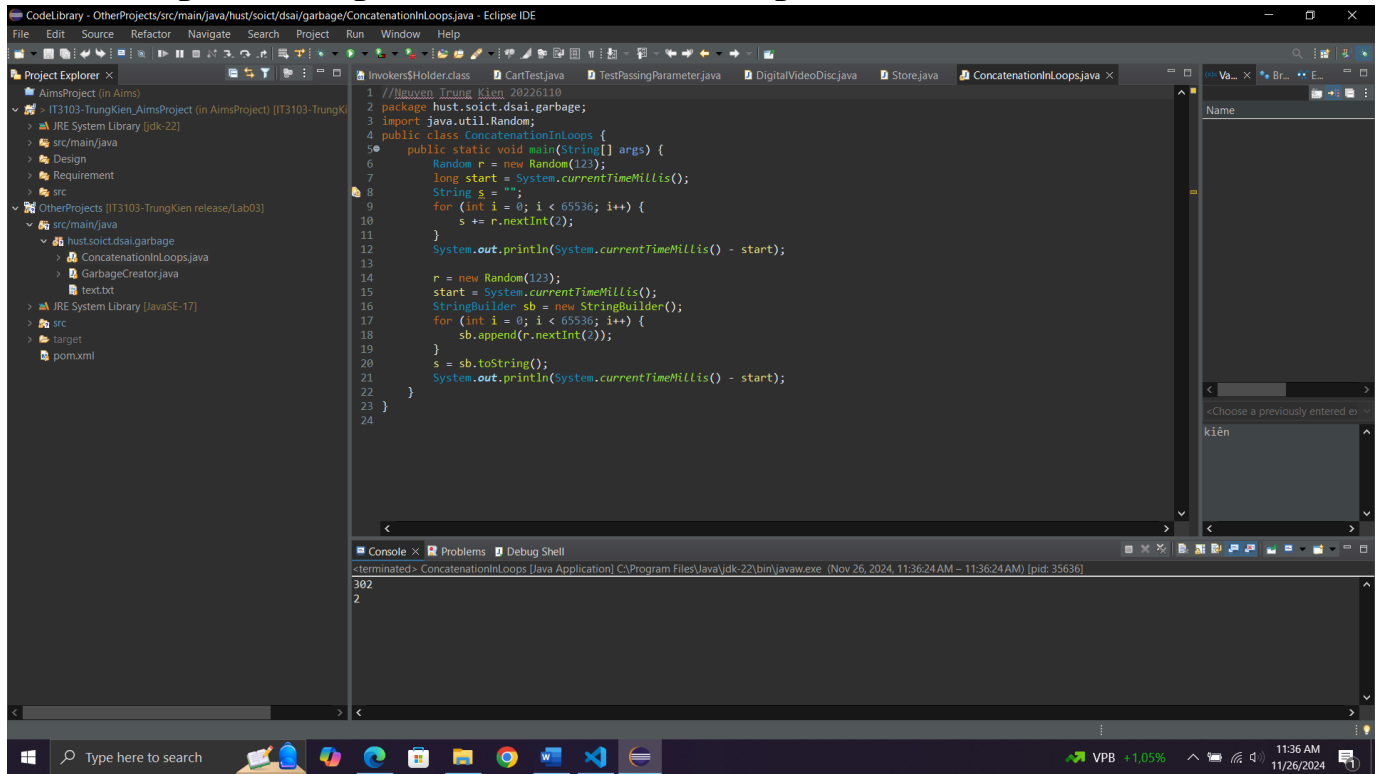


Figure 14. So sánh String và StringBuilder

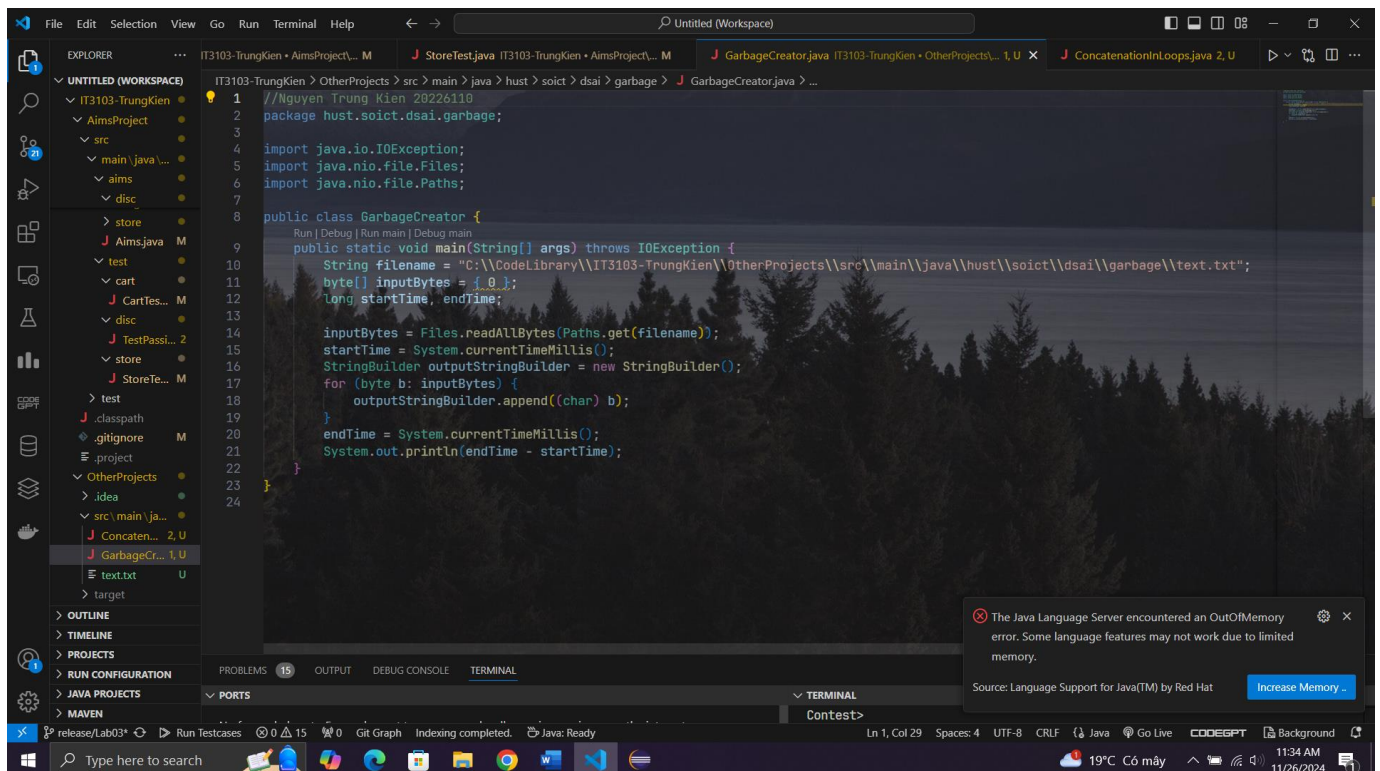


Figure 15. Sample code for GarbageCreator

## Giải thích hiện tượng:

Ban đầu, với file nhỏ, chương trình xử lý nhanh vì số lượng dữ liệu ít, việc nối chuỗi bằng + chưa tạo áp lực lớn lên bộ nhớ. Khi file lớn dần, thời gian thực thi tăng mạnh, và với file rất lớn, chương trình có thể treo hoặc dừng vì tốn quá nhiều bộ nhớ và CPU.

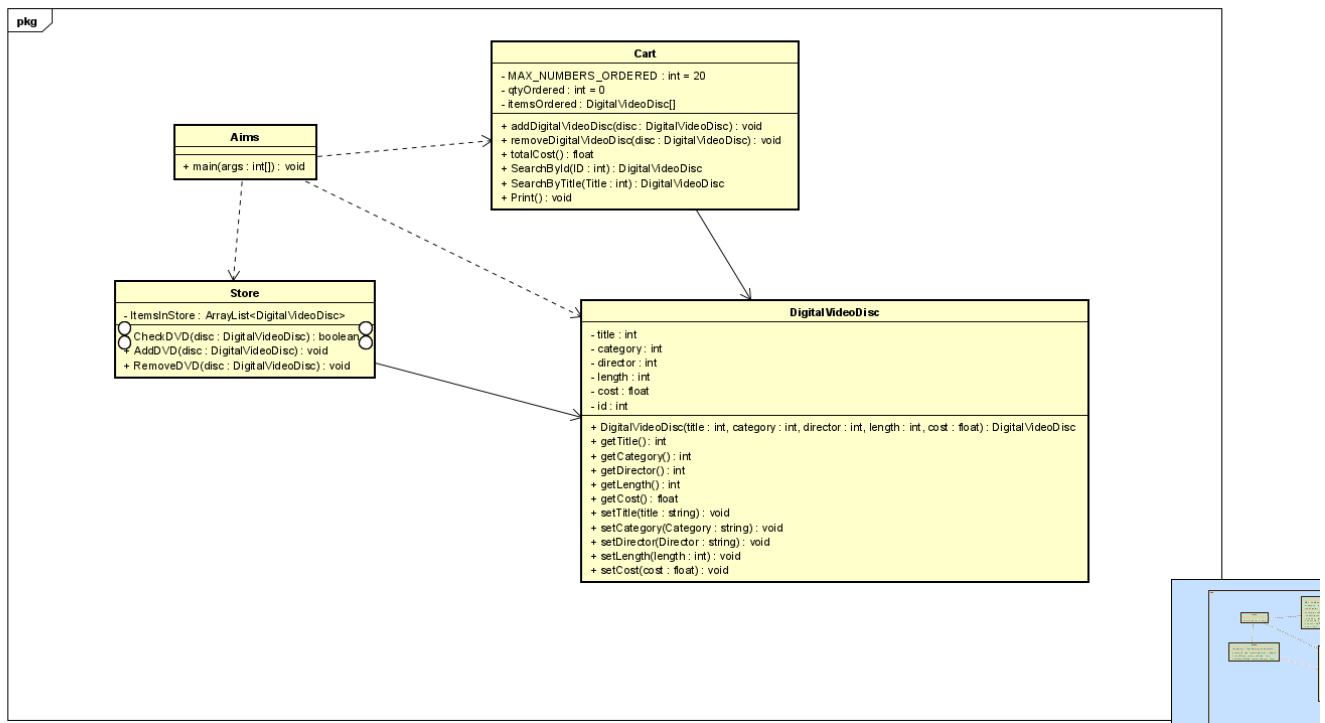
## Nguyên nhân:

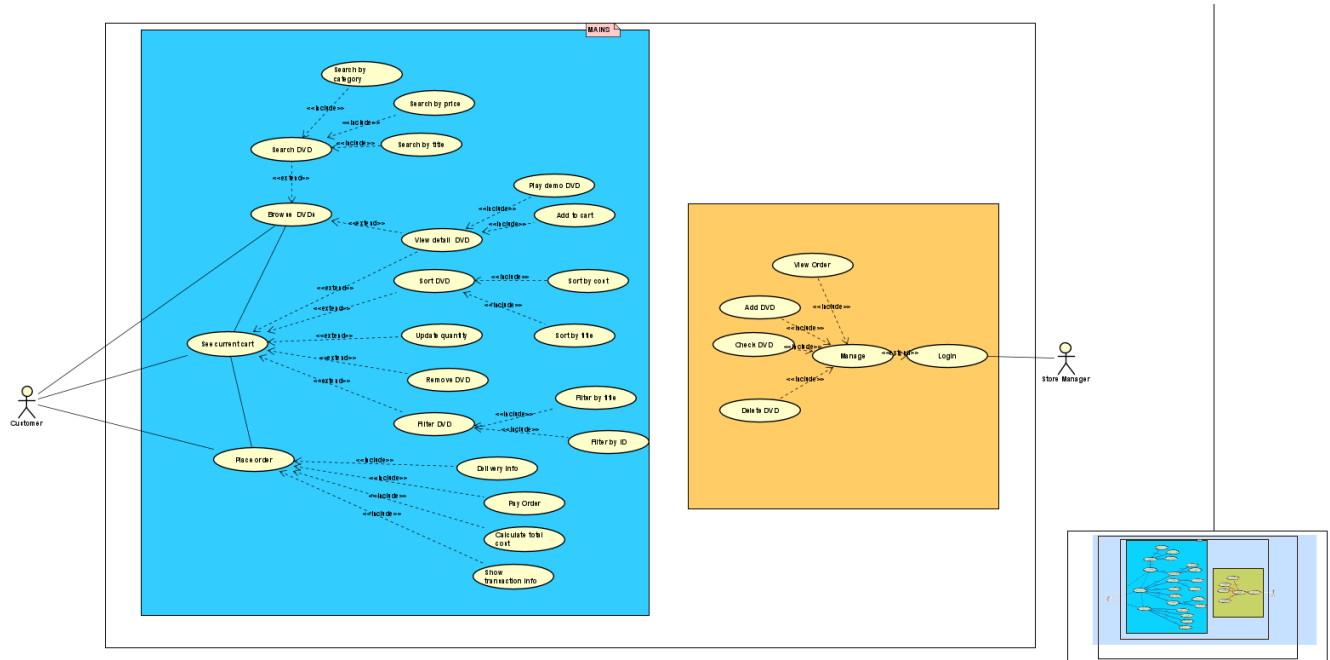
1. **String là immutable:** Mỗi lần nối chuỗi bằng +, Java tạo một đối tượng String mới, sao chép toàn bộ dữ liệu cũ, dẫn đến tăng chi phí bộ nhớ và CPU.
2. **Quản lý bộ nhớ:** Tạo nhiều đối tượng làm Garbage Collector hoạt động quá tải, gây chậm và lỗi `OutOfMemoryError`.
3. **Độ phức tạp cao:** Nối chuỗi bằng + có độ phức tạp  $O(n^2)$  khi file lớn.

## Giải pháp:

1. **Dùng StringBuffer/StringBuilder:** Nối chuỗi hiệu quả hơn với độ phức tạp  $O(n)$ , giảm rác bộ nhớ.
2. **Chia nhỏ file:** Xử lý từng phần nhỏ để giảm tải bộ nhớ.
3. **Tối ưu hóa đọc file:** Hạn chế đọc toàn bộ file vào chuỗi một lần.

## 10. Diagram





## 11. Release flow demonstration

(In Github)