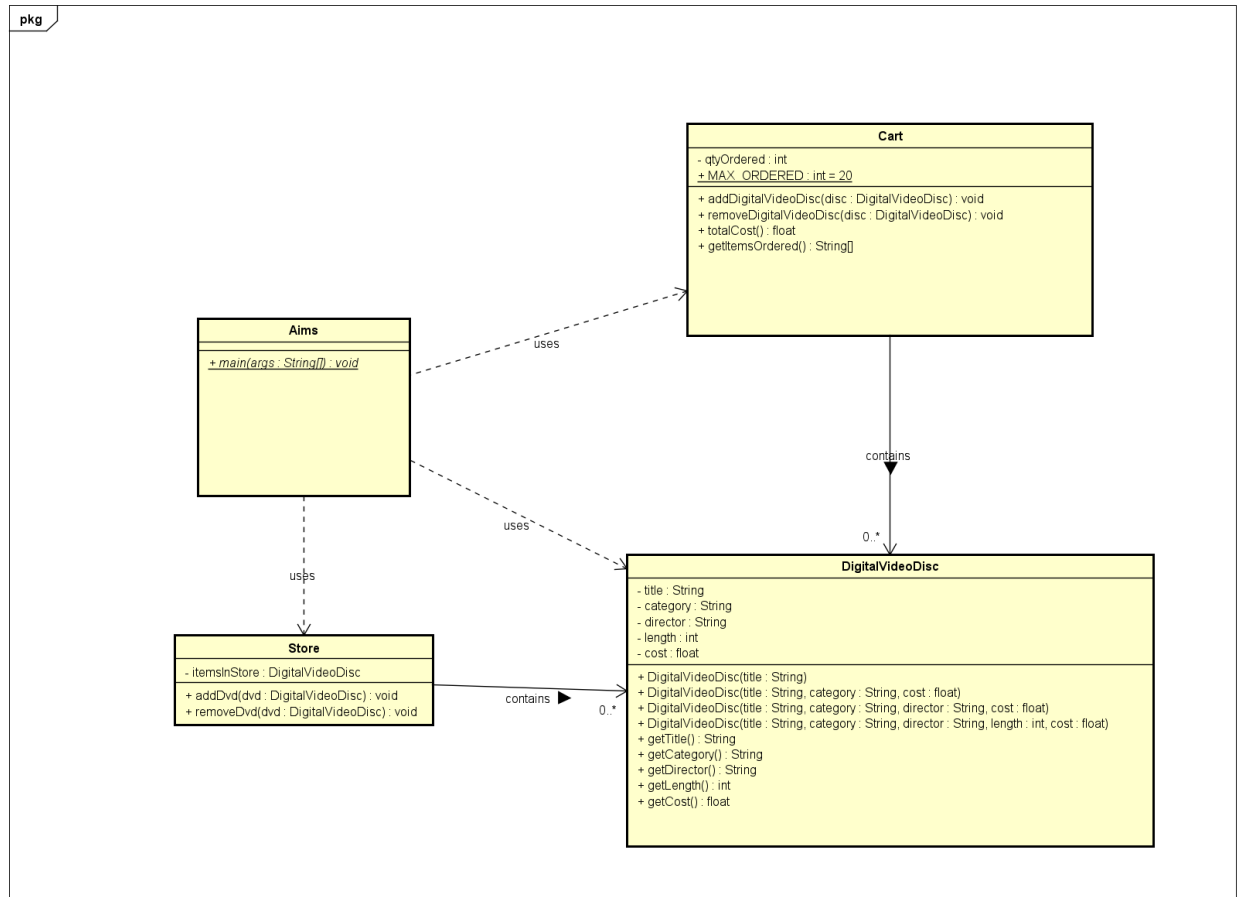


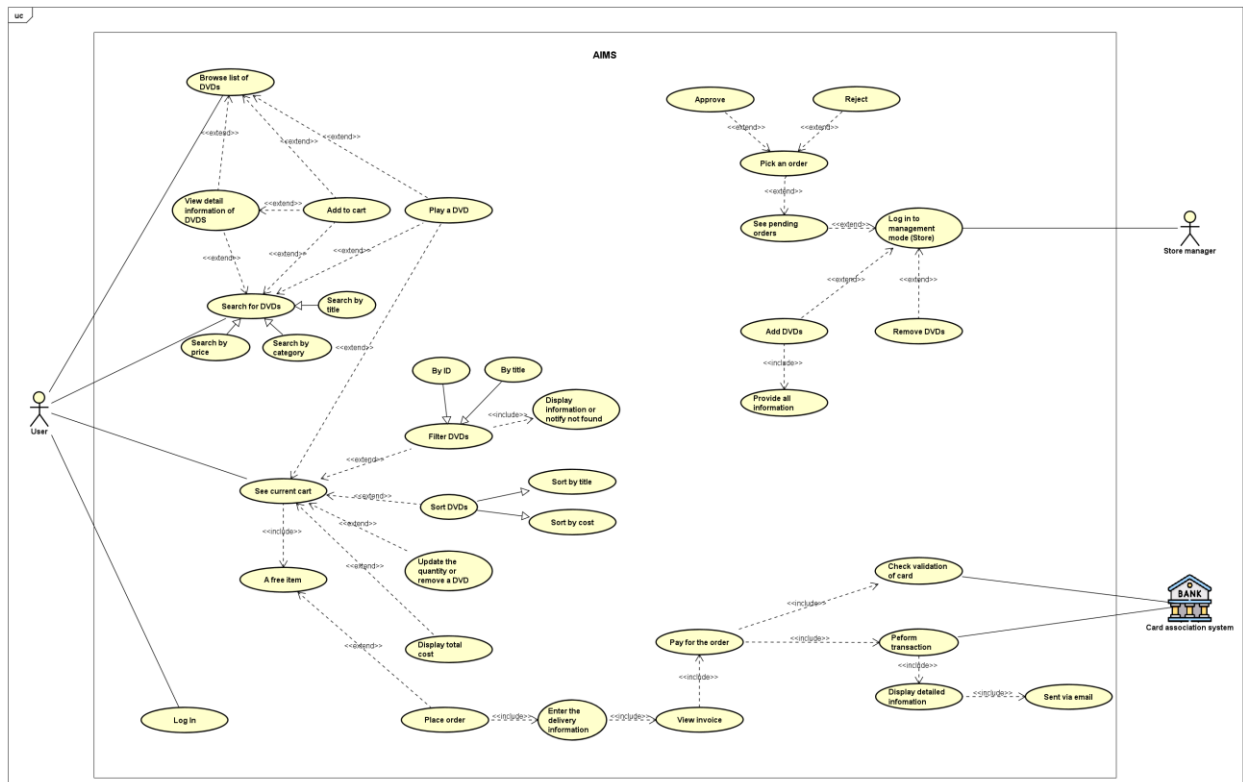
REPORT LAB 03

Name: Nguyễn Vũ Thủy

Student ID: 20235562

1. Update use-case diagram and class diagram





2. Working with method overloading

```

18
19
20
21 // Overload addDigitalVideoDisc allow list DVD as params
22 public void addDigitalVideoDisc( DigitalVideoDisc [] dvdList ) {
23     if ( qtyOrdered >= MAX_ORDERED ) {
24         System.out.println("The cart is not capable of add these ammount of dvds");
25     } else {
26         for (int i = 0; i < dvdList.length; i++) {
27             itemsOrdered[qtyOrdered] = dvdList[i];
28             qtyOrdered += 1;
29         }
30         System.out.println("Added");
31     }
32 }
33
34
35 // Overload addDigitalVideoDisc allow 2 DVD as params -> Ktra them 2 cai co vuot qua so luong cho phep khong. Neu khong thi add vao neu co thi bao
36 public void addDigitalVideoDisc( DigitalVideoDisc dvd1, DigitalVideoDisc dvd2 ) {
37     if ( qtyOrdered + 2 >= MAX_ORDERED ) {
38         System.out.println("The cart is not capable of adding two more dvds");
39     } else {
40         itemsOrdered[qtyOrdered] = dvd1;
41         qtyOrdered += 1;
42         itemsOrdered[qtyOrdered] = dvd2;
43         qtyOrdered += 1;
44         System.out.println("Added");
45     }
46 }
47

```

- 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 think I prefer using the array parameter because it allows me to easily determine how many DVDs I need to add to the cart for validation, whereas handling an arbitrary number of DVDs would be more difficult.

3. Passing parameter

```
1 package hust.soict.dsai.test.disc;
2 import hust.soict.dsai.aims.disc.DigitalVideoDisc;
3
4 public class TestPassingParameter {
5     public static void main(String[] args) {
6         DigitalVideoDisc jungleDVD = new DigitalVideoDisc("Jungle");
7         DigitalVideoDisc cinderllaDVD = new DigitalVideoDisc("Cinderella");
8
9         swap(jungleDVD, cinderllaDVD);
10        System.out.println("jungle dvd title: " + jungleDVD.getTitle());
11
12        trueSwap(jungleDVD, cinderllaDVD);
13        System.out.println("jungle dvd title: " + jungleDVD.getTitle());
14
15        System.out.println("cinerella dvd title: " + cinderllaDVD.getTitle());
16
17        changeTitle(jungleDVD, cinderllaDVD.getTitle());
18        System.out.println("jungle dvd title: " + jungleDVD.getTitle());
19    }
20
21    public static void swap(Object o1, Object o2) {
22        Object tmp = o1;
23        o1 = o2;
24        o2 = tmp;
25    }
26
27    // Using a wrapper class to hold the references
28    public static void trueSwap(DigitalVideoDisc dvd1, DigitalVideoDisc dvd2) {
29        String tempTitle = dvd1.getTitle();
30        dvd1.setTitle(dvd2.getTitle());
31        dvd2.setTitle(tempTitle);
32    }
33
34    public static void changeTitle(DigitalVideoDisc dvd, String title) {
35        String oldTitle = dvd.getTitle();
36        dvd.setTitle(title);
37        dvd = new DigitalVideoDisc(oldTitle);
38    }
39 }
40
```

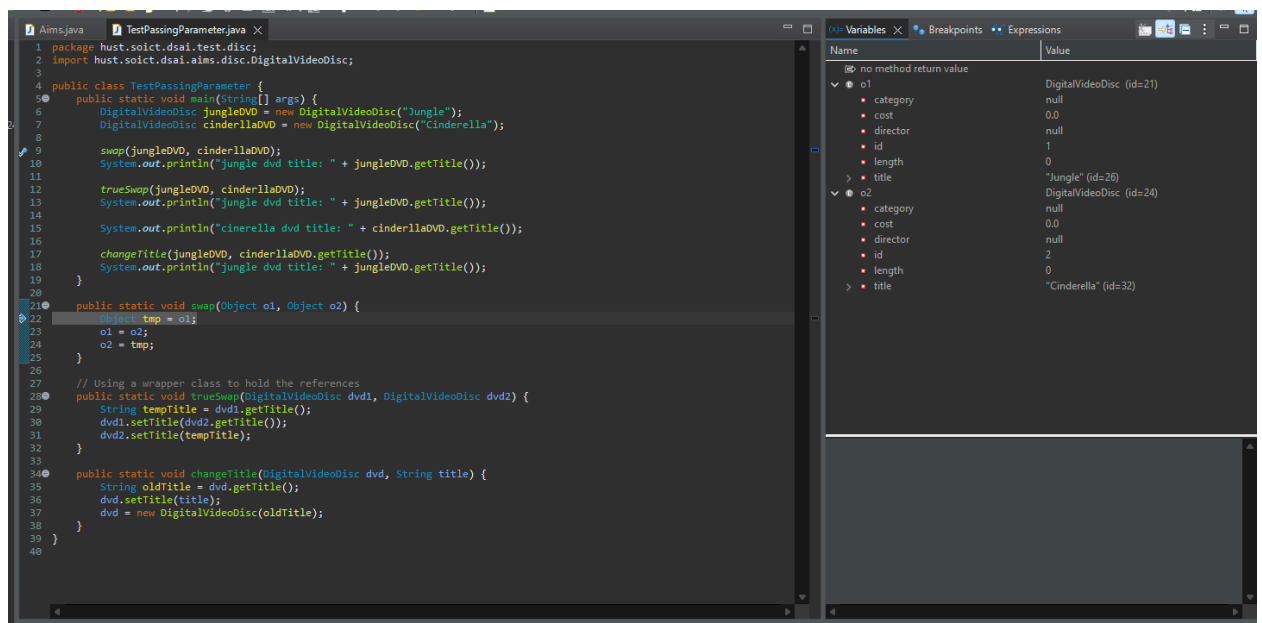
```
<terminated> TestPassingParameter [Java Application] C:\Program Files\Eclipse Adoptium\jdk-17.0.12.7-hotspot
jungle dvd title: Jungle
jungle dvd title: Cinderella
cinerella dvd title: Jungle
jungle dvd title: Jungle
```

- ***Is JAVA a Pass by Value or a Pass by Reference programming language?***
 - Java Pass by Value. For example, if we pass an object into a method in Java (swap(DVD dvd1, DVD dvd2)) then in the method swap only recieve the address value point to the dvd1 Object and dvd2 object in the memory so if we try to swap

by Obj tmp = dvd1; dvd1 = dvd2; dvd2 = tmp then it won't work. Because it is only dvd1 and dvd2 in the method change value for each other which does not affect original 2 objects.

- After the call of **swap(jungleDVD, cinderellaDVD)** why does the title of these two objects still remain?
 - As i said earlier, o1 and o2 are just local variables of method swap so swap value of o1 and o2 does not affect the value of original objects which are jungleDVD and cinderellaDVD. So the value of jungleDVD and cinderellaDVD still remain
- After the call of **changeTitle(jungleDVD, cinderellaDVD.getTitle())** why is the title of the JungleDVD changed?
 - In changeTitle, we have passed down the address of jungleDVD object so when we modify the title of dvd (which is the jungleDVD object) it also changes the title of jungleDVD because both point to the same Object.

4. Debugging Java in Eclipse



IDE screenshot showing the initial state of the program. The code in `TestPassingParameter.java` defines a `main` method and several helper methods: `swap`, `trueSwap`, `changeTitle`, and `swap`. The `main` method creates two `DigitalVideoDisc` objects, `jungleDVD` and `cinderllaDVD`, and prints their titles. It then calls `swap(jungleDVD, cinderllaDVD)`, `trueSwap(jungleDVD, cinderllaDVD)`, `changeTitle(jungleDVD, cinderllaDVD.getTitle())`, and `swap(jungleDVD, cinderllaDVD)` again. The `swap` method uses a temporary variable `tmp` to swap the references. The `trueSwap` method uses a wrapper class `DvdWrapper` to hold the references. The `changeTitle` method creates a new `DigitalVideoDisc` object with the same title as the original object.

The Variables window shows the state of the program. The `no method return value` section shows the state of the `swap` method. The `args` array contains the arguments passed to the `main` method. The `jungleDVD` and `cinderllaDVD` variables are shown with their respective attributes: `category`, `cost`, `director`, `id`, and `length`. The `tmp` variable is shown with its value, which is a `DigitalVideoDisc` object with `id=21`.

IDE screenshot showing the state of the program after the first `swap` call. The `main` method has executed the first `swap(jungleDVD, cinderllaDVD)` call. The `swap` method has swapped the references of `jungleDVD` and `cinderllaDVD`. The `tmp` variable now holds the reference to the `DigitalVideoDisc` object with `id=24`. The `no method return value` section shows the state of the `swap` method. The `args` array contains the arguments passed to the `main` method. The `jungleDVD` and `cinderllaDVD` variables are shown with their respective attributes. The `tmp` variable is shown with its value, which is a `DigitalVideoDisc` object with `id=24`.

IDE screenshot showing the state of the program after the second `swap` call. The `main` method has executed the second `swap(jungleDVD, cinderllaDVD)` call. The `swap` method has swapped the references of `jungleDVD` and `cinderllaDVD`. The `tmp` variable now holds the reference to the `DigitalVideoDisc` object with `id=21`. The `no method return value` section shows the state of the `swap` method. The `args` array contains the arguments passed to the `main` method. The `jungleDVD` and `cinderllaDVD` variables are shown with their respective attributes. The `tmp` variable is shown with its value, which is a `DigitalVideoDisc` object with `id=21`.

Result:

The screenshot shows an IDE with a Java file named `TestPassingParameter.java`. The code defines a `DigitalVideoDisc` class and a `TestPassingParameter` class. The `TestPassingParameter` class has a `main` method that creates two `DigitalVideoDisc` objects, `jungleDVD` and `cinderellaDVD`, and calls `swap` and `changeTitle` methods. The `swap` method swaps the `id` values of two objects, and the `changeTitle` method changes the `title` of a `DigitalVideoDisc` object. The `main` method prints the titles of the objects before and after the swaps and title changes. The IDE also shows the `Variables` tab, which displays the state of the program at the current line of execution. The `Variables` tab shows that the `println()` method returned `(No explicit return value)`, and the `args` array is `String[0] (id=20)`. The `jungleDVD` object is a `DigitalVideoDisc` object with `id=21`, `category=null`, `cost=0.0`, `director=null`, `id=1`, and `length=0`. The `cinderellaDVD` object is a `DigitalVideoDisc` object with `id=24`, `category=null`, `cost=0.0`, `director=null`, `id=2`, and `length=0`. The `Ball` object is a `DigitalVideoDisc` object with `id=32`, `category=null`, `cost=0.0`, `director=null`, `id=2`, and `length=0`. The `Console` tab shows the output of the program, which is `jungle dvd title: Ball`.

```
1 package hust.solt.dsai.test.dsai;
2 import hust.solt.dsai.test.dsai.DigitalVideoDisc;
3
4 public class TestPassingParameter {
5     public static void main(String[] args) {
6         DigitalVideoDisc jungleDVD = new DigitalVideoDisc("Jungle");
7         DigitalVideoDisc cinderellaDVD = new DigitalVideoDisc("Cinderella");
8
9         swap(jungleDVD, cinderellaDVD);
10        System.out.println("jungle dvd title: " + jungleDVD.getTitle());
11
12        // swap(jungleDVD, cinderellaDVD);
13        System.out.println("jungle dvd title: " + jungleDVD.getTitle());
14
15        System.out.println("cinderella dvd title: " + cinderellaDVD.getTitle());
16
17        changeTitle(jungleDVD, cinderellaDVD.getTitle());
18        System.out.println("jungle dvd title: " + jungleDVD.getTitle());
19    }
20
21    public static void swap(Object o1, Object o2) {
22        Object tmp = o1;
23        o1 = o2;
24        o2 = tmp;
25    }
26
27    // Using a wrapper class to hold the references
28    public static void swap(DigitalVideoDisc dvd1, DigitalVideoDisc dvd2) {
29        String tempTitle = dvd1.getTitle();
30        dvd1.setTitle(dvd2.getTitle());
31        dvd2.setTitle(tempTitle);
32    }
33
34    public static void changeTitle(DigitalVideoDisc dvd, String title) {
35        String oldTitle = dvd.getTitle();
36        dvd.setTitle(title);
37        dvd = new DigitalVideoDisc(oldTitle);
38    }
39 }
40
```

Variables

Name	Value
println() returned	(No explicit return value)
args	String[0] (id=20)
jungleDVD	DigitalVideoDisc (id=21)
category	null
cost	0.0
director	null
id	1
length	0
title	"Ball" (id=37)
cinderellaDVD	DigitalVideoDisc (id=24)
category	null
cost	0.0
director	null
id	2
length	0
title	"Cinderella" (id=32)

Console

```
TestPassingParameter [Java Application] C:\Program Files\Eclipse Adoptium\jdk-17.0.12-hotspot\bin\javaw.exe (Nov 24, 2024, 3:17:01 PM) [pid: 16588]
jungle dvd title: Ball
```

5. Classifier Member and Instance Member

The screenshot shows an IDE with a Java file named `DigitalVideoDisc.java`. The code defines the `DigitalVideoDisc` class, which has private fields for `title`, `category`, `director`, `length`, `cost`, and `id`. It also has a static field `nbDigitalVideoDiscs`. The class has four constructors: a no-argument constructor, a constructor with `title`, a constructor with `title`, `category`, and `cost`, and a constructor with `title`, `category`, `director`, and `cost`. The `nbDigitalVideoDiscs` field is incremented for each instance of the class. The `getTitle` method returns the `title` of the object.

```
1 private String title;
2 private String category;
3 private String director;
4 private int length;
5 private float cost;
6
7 private static int nbDigitalVideoDiscs = 0;
8
9 private int id;
10
11 public DigitalVideoDisc(String title) {
12     super();
13     this.title = title;
14
15     nbDigitalVideoDiscs += 1;
16     this.id = nbDigitalVideoDiscs;
17 }
18
19 public DigitalVideoDisc(String title, String category, float cost) {
20     super();
21     this.title = title;
22     this.category = category;
23     this.cost = cost;
24
25     nbDigitalVideoDiscs += 1;
26     this.id = nbDigitalVideoDiscs;
27 }
28
29 public DigitalVideoDisc(String title, String category, String director, float cost) {
30     super();
31     this.title = title;
32     this.category = category;
33     this.director = director;
34     this.cost = cost;
35
36     nbDigitalVideoDiscs += 1;
37     this.id = nbDigitalVideoDiscs;
38 }
39
40 public DigitalVideoDisc(String title, String category, String director, int length, float cost) {
41     super();
42     this.title = title;
43     this.category = category;
44     this.director = director;
45     this.length = length;
46     this.cost = cost;
47
48     nbDigitalVideoDiscs += 1;
49     this.id = nbDigitalVideoDiscs;
50 }
```

6. Open the Cart class

```

38     }
39     this.id = nbDigitalVideoDiscs;
40 }
41 public DigitalVideoDisc(String title, String category, String director, int length, float cost) {
42     super();
43     this.title = title;
44     this.category = category;
45     this.director = director;
46     this.length = length;
47     this.cost = cost;
48     nbDigitalVideoDiscs += 1;
49     this.id = nbDigitalVideoDiscs;
50 }
51
52 public boolean isMatch(String title) {
53     return this.title.equals(title);
54 }
55
56 public String toString() {
57     return "DVD" + "-" + this.title + "-" + this.category + "-" + this.director + "-" + String.valueOf(this.length) + ": " + String.valueOf(this.cost) + "$";
58 }
59
60

```

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

+ The method should return a String.

```

43         qtyOrdered += 1;
44         System.out.println("Added");
45     }
46 }
47
48 public void printOrders() {
49     for (int i = 0; i < qtyOrdered; i++) {
50         System.out.println(itemsOrdered[i].toString());
51     }
52 }
53
54 public void searchById(int id) {
55     boolean found = false;
56     for (int i = 0; i < qtyOrdered; i++) {
57         if (itemsOrdered[i].getId() == id) {
58             System.out.println("DVD found: " + itemsOrdered[i].toString());
59             found = true;
60             break;
61         }
62     }
63
64     if (!found) {
65         System.out.println("No DVD found with ID: " + id);
66     }
67 }
68
69 public void searchByTitle(String title) {
70     boolean found = false;
71     for (int i = 0; i < qtyOrdered; i++) {
72         if (itemsOrdered[i].isMatch(title)) {
73             System.out.println("DVD found: " + itemsOrdered[i].toString());
74             found = true;
75             break;
76         }
77     }
78
79     if (!found) {
80         System.out.println("No DVD found with title: " + title);
81     }
82 }
83

```

```
1 package hust.soict.dsai.test.cart;
2 import hust.soict.dsai.aims.cart.Cart;
3
4
5 public class CartTest {
6     public static void main(String[] args) {
7         //Create a new cart
8         Cart cart = new Cart();
9
10        //Create new dvd objects and add them to the cart
11        DigitalVideoDisc dvd1 = new DigitalVideoDisc("The Lion King",
12            "Animation", "Roger Allers", 87, 19.95f);
13        cart.addDigitalVideoDisc(dvd1);
14
15        DigitalVideoDisc dvd2 = new DigitalVideoDisc("Star Wars",
16            "Science Fiction", "George Lucas", 87, 24.95f);
17        cart.addDigitalVideoDisc(dvd2);
18
19        DigitalVideoDisc dvd3 = new DigitalVideoDisc("Aladin",
20            "Animation", 18.99f);
21        cart.addDigitalVideoDisc(dvd3);
22
23        //Test the print method
24        cart.printOrders();
25
26        //Test the search method
27        cart.searchByTitle("Aladin");
28        cart.searchByTitle("Ball");
29
30        cart.searchById(1);
31        cart.searchById(10);
32    }
33 }
```

Console

```
Added
Added
Added
DVD-The Lion King-Animation-Roger Allers-87: 19.95$
DVD-Star Wars-Science Fiction-George Lucas-87: 24.95$
DVD-Aladin-Animation-null-0: 18.99$
DVD found: DVD-Aladin-Animation-null-0: 18.99$
No DVD found with title: Ball
DVD found: DVD-The Lion King-Animation-Roger Allers-87: 19.95$
No DVD found with ID: 10
```

7. Implement the Store class

```
1 package hust.soict.dsai.aims.store;
2
3 import hust.soict.dsai.aims.disc.DigitalVideoDisc;
4 import java.util.ArrayList;
5
6 public class Store {
7     private ArrayList<DigitalVideoDisc> itemsInStore;
8
9     public Store() {
10         itemsInStore = new ArrayList<>();
11     }
12
13     public void addDvd(DigitalVideoDisc dvd) {
14         if (dvd != null) {
15             itemsInStore.add(dvd);
16             System.out.println("DVD added: " + dvd.getTitle());
17         } else {
18             System.out.println("Cannot add null DVD.");
19         }
20     }
21
22     public void removeDvd(DigitalVideoDisc dvd) {
23         if (itemsInStore.contains(dvd)) {
24             itemsInStore.remove(dvd);
25             System.out.println("DVD removed: " + dvd.getTitle());
26         } else {
27             System.out.println("DVD not found in the store.");
28         }
29     }
30 }
31
```



```
1 package hust.soict.dsai.test.store;
2
3 import hust.soict.dsai.aims.disc.DigitalVideoDisc;
4
5
6 public class StoreTest {
7     public static void main(String[] args) {
8         Store store = new Store();
9
10        // Create DigitalVideoDisc instances
11        DigitalVideoDisc dvd1 = new DigitalVideoDisc("Inception");
12        DigitalVideoDisc dvd2 = new DigitalVideoDisc("Interstellar");
13        DigitalVideoDisc dvd3 = new DigitalVideoDisc("The Matrix");
14
15        // Test adding DVDs
16        System.out.println("Testing addDVD method:");
17        store.addDvd(dvd1);
18        store.addDvd(dvd2);
19        store.addDvd(null);
20
21        // Test removing DVDs
22        System.out.println("\nTesting removeDVD method:");
23        store.removeDvd(dvd2);
24        store.removeDvd(dvd3);
25    }
26 }
27
```

Console

<terminated> StoreTest (2) [Java Application] C:\Program Files\Eclipse Adoptium\jdk-17.0.12-hotspot\bin\javaw.exe (Nov 24, 2024, 6:02:22 PM - 6:02:22 PM) [pid: 21108]

Testing addDVD method:
DVD added: Inception
DVD added: Interstellar
Cannot add null DVD.

Testing removeDVD method:
DVD removed: Interstellar
DVD not found in the store.

9. String, StringBuilder and StringBuffer

```
1 package hust.soict.dsai.garbage;
2
3 import java.util.Random;
4
5 public class ConcatenationInLoops {
6     public static void main(String[] args) {
7         Random r = new Random(123);
8         long start = System.currentTimeMillis();
9         String s = "";
10        for (int i = 0; i < 65536; i++)
11            s += r.nextInt(2);
12        System.out.println(System.currentTimeMillis() - start); // This prints roughly 4500.
13
14        r = new Random(123);
15        start = System.currentTimeMillis();
16        StringBuilder sb = new StringBuilder();
17        for (int i = 0; i < 65536; i++)
18            sb.append(r.nextInt(2));
19        s = sb.toString();
20        System.out.println(System.currentTimeMillis() - start); // This prints 5.
21    }
22 }
23
```

Console

<terminated> ConcatenationInLoops [Java Application] C:\Users\Admin\p2\pool\plugins\org.eclipse.justi.openjdk.hotspot.jre.full.win32.x86_64_23.0.0.v20240919-1706\jre\bin\javaw.exe (Nov 24, 2024, 5:24:56 PM - 5:24:57 PM) [pid: 18656]

347
2

String concatenation

```
1 package hust.socit.dsai.garbage;
2
3 import java.io.*;
4 import java.nio.file.*;
5
6 public class GarbageCreator {
7     public static void main(String[] args) {
8         // Specify a large file path - you can use any large text file
9         String filePath = "large_file.txt";
10
11         try {
12             // Create a large file for testing if it doesn't exist
13             if (!Files.exists(Paths.get(filePath))) {
14                 createLargeFile(filePath);
15             }
16
17             System.out.println("Starting to read file with String concatenation...");
18             long startTime = System.currentTimeMillis();
19
20             // Read file character by character with String concatenation
21             try (FileReader fr = new FileReader(filePath)) {
22                 String content = "";
23                 int character;
24                 int charCount = 0;
25
26                 while ((character = fr.read()) != -1) {
27                     content += (char) character; // This creates many String objects in memory
28                     charCount++;
29
30                     if (charCount % 1000000 == 0) {
31                         System.out.println("Read " + charCount + " characters...");
32                     }
33                 }
34
35                 System.out.println("Final string length: " + content.length());
36             }
37
38             long endTime = System.currentTimeMillis();
39             System.out.println("Time taken: " + (endTime - startTime) + " ms");
40
41         } catch (OutOfMemoryError e) {
42             System.out.println("Out of Memory Error occurred!");
43             System.out.println("Error: " + e.getMessage());
44         }
45     }
46
47     private static void createLargeFile(String filePath) throws IOException {
48         try (FileWriter writer = new FileWriter(filePath)) {
49             // Create a 4GB file
50             for (int i = 0; i < 100000000; i++) {
51                 writer.write("This is a test line to create a large file.\n");
52             }
53         }
54     }
55 }
```

GarbageCreator [Java Application] C:\Users\Admin\p2\pool\plugins\org.eclipse.justi.openjdk.hotspot.jre.full.win32.x86_64.23.0.0.v20240919-1706\jre\bin\javaw.exe (Nov 24, 2024, 5:29:18 PM) [pid: 26444]

Starting to read file with String concatenation...
Read 1000000 characters...
Read 2000000 characters...

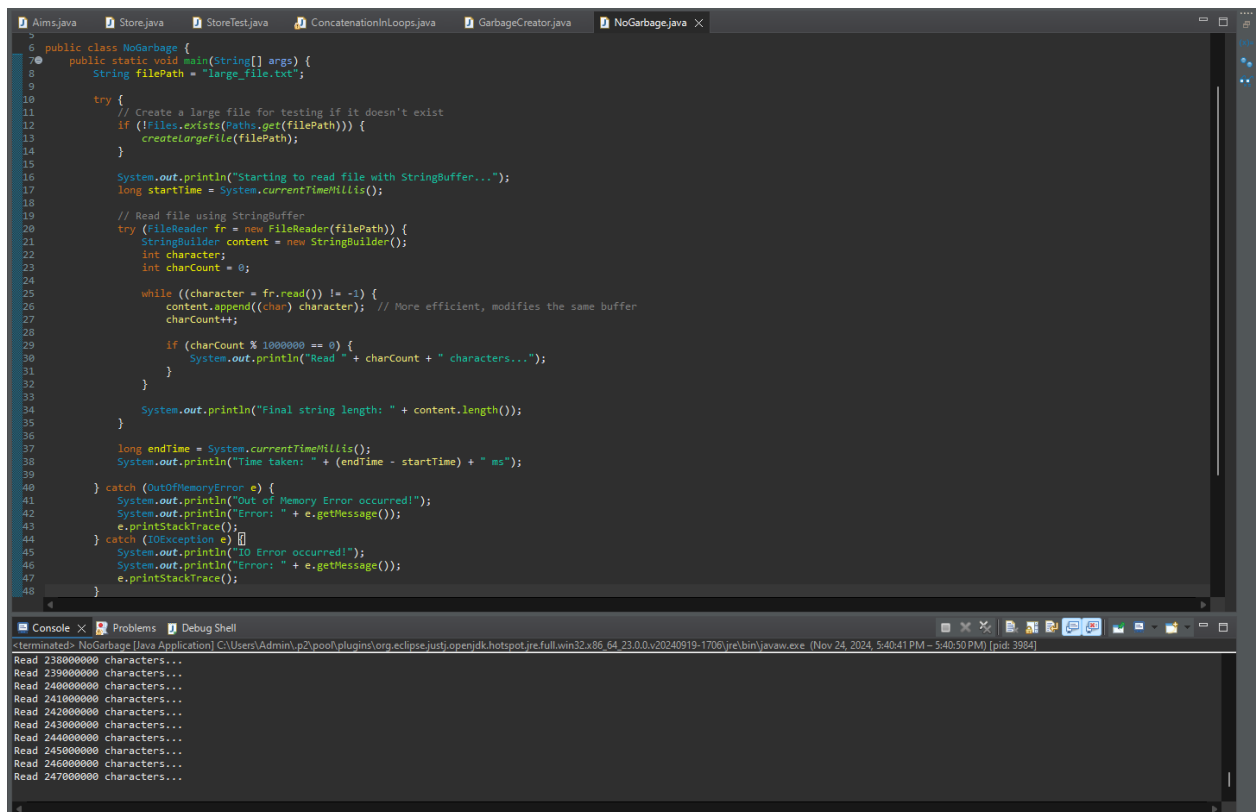
⇒ Very slow. Take so much time to read the file\

Using stringBuffer & string Builder:

```
17 long startTime = System.currentTimeMillis();
18
19 // Read file using StringBuffer
20 try (FileReader fr = new FileReader(filePath)) {
21     StringBuffer content = new StringBuffer();
22     int character;
23     int charCount = 0;
24
25     while ((character = fr.read()) != -1) {
26         content.append((char) character); // More efficient, modifies the same buffer
27         charCount++;
28
29         if (charCount % 1000000 == 0) {
30             System.out.println("Read " + charCount + " characters...");
31         }
32     }
33
34     System.out.println("Final string length: " + content.length());
35 }
36
37 long endTime = System.currentTimeMillis();
38 System.out.println("Time taken: " + (endTime - startTime) + " ms");
39
40 } catch (OutOfMemoryError e) {
41     System.out.println("Out of Memory Error occurred!");
42     System.out.println("Error: " + e.getMessage());
43     e.printStackTrace();
44 } catch (IOException e) {
45     System.out.println("IO Error occurred!");
46     System.out.println("Error: " + e.getMessage());
47     e.printStackTrace();
48 }
49
50 private static void createLargeFile(String filePath) throws IOException {
51     try (FileWriter writer = new FileWriter(filePath)) {
52         // Create a 4GB file
53         for (int i = 0; i < 100000000; i++) {
54             writer.write("This is a test line to create a large file.\n");
55         }
56     }
57 }
58 }
59
60 }
```

sterminated: NoGarbage [Java Application] C:\Users\Admin\p2\pool\plugins\org.eclipse.justi.openjdk.hotspot.jre.full.win32.x86_64.23.0.0.v20240919-1706\jre\bin\javaw.exe (Nov 24, 2024, 5:34:50 PM - 5:36:30 PM) [pid: 21328]

Read 2146000000 characters...
Read 2147000000 characters...
Out of Memory Error occurred!
Error: Requested array size exceeds VM limit
java.lang.OutOfMemoryError: Requested array size exceeds VM limit
at java.base/java.util.Arrays.copyOf(Arrays.java:3540)
at java.base/java.lang.AbstractStringBuilder.ensureCapacityInternal(AbstractStringBuilder.java:245)
at java.base/java.lang.AbstractStringBuilder.append(AbstractStringBuilder.java:813)
at java.base/java.lang.StringBuffer.append(StringBuffer.java:425)
at hust.socit.dsai.garbage.NoGarbage.main(NoGarbage.java:26)



```
5 public class NoGarbage {
6     public static void main(String[] args) {
7         String filePath = "large_file.txt";
8
9         try {
10             // Create a large file for testing if it doesn't exist
11             if (!Files.exists(Paths.get(filePath))) {
12                 createLargeFile(filePath);
13             }
14
15             System.out.println("Starting to read file with StringBuffer...");
16             long startTime = System.currentTimeMillis();
17
18             // Read file using StringBuffer
19             try (FileReader fr = new FileReader(filePath)) {
20                 StringBuilder content = new StringBuilder();
21                 int character;
22                 int charCount = 0;
23
24                 while ((character = fr.read()) != -1) {
25                     content.append((char) character); // More efficient, modifies the same buffer
26                     charCount++;
27
28                     if (charCount % 1000000 == 0) {
29                         System.out.println("Read " + charCount + " characters...");
30                     }
31                 }
32
33                 System.out.println("Final string length: " + content.length());
34             }
35
36             long endTime = System.currentTimeMillis();
37             System.out.println("Time taken: " + (endTime - startTime) + " ms");
38
39         } catch (OutOfMemoryError e) {
40             System.out.println("Out of Memory Error occurred!");
41             System.out.println("Error: " + e.getMessage());
42             e.printStackTrace();
43         } catch (IOException e) {
44             System.out.println("IO Error occurred!");
45             System.out.println("Error: " + e.getMessage());
46             e.printStackTrace();
47         }
48     }
49 }
```

Console Output:

```
Read 238000000 characters...
Read 239000000 characters...
Read 240000000 characters...
Read 241000000 characters...
Read 242000000 characters...
Read 243000000 characters...
Read 244000000 characters...
Read 245000000 characters...
Read 246000000 characters...
Read 247000000 characters...
```

- It is significantly faster and more efficient.
- The main difference between StringBuffer and StringBuilder is thread safety. In most cases, I believe StringBuilder is the preferred choice unless thread safety is a concern.
- Although it still reaches the limit, this is due to the file I created being very large (around 4GB). → It is definitely an improvement over string concatenation.