Report Lab 03

Name: Bùi Công Minh

Student ID: 20235601

1. Method overloading

a. Overloading by differing types of parameter

```
public void addDigitalVideoDisc(DigitalVideoDisc[] dvdlist) {
    for (DigitalVideoDisc disc: dvdlist) {
        if (qtyOrdered < MAX_NUMBERS_ORDERED) {
            itemsOrdered[qtyOrdered] = disc;
            qtyOrdered++;
            System.out.println("The disc has been added");
        }
        else {
            System.out.println("The cart is full");
            break;
        }
    }
}
```

Figure 1. Overloading by differing types of parameter

b. Overloading by differing the number of parameters

```
public void addDigitalVideoDisc(DigitalVideoDisc dvd1,DigitalVideoDisc dvd2) {
    if (qtyOrdered <= MAX_NUMBERS_ORDERED - 2) {
        itemsOrdered[qtyOrdered] = dvd1;
        qtyOrdered++;
        itemsOrdered[qtyOrdered] = dvd2;
        qtyOrdered++;
        System.out.println("The discs have been added");
    }
    else if (qtyOrdered <= MAX_NUMBERS_ORDERED - 1) {
        itemsOrdered[qtyOrdered] = dvd1;
        qtyOrdered++;
        System.out.println("The first disc has been added. The other one cannot be added"
                + " since the cart is full");
    }
    else {
        System.out.println("The cart is full");
    }
}
```

Figure 2. Overloading by differing the number of parameters

2. Passing parameters

```java
public class TestPassingParameter {

    public static void main(String[] args) {
        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);
    }
}
```

Figure 2: Original source code of TestPassingParameter

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

Figure 3: Results of the given source code

Questions:

- After the call of **`swap(jungleDVD, cinderellaDVD)`** why does the title of these two objects still remain?
  Answer: When the swap function is called, this function creates mirror references of two objects jungleDVD and cinderellaDVD (o1 and o2). Therefore, the changes to o1 and o2 do not affect the original ones. This means that the title of two objects still remain after calling swap function.

- After the call of `changeTitle(jungleDVD, cinderellaDVD.getTitle())` why is the title of the JungleDVD changed?
  Answer:
  When the function is called, a new reference dvd is created and point to jungleDVD. The change of the title in reference dvd with the method setTitle leads to the change of the object jungleDVD.
  After that, we assigned the new object to dvd. However, jungleDVD is not affected since dvd is now pointing to the newly created object.

Modified source code:

```
public class TestPassingParameter {

    public static void main(String[] args) {
        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(DigitalVideoDisc o1, DigitalVideoDisc o2) {
        DigitalVideoDisc tmp = new DigitalVideoDisc("");
        tmp.setTitle(o1.getTitle());
        o1.setTitle(o2.getTitle());
        o2.setTitle(tmp.getTitle());
    }

    public static void changeTitle(DigitalVideoDisc dvd, String title) {
        String oldTitle = dvd.getTitle();
        dvd.setTitle(title);
        dvd = new DigitalVideoDisc(oldTitle);
    }
}
```

Figure 4: Modified source code for TestPassingParameter

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

Figure 5: The results

3. Use debug run

a. Setting, deleting, & deactivae breakpoints

```java
package hust.soict.cybersecurity.test.disc;
import hust.soict.cybersecurity.aims.disc.DigitalVideoDisc;

public class TestPassingParameter {

    public static void main(String[] args) {
        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);
    }
}

```

Figure 6: The breakpoint is activated

```java
package hust.soict.cybersecurity.test.disc;
import hust.soict.cybersecurity.aims.disc.DigitalVideoDisc;

public class TestPassingParameter {

    public static void main(String[] args) {
        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);
    }
}
```

Figure 7: The breakpoint is deactivated

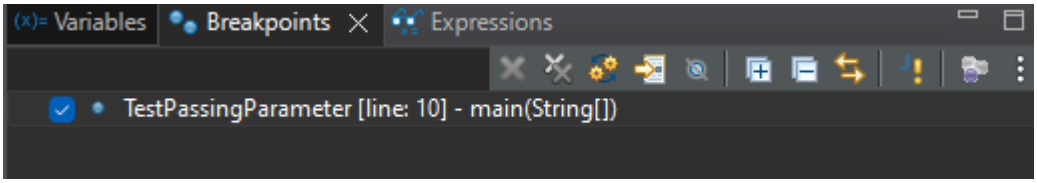

Figure 8: Breakpoints View

b. Run in debug mode

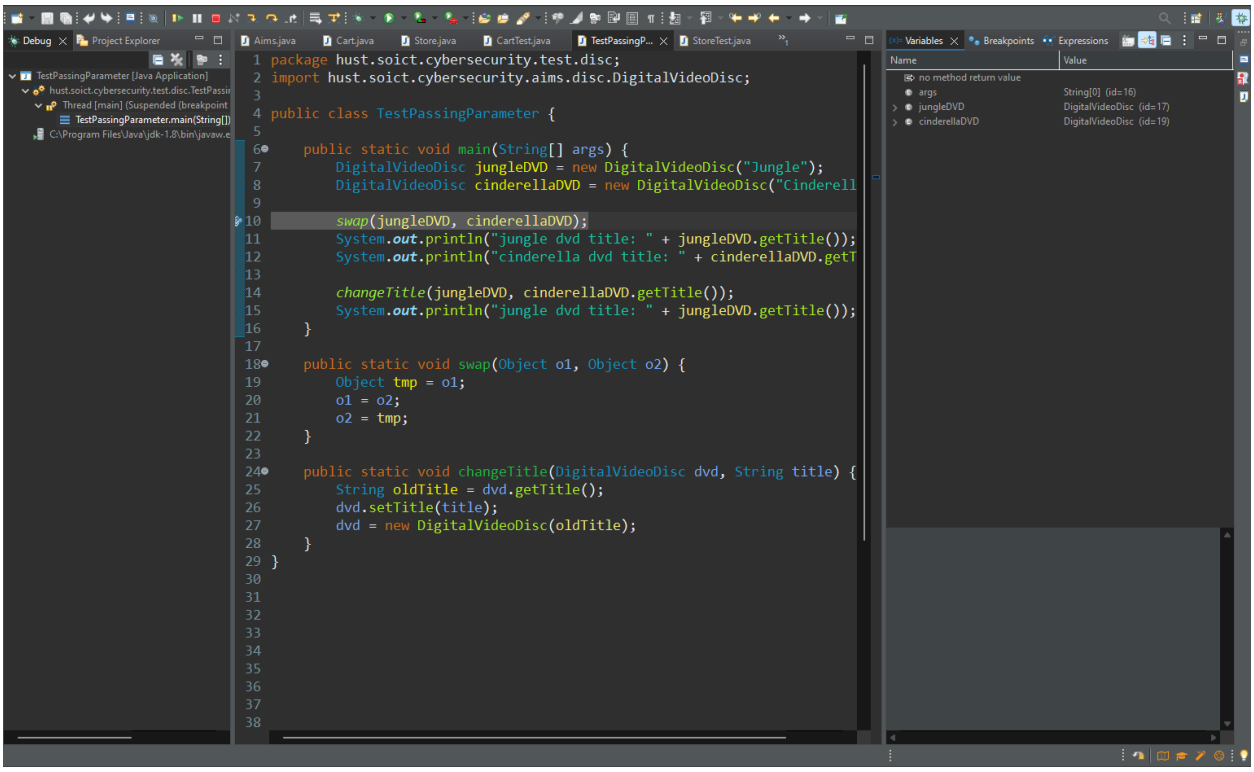

Figure 9: Debug mode

c. Step Into, Step Over, Step Return, Resume

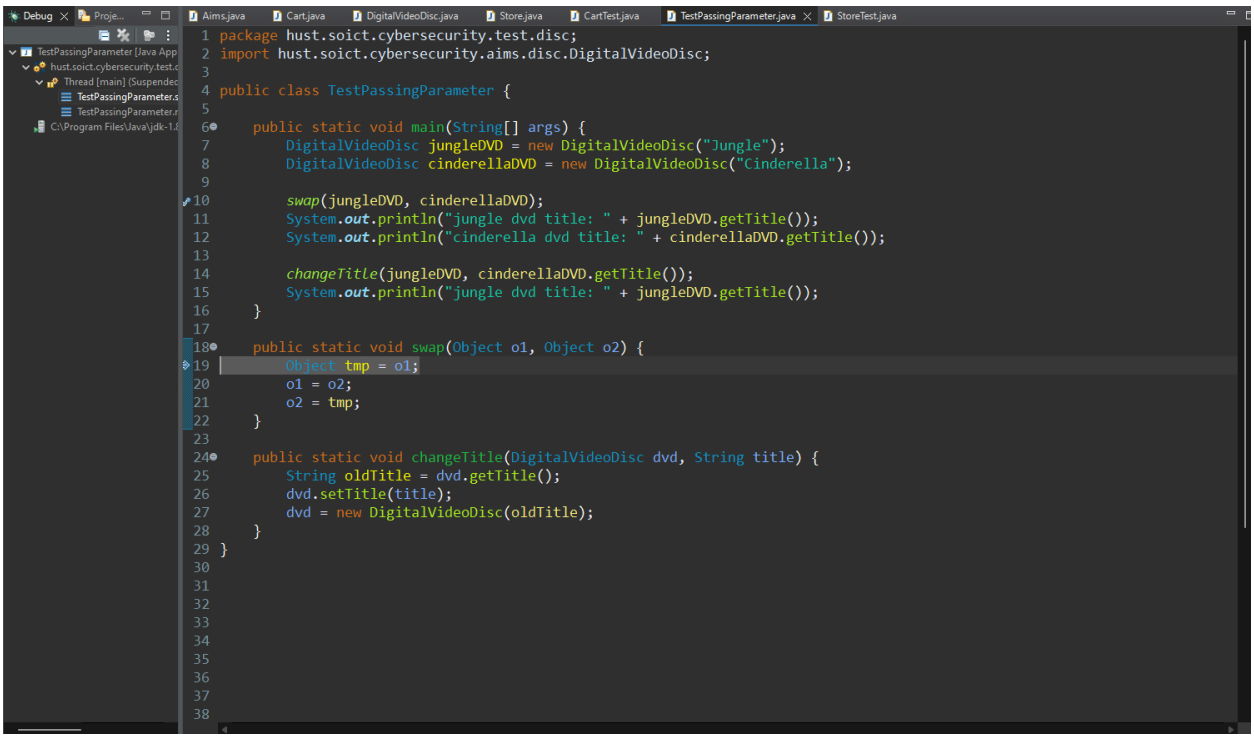

Figure 10: Step Into

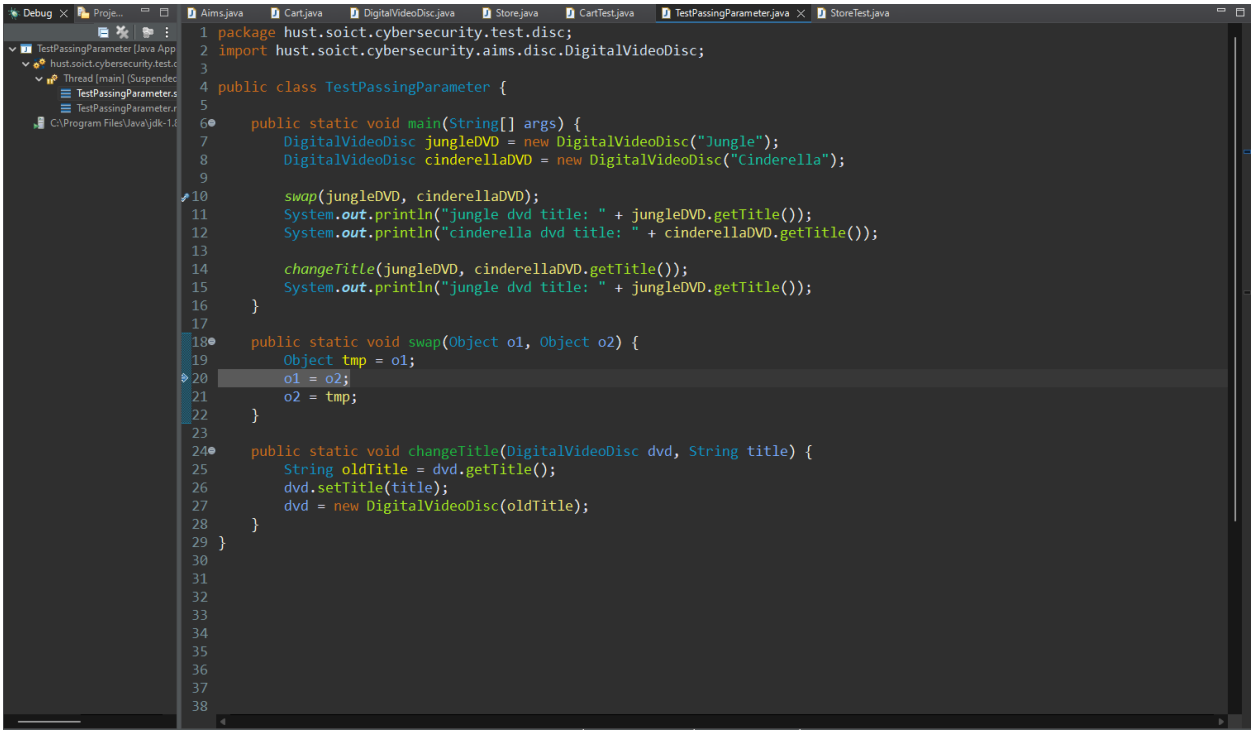

Figure 11: Step Over

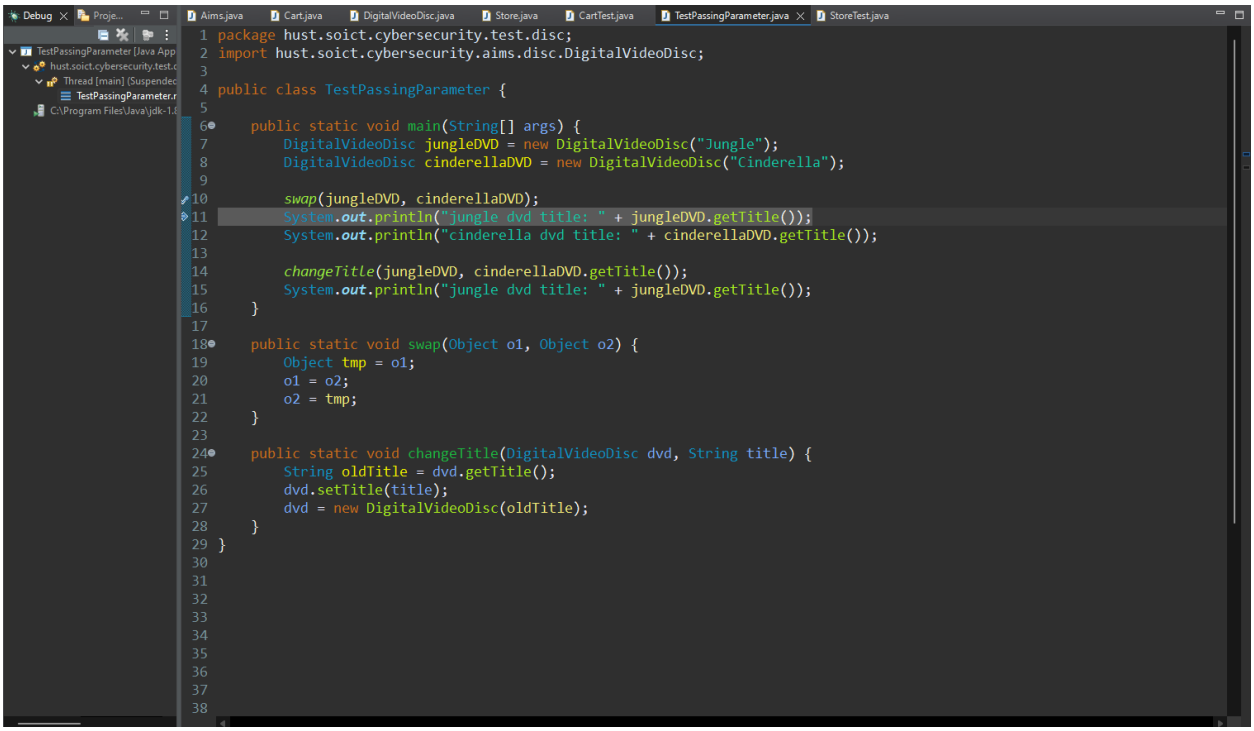

Figure 12: Step Return

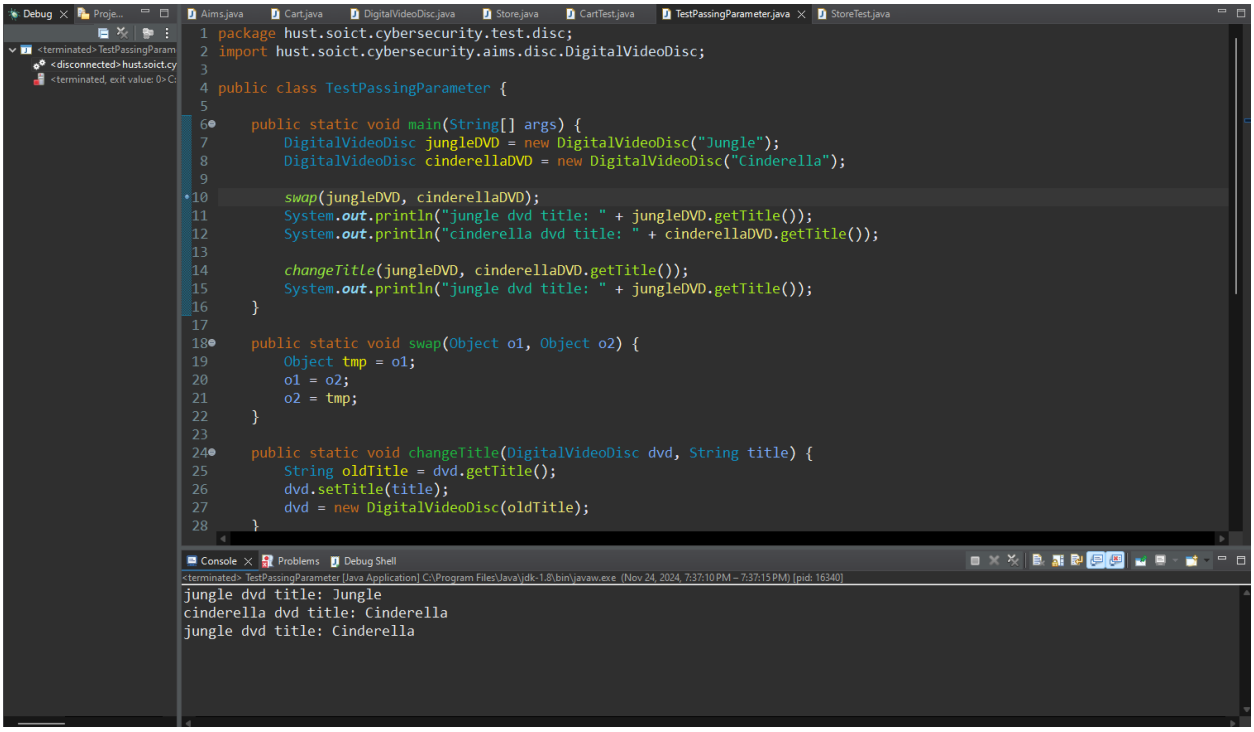

Figure 13: Resume

d. Investigate value of variables

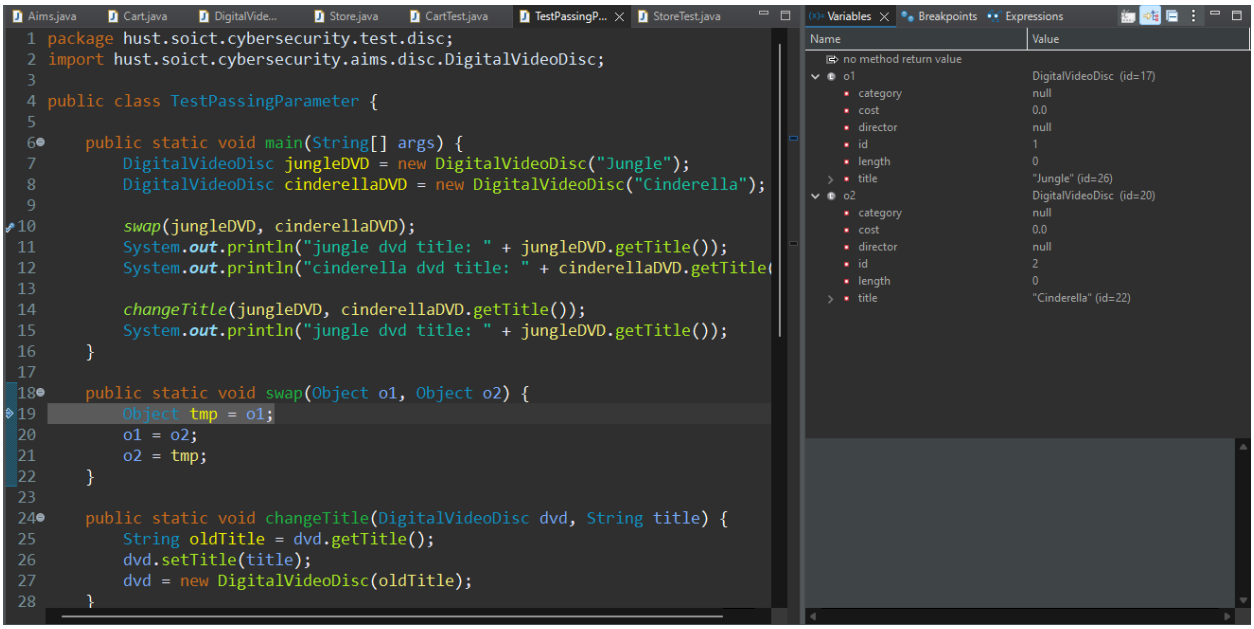

Figure 14: Variables View

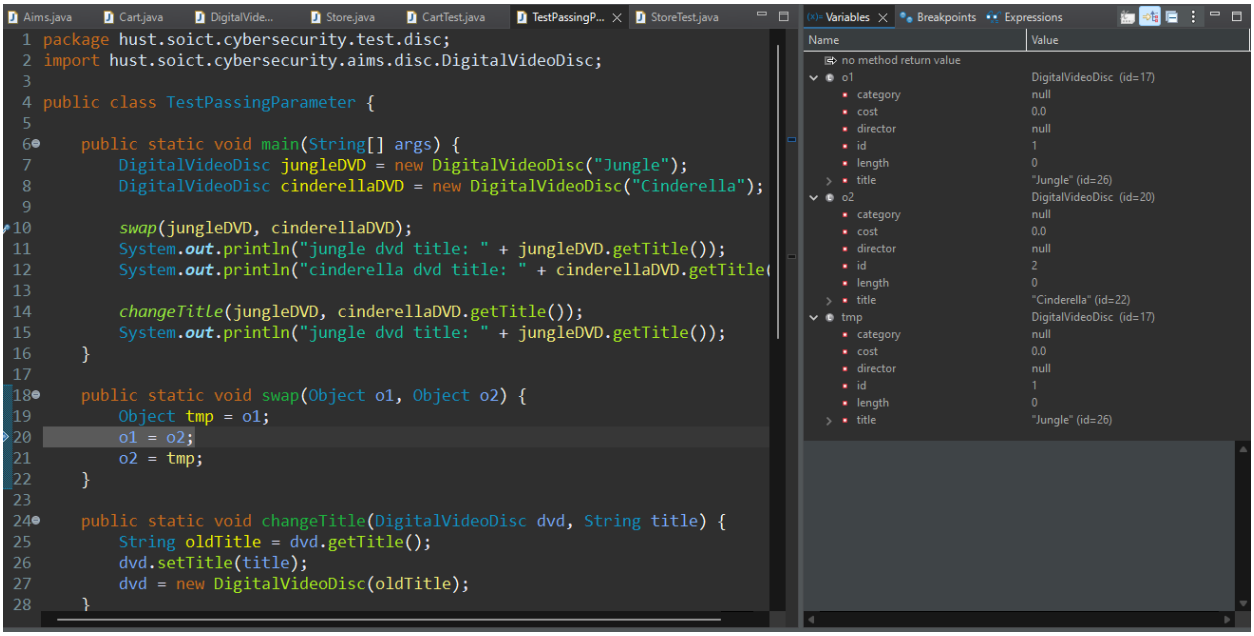

Figure 15: Step Over line 19

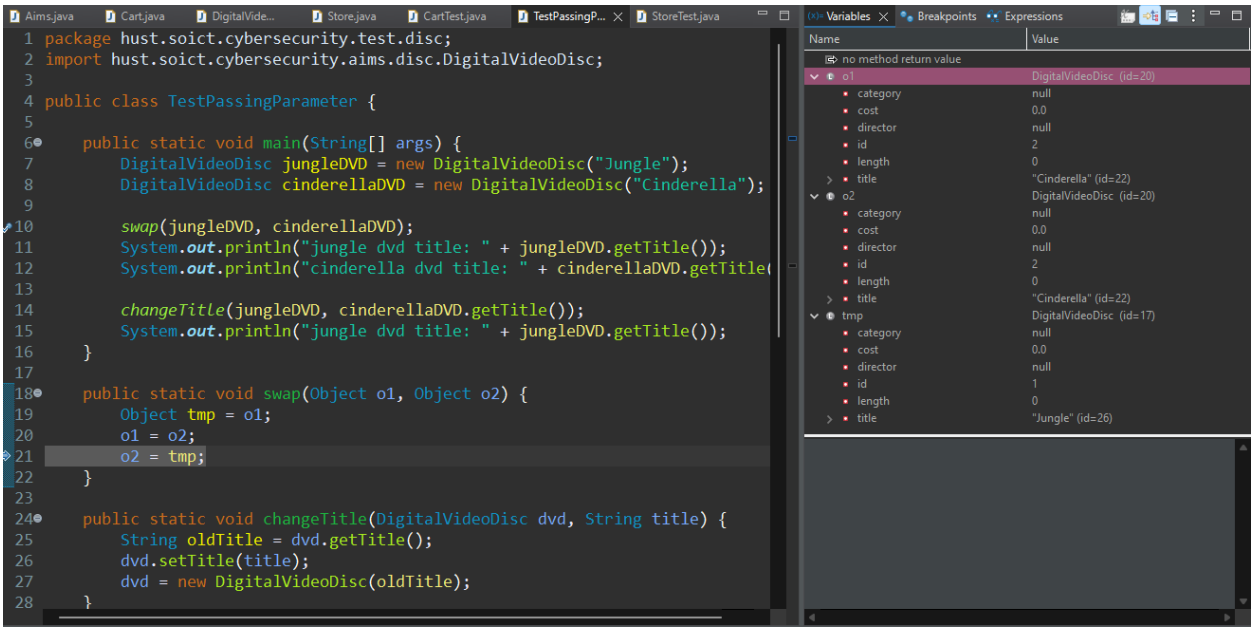

Figure 16: Step Over line 20

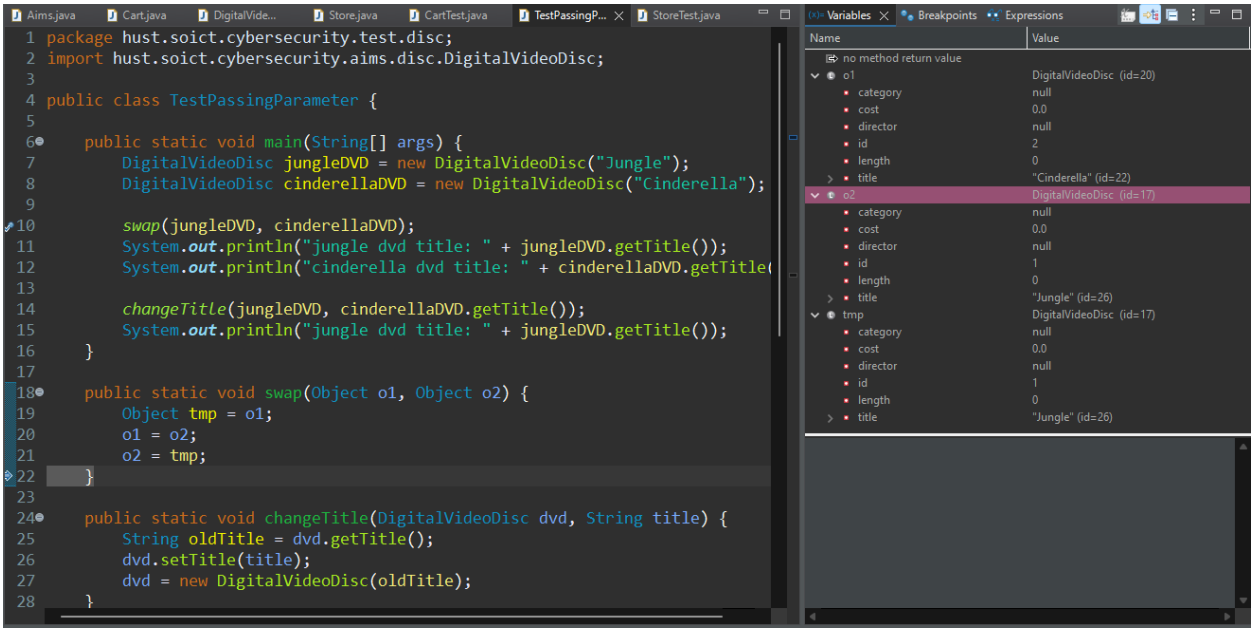

Figure 17: Step Over line 21

e. Change value of variables

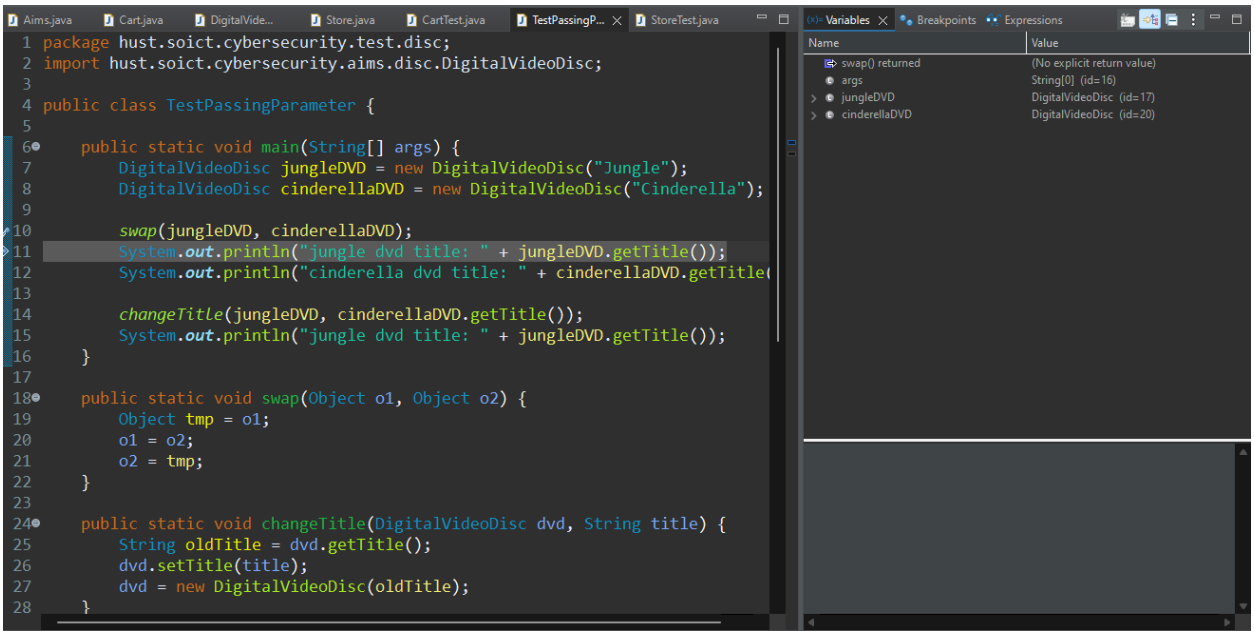

Figure 18: Step Return to the main function

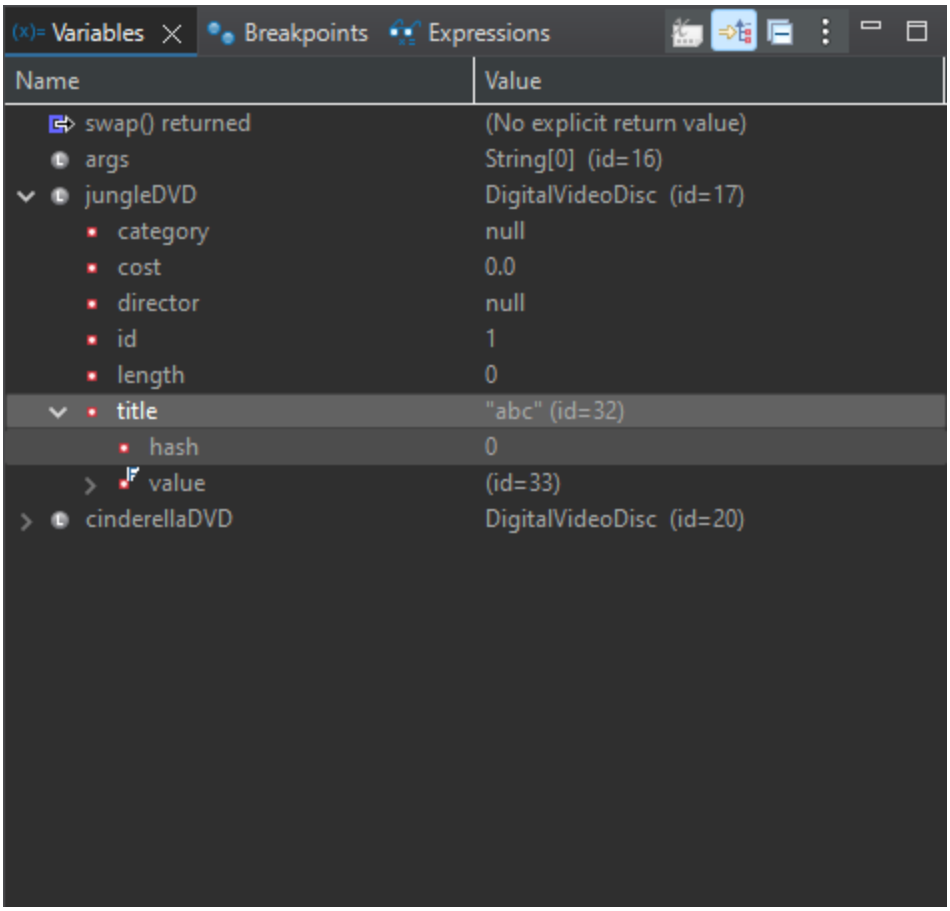

Figure 19: Change title of jungleDVD

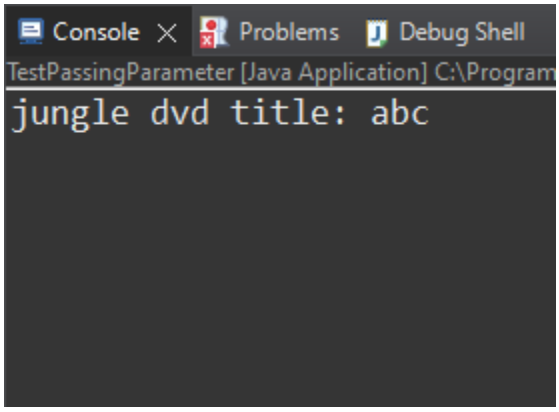

Figure 20: Result

4. Classifier Member and Instance Member

```java
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;
```

Figure 21: Create new attributes id and nbDigitalVideoDiscs

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

Figure 22: Updating nbDigitalVideoDiscs and assigning id in each constructor

5. Cart Class

Writing method to print items in cart

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

Figure 23: Method toString in class DigitalVideoDisc

```
public void print() {
    total = 0;
    System.out.println("**********CART**********");
    System.out.println("Ordered Items:");
    for (int i = 0; i < qtyOrdered; i++) {
        System.out.println((i + 1) + ". DVD - " + itemsOrdered[i].toString());
    }
    System.out.println("Total cost: " + totalCost());
    System.out.println("**************************************************");
}
```

Figure 24: Method to print items in cart

Writing methods for searching:

```
public boolean isMatch(String title) {
    return this.title.equals(title);
}
```

Figure 25: Method isMatch for title

```
public void searchByID(int id) {
    boolean found = false;
    for (int i = 0; i < qtyOrdered; i++) {
        if (itemsOrdered[i].getID() == id) {
            System.out.println("DVD found: " + itemsOrdered[i].toString());
            found = true;
            break;
        }
    }
    if (!found) {
        System.out.println("There are no DVDs that match your serach");
    }
}
```

Figure 26: Method searchByID

```java
public void searchByTitle(String title) {
    boolean found = false;
    for (int i = 0; i < qtyOrdered; i++) {
        if (itemsOrdered[i].isMatch(title)) {
            System.out.println("DVD found: " + itemsOrdered[i].toString());
            found = true;
            break;
        }
    }
    if (!found) {
        System.out.println("There are no DVDs that match your serach");
    }
}
```

Figure 27: Method searchByTitle

```java
public class CartTest {
    public static void main(String[] args) {
        Cart cart = new Cart();

        DigitalVideoDisc dvd1 = new DigitalVideoDisc("The Lion King", "Animation",
                "Roger Allers", 87, 19.95f);
        cart.addDigitalVideoDisc(dvd1);

        DigitalVideoDisc dvd2 = new DigitalVideoDisc("Star Wars", "Science Fiction",
                "George Lucas", 87, 24.95f);
        cart.addDigitalVideoDisc(dvd2);
        cart.removeDigitalVideoDisc(dvd2);

        DigitalVideoDisc dvd3 = new DigitalVideoDisc("Aladin", "Animation", 18.99f);
        cart.addDigitalVideoDisc(dvd3);

        cart.print();

        cart.searchByID(1);
        cart.searchByID(3);

        cart.searchByTitle("The Lion King");
        cart.searchByTitle("Venom");
    }
}
```

Figure 28: Testing search methods in CartTest

6. Store Class

```java
public class Store {
    DigitalVideoDisc itemsInStore[] = new DigitalVideoDisc[10000000];
    private int storeQty = 0;
    public void addDVD(DigitalVideoDisc disc) {
        itemsInStore[storeQty] = disc;
        storeQty++;
    }

    public void removeDVD(DigitalVideoDisc disc) {
        int ind = 0;
        for (int i = 0; i < storeQty; i++) {
            if(disc.equals(itemsInStore[i])) {
                ind = i;
            }
        }
        for (int i = ind; i < storeQty - 1; i++) {
            itemsInStore[i] = itemsInStore[i + 1];
        }
        itemsInStore[storeQty] = null;
        storeQty--;
    }

    public void print() {
        System.out.println("**********CART**********");
        System.out.println("Store:");
        for (int i = 0; i < storeQty; i++) {
            System.out.println((i + 1) + ". DVD - " + itemsInStore[i].toString());
        }
        System.out.println("**************************************************");
    }
}
```

Figure 29: Class Store

```java
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);
        store.addDVD(dvd1);

        DigitalVideoDisc dvd2 = new DigitalVideoDisc("Star Wars", "Science Fiction",
                "George Lucas", 87, 24.95f);
        store.addDVD(dvd2);
        store.removeDVD(dvd2);

        DigitalVideoDisc dvd3 = new DigitalVideoDisc("Aladin", "Animation", 18.99f);
        store.addDVD(dvd3);

        store.print();
    }
}
```

Figure 30: Class StoreTest

7. Reorganize projects

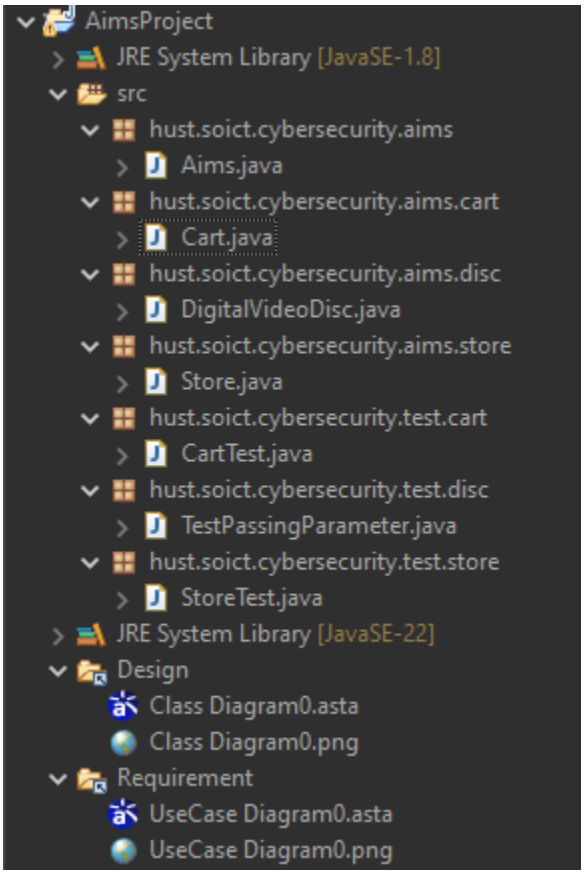

Figure 31: Structure for AimsProject

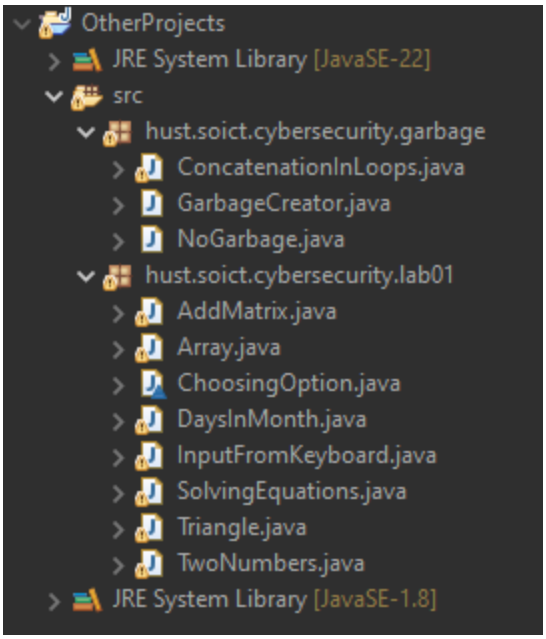

Figure 32: Structure for OtherProjects

8. String, StringBuilder and StringBuffer

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

import java.util.Random;

public class ConcatenationInLoops {
    public static void main(String[] args) {
        Random r = new Random(123);
        long start = System.currentTimeMillis();
        String s = "";
        for (int i = 0; i < 65536; i++) {
            s += r.nextInt(2);
        }
        System.out.println(System.currentTimeMillis() - start);

        r = new Random(123);
        start = System.currentTimeMillis();
        StringBuilder sb = new StringBuilder();
        for (int i = 0; i < 65536; i++) {
            sb.append(r.nextInt(2));
        }
        s = sb.toString();
        System.out.println(System.currentTimeMillis() - start);
    }
}
```

Figure 33: Class ConcatenationInLoops

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

import java.io.*;
import java.nio.file.*;

public class GarbageCreator {
    public static void main(String[] args) {
        String filename = "test.exe";

        try {
            if (!Files.exists(Paths.get(filename))) {
                createLargeFile(filename);
            }

            System.out.println("Starting to read file with String concatenation...");
            long startTime = System.currentTimeMillis();

            try (FileReader fr = new FileReader(filename)) {
                String content = "";
                int character;
                int charCount = 0;

                while ((character = fr.read()) != -1) {
                    content += (char) character;
                    charCount++;

                    if (charCount % 1000000 == 0) {
                        System.out.println("Read " + charCount + " characters...");
                    }
                }
```

```
31
32                  System.out.println("Final string length: " + content.length());
33              }
34
35              long endTime = System.currentTimeMillis();
36              System.out.println("Time taken: " + (endTime - startTime) + " ms");
37
38          } catch (OutOfMemoryError e) {
39              System.out.println("Out of Memory Error occurred!");
40              System.out.println("Error: " + e.getMessage());
41              e.printStackTrace();
42          } catch (IOException e) {
43              System.out.println("IO Error occurred!");
44              System.out.println("Error: " + e.getMessage());
45              e.printStackTrace();
46          }
47      }
48
49      private static void createLargeFile(String filePath) throws IOException {
50          try (FileWriter writer = new FileWriter(filePath)) {
51              for (int i = 0; i < 100000000; i++) {
52                  writer.write("This is a test line to create a large file.\n");
53              }
54          }
55      }
56  }
```

Figure 34: Class GarbageCreator

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

import java.io.*;
import java.nio.file.*;

public class NoGarbage {
    public static void main(String[] args) {
        String filename = "test.exe";

        try {
            if (!Files.exists(Paths.get(filename))) {
                createLargeFile(filename);
            }

            System.out.println("Starting to read file with StringBuffer...");
            long startTime = System.currentTimeMillis();

            try (FileReader fr = new FileReader(filename)) {
                StringBuilder content = new StringBuilder();
                int character;
                int charCount = 0;

                while ((character = fr.read()) != -1) {
                    content.append((char) character);
                    charCount++;

                    if (charCount % 1000000 == 0) {
                        System.out.println("Read " + charCount + " characters...");
                    }
                }
```

```
31
32                  System.out.println("Final string length: " + content.length());
33              }
34
35              long endTime = System.currentTimeMillis();
36              System.out.println("Time taken: " + (endTime - startTime) + " ms");
37
38          } catch (OutOfMemoryError e) {
39              System.out.println("Out of Memory Error occurred!");
40              System.out.println("Error: " + e.getMessage());
41              e.printStackTrace();
42          } catch (IOException e) {
43              System.out.println("IO Error occurred!");
44              System.out.println("Error: " + e.getMessage());
45              e.printStackTrace();
46          }
47      }
48
49      private static void createLargeFile(String filePath) throws IOException {
50          try (FileWriter writer = new FileWriter(filePath)) {
51              for (int i = 0; i < 100000000; i++) {
52                  writer.write("This is a test line to create a large file.\n");
53              }
54          }
55      }
56  }
```

Figure 35: Class NoGarbage

9. UseCase Diagram

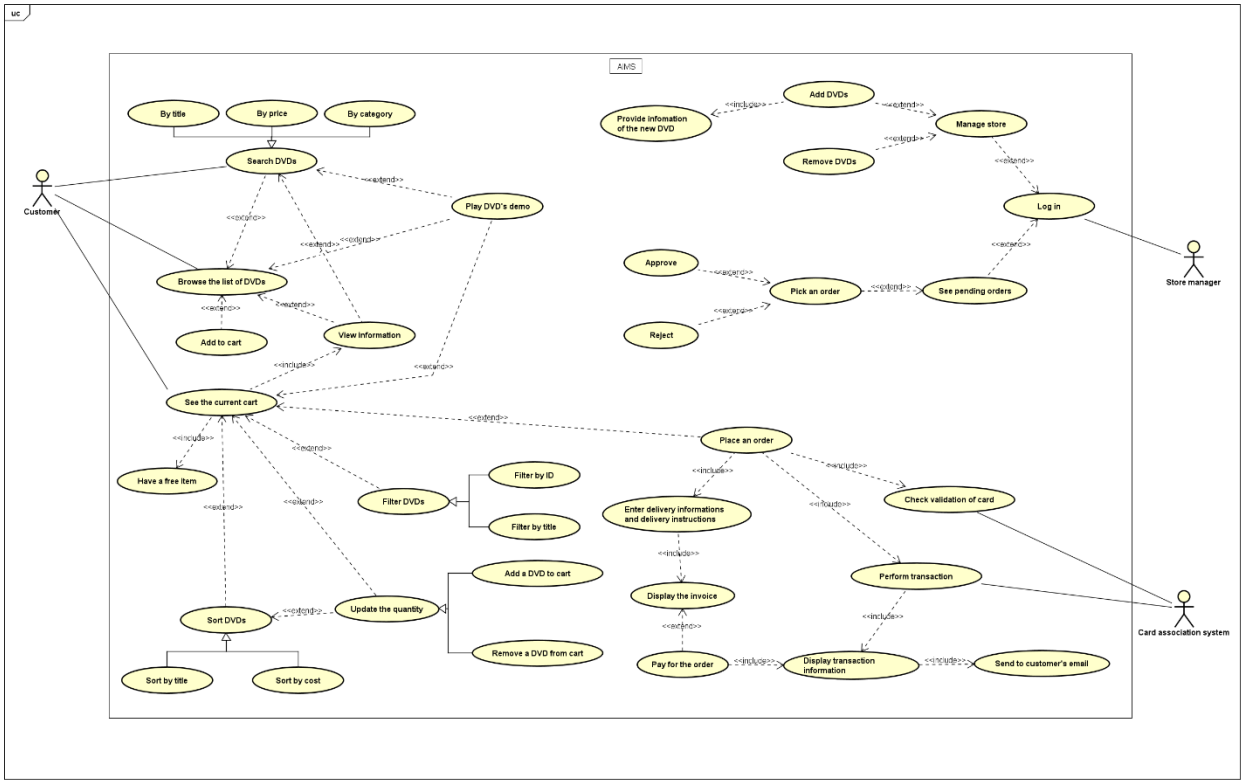

Figure 36: UseCase Diagram

10. Class Diagram

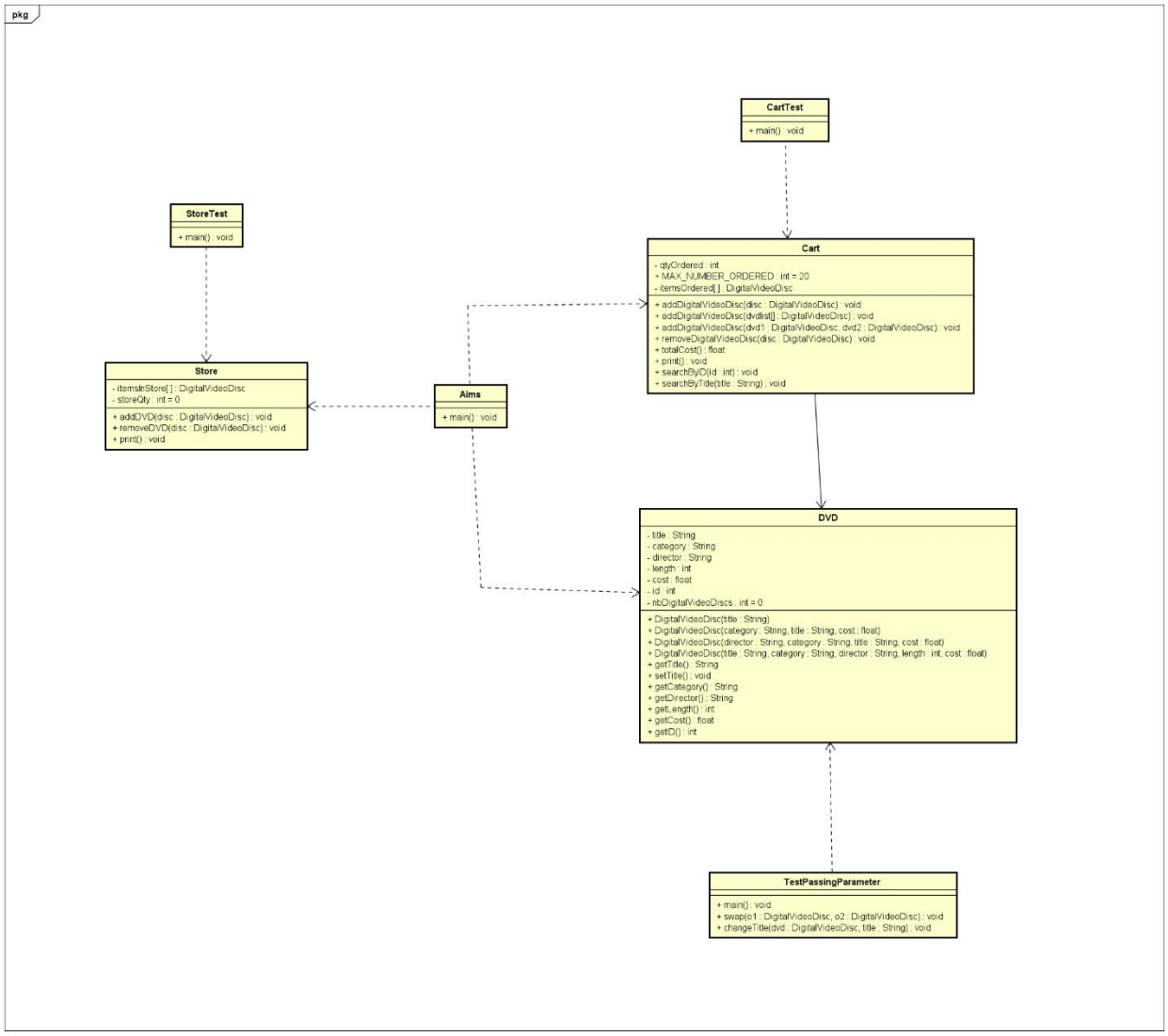

Figure 37: Class Diagram