ĐẠI HỌC BÁCH KHOA HÀ NỘI
TRƯỜNG CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG

BÁO CÁO THỰC HÀNH
IT3103-744527-2024.1

BÀI THỰC HÀNH – LAB 3

Họ và tên SV: Đặng Hồng Minh
MSSV: 20225740
Lớp: CNTT Việt Nhật – K67
GVHD: Lê Thị Hoa
HTGD: Đặng Mạnh Cường

Hà Nội 11/2024

Mục lục

Danh mục hình ảnh

BÁO CÁO THỰC HÀNH LAB 3

1. Working with method overloading

1.1. Overloading by differing types of parameter

Code:

- The current method has one input parameter of class DigitalVideoDisc

```
10      //Original version
11      public void addDigitalVideoDisc(DigitalVideoDisc dvd) {
12          //If cart not full
13          if (qtyOrdered < MAX_ITEMS) {
14              itemOrdered[qtyOrdered] = dvd;
15              qtyOrdered++;
16              System.out.println(dvd.getTitle()+" has been added!");
17              if (MAX_ITEMS - qtyOrdered == 1)
18                  System.out.println(x:"Warning! The cart is almost full.");
19          } else {
20              System.out.println(x:"The cart is full! Cannot add more DVDs.");
21          }
22      }
```

Figure 1: Method addDigitalVideoDisc(DigitalVideoDisc dvd) – original

- A new method that has the same name but with array type parameter

```
23      //Overloading version
24      //Array method
25      public void addDigitalVideoDisc(DigitalVideoDisc[] dvdList) {
26          for (DigitalVideoDisc dvd : dvdList) {
27              if (qtyOrdered < MAX_ITEMS) {
28                  itemOrdered[qtyOrdered++] = dvd;
29                  qtyOrdered++;
30                  System.out.println(dvd.getTitle() + " has been added!");
31              } else {
32                  System.out.println(x:"The cart is full! Cannot add more DVDs.");
33                  break;
34              }
35          }
36          if (MAX_ITEMS - qtyOrdered == 1) {
37              System.out.println(x:"Warning! The cart is almost full.");
38          }
39      }
```

Figure 2: Method addDigitalVideoDisc(DigitalVideoDisc[] dvdList) – array type parameter

- A new method which allows to pass an arbitrary number of DVDs

```java
    //Variable-length argument lists
    public void addDigitalVideoDisc(DigitalVideoDisc... dvds) {
        for (DigitalVideoDisc dvd : dvds) {
            if (qtyOrdered < MAX_ITEMS) {
                itemOrdered[qtyOrdered++] = dvd;
                qtyOrdered++;
                System.out.println(dvd.getTitle() + " has been added!");
            } else {
                System.out.println(x:"The cart is full! Cannot add more DVDs.");
                break;
            }
        }
        if (MAX_ITEMS - qtyOrdered == 1) {
            System.out.println(x:"Warning! The cart is almost full.");
        }
    }
```

Figure 3: Method addDigitalVideoDisc(DigitalVideoDisc... dvds) – variable-length argument

Result:

```
AIMS Store Project:
---by Dang Hong Minh - 20225740---
Green Miles has been added!
Kimi no Na wa has been added!
Spider Man has been added!

Current DVDs in Cart:
Title: Green Miles | Cost: $18.99
Title: Kimi no Na wa | Cost: $12.59
Title: Spider Man | Cost: $15.29
Total = 46.87
```

Figure 4: Result 1

```
AIMS Store Project:
---by Dang Hong Minh - 20225740---
The Lion King has been added!
Star Wars has been added!
Aladin has been added!

Current DVDs in Cart:
Title: The Lion King | Cost: $19.95
Title: Star Wars | Cost: $24.95
Title: Aladin | Cost: $18.99
Total = 63.89
```

Figure 5: Result 2

- Implementations of new methods: These methods will add a list of DVDs to the current cart. Please note that we cannot compile both because they have the same name and signature (one list as a parameter).
- Which method do you prefer in this case?

=> Since we can compile only one method, we should use varargs parameter method. Array method is more readable and easier to understand. But varargs method is more convenient for testing purpose.

1.2. Overloading by differing the number of parameters

Code:

```java
    public void addDigitalVideoDisc(DigitalVideoDisc dvd1,DigitalVideoDisc dvd2) {
        if (qtyOrdered + 2 <= MAX_ITEMS){
            itemOrdered[qtyOrdered] = dvd1;
            qtyOrdered++;
            itemOrdered[qtyOrdered] = dvd2;
            System.out.println(dvd1.getTitle() + " and " + dvd2.getTitle() + " has been added!");
            qtyOrdered++;
        } else {
            System.out.println(x:"The cart is full! Cannot add more DVDs.");
        }
        if (MAX_ITEMS - qtyOrdered == 1) {
            System.out.println(x:"Warning! The cart is almost full.");
        }
    }
```

Figure 6: Method addDigitalVideoDisc(DigitalVideoDisc dvd1, DigitalVideoDisc dvd2)

Result:

```
AIMS Store Project:
---by Dang Hong Minh - 20225740---
The Lion King has been added!
Star Wars has been added!
Aladin has been added!
The Lion King and Star Wars has been added!

Current DVDs in Cart:
Title: The Lion King | Cost: $19.95
Title: Star Wars | Cost: $24.95
Title: Aladin | Cost: $18.99
Title: The Lion King | Cost: $19.95
Title: Star Wars | Cost: $24.95
Total = 108.79
```

Figure 7: Result 3

2. Passing parameter

Code:

```java
//Dang Hong Minh - 20225740
package lab3.AimsProject;
import lab2.DigitalVideoDisc;
public class TestPassingParameter {
    Run | Debug
    public static void main (String[] args) {
        //TO DO Auto-generated method stub
        DigitalVideoDisc jungleDVD = new DigitalVideoDisc(title:"Jungle");
        DigitalVideoDisc cinderellaDVD = new DigitalVideoDisc(title:"Cinderella");
        swap(jungleDVD, cinderellaDVD);
        System.out.println("Jungle DVD's title: " + jungleDVD.getTitle());
        System.out.println("Cinderella DVD's title: " + cinderellaDVD.getTitle());
        changeTitle(jungleDVD, cinderellaDVD.getTitle());
        System.out.println("Jungle DVD's 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 8: Passing parameter code

Result:

```
Jungle DVD's title: Jungle
Cinderella DVD's title: Cinderella
Jungle DVD's title: Cinderella
```

Figure 9: Result 4

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

=> After the call of swap(), the title of these two objects still remains because the swap() method only swaps the references to the objects, not the objects themselves. This means that after the call to swap(), o1 and o2 still refer to the same objects as they did before the call.

- After the call of changeTitle(), why is the title of the JungleDVD changed?

=> After the call of changeTitle(), the title of the DVD is changed because the setTitle(title) method changes the title by value directly. The dvd = new DigitalVideoDisc(oldTitle) creates a new DVD object with the old title. It does not affect the original dvd object as the reference variable no longer points to the original object.

- Is Java a Pass-by-value or Pass-by-reference?

=> In conclusion, Java is always a pass-by-value programming language.

- A swap() method that can correctly swap the two objects

```
19      }*/
20      //A swap() method that can correctly swap the two objects.
21      //Hint: Swap attributes of two objects
22      public static void swap(DigitalVideoDisc o1, DigitalVideoDisc o2){
23          String tmp = o1.getTitle();
24          o1.setTitle(o2.getTitle());
25          o2.setTitle(tmp);
26      }
```

Figure 10: A swap() method that can correctly swap the two objects

```
Jungle DVD's title: Cinderella
Cinderella DVD's title: Jungle
Jungle DVD's title: Jungle
```

Figure 11: Result 5

3. Class Member and Instance Member

Code:

```java
//Dang Hong Minh - 20225740
package lab3.AimsProject;
public class DigitalVideoDisc {
    //Attribute
    private String title;
    private String category;
    private String director;
    private int length;
    private float cost;
    private static int nbDigitalVideoDiscs = 0;
    private int id;
    //Construct by title
    public DigitalVideoDisc(String title) {
        super();
        this.title = title;
        this.id = ++nbDigitalVideoDiscs;
    }
    //Construct by title, cost
    public DigitalVideoDisc(String title, float cost) {
        super();
        this.title = title;
        this.cost = cost;
        this.id = ++nbDigitalVideoDiscs;
    }
    //Construct by title, category, cost
    public DigitalVideoDisc(String title, String category, float cost) {
        super();
        this.title = title;
        this.category = category;
        this.cost = cost;
        this.id = ++nbDigitalVideoDiscs;
    }
    //Construct by title, category, director, cost
    public DigitalVideoDisc(String title, String category, String director, float cost){
        super();
        this.title = title;
        this.category = category;
        this.director = director;
        this.cost = cost;
        this.id = ++nbDigitalVideoDiscs;
    }
    //Construct by title, category, director, length, cost
    public DigitalVideoDisc(String title, String category, String director, int length, float cost) {
        super();
        this.title = title;
        this.category = category;
        this.cost = cost;
        this.director = director;
        this.length = length;
        this.id = ++nbDigitalVideoDiscs;
    }
    public String getTitle() {
        return title;
    }
    public String getCategory() {
        return category;
    }
    public float getCost() {
        return cost;
    }
    public String getDirector() {
        return director;
    }
    public int getLength() {
        return length;
    }
    public int getId(){
        return id;
    }
    public static int getNbDigitalVideoDiscs() {
        return nbDigitalVideoDiscs;
    }
}
```

Figure 12: Classifier Member and Instance Member Code

```java
//Dang Hong Minh - 20225740
package lab3.AimsProject;
public class Aims {
    Run | Debug
    public static void main(String[] args) {
        System.out.println(x:"\nAIMS Store Project:");
        System.out.println(x:"---by Dang Hong Minh - 20225740---");
        //Create new cart
        Cart cart = new Cart();
        //Create new DVD object and add to cart
        DigitalVideoDisc dvd1 = new DigitalVideoDisc
        (title:"The Lion King", category:"Animation", director:"Roger Allers", length:87, cost:19.95f);
        cart.addDigitalVideoDisc(dvd1);
        DigitalVideoDisc dvd2 = new DigitalVideoDisc
        (title:"Star Wars", category:"Sci-Fi", director:"George Lucas", cost:24.95f);
        cart.addDigitalVideoDisc(dvd2);
        DigitalVideoDisc dvd3 = new DigitalVideoDisc
        (title:"Aladin", cost:18.99f);
        cart.addDigitalVideoDisc(dvd3);
        DigitalVideoDisc dvd4 = new DigitalVideoDisc
        (title:"Avatar", category:"Sci-Fi", cost:32.99f);
        cart.addDigitalVideoDisc(dvd4);
        DigitalVideoDisc dvd5 = new DigitalVideoDisc
        (title:"Fate/Stay Night", category:"Animation", cost:12.99f);
        cart.addDigitalVideoDisc(dvd5);
        //Display all DVDs in the cart
        cart.displayCart();
        //Print number of DVD in cart
        System.out.println(x:"Number of DVD is: ");
        System.out.println(DigitalVideoDisc.getNbDigitalVideoDiscs());
        //Print ID of dvd3
        System.out.println(x:"ID of dvd3 is: ");
        System.out.println(dvd3.getId());
    }
}
```

Figure 13: Classifier Member and Instance Member Code – Aims class

Result:

```
AIMS Store Project:
---by Dang Hong Minh - 20225740---
The Lion King has been added!
Star Wars has been added!
Aladin has been added!
Avatar has been added!
Fate/Stay Night has been added!

Current DVDs in Cart:
Title: The Lion King | Cost: $19.95
Title: Star Wars | Cost: $24.95
Title: Aladin | Cost: $18.99
Title: Avatar | Cost: $32.99
Title: Fate/Stay Night | Cost: $12.99
Number of DVD is:
5
ID of dvd3 is:
3
```

Figure 14: Result 6

4. Open the Cart class

Code:

```
73      public String toString(){
74          String result = "";
75          for (int i = 0; i < qtyOrdered; i++) {
76              result += "\n" + itemOrdered[i].toString();
77          }
78          return result;
79          }
80      //Show cart info
81      public void printCart(){
82          System.out.printf(format:"\n**********CART**********\n");
83          System.out.printf(format:"Ordered Items:");
84          System.out.println(toString());
85          System.out.printf(format:"Total cost: $%.2f\n", totalCost());
86          System.out.printf(format:"**********\n");
87      }
```

Figure 15: Code method to print the content in Cart

```
88      //Search in cart
89      public void searchCart(int id){ //Search by ID
90          int check = 0;
91          for(int i = 0; i < qtyOrdered; i++) {
92              if(itemOrdered[i].isMatch(id)) {
93                  check = 1;
94                  System.out.printf(format:"Found item with ID %d!\n", id);
95                  System.out.println(itemOrdered[i].toString());
96                  break;
97              }
98          }
99          if (check == 0) System.out.println(x:"Not Found!\n");
100     }
101     public void searchCart(String title){ //Search by title
102         int check = 0;
103         for (int i = 0; i < qtyOrdered; i++) {
104             if (itemOrdered[i].isMatch(title)) {
105                 check = 1;
106                 System.out.printf(format:"Found item with title included %s!\n", title);
107                 System.out.println(itemOrdered[i].toString());
108                 break;
109             }
110         }
111         if (check == 0) System.out.println(x:"Not Found!\n");
112     }
113 }
```

Figure 16: Code method search by id and title

```
77      public boolean isMatch(int id) {
78          return this.id == id;
79      }
80      public boolean isMatch(String title) {
81          String[] tmp = title.split(regex:" ", limit:0);
82          for (String x : tmp) {
83              if (getTitle().toLowerCase().contains(x.toLowerCase())) return true;
84          }
85          return false;
86      }
87  }
```

Figure 17: Method isMatch in class DigitalVideoDisc to check title and ID of a DVD

Result:

```java
//Dang Hong Minh - 20225740
package lab3.AimsProject;
public class CartTest {
    Run | Debug
    public static void main(String[] args){
    Cart cart = new Cart();
    DigitalVideoDisc [] dvdList = new DigitalVideoDisc[3];
    dvdList[0] = new DigitalVideoDisc(title:"Green Miles", category:"Drama", director:"Frank Darabont", length:279, cost:18.99f);
    dvdList[1] = new DigitalVideoDisc(title:"Kimi no Na wa", category:"Romance", director:"Shinkai Makoto", length:110, cost:12.59f);
    dvdList[2] = new DigitalVideoDisc(title:"Spider Man", category:"Superhero", director:"Sam Raimi", length:200, cost:15.29f);
    cart.addDigitalVideoDisc(dvdList);
    // print items in cart and total cost
    cart.printCart();
    cart.searchCart(title:"Kimi no Na wa");
    cart.searchCart(id:1);
    }
}
```

Figure 18: CartTest class

```
Green Miles has been added!
Kimi no Na wa has been added!
Spider Man has been added!

**********CART**********
Ordered Items:
1. Green Miles - Drama - Frank Darabont - 279: 18.99$
2. Kimi no Na wa - Romance - Shinkai Makoto - 110: 12.59$
3. Spider Man - Superhero - Sam Raimi - 200: 15.29$
Total cost: $46.87
**************************************************
Found item with title included Kimi no Na wa!
2. Kimi no Na wa - Romance - Shinkai Makoto - 110: 12.59$
Found item with ID 1!
1. Green Miles - Drama - Frank Darabont - 279: 18.99$
```

Figure 19: Result 7

5. Implement the Store class

Code:

```java
//Dang Hong Minh - 20225740
package lab3.AimsProject;
public class Store {
    private DigitalVideoDisc itemInStore[];
    private int numberOfItems;
    public Store(int maxSize) {
        itemInStore = new DigitalVideoDisc[maxSize];
        numberOfItems = 0;
    }
    //Add DVDs
    public void addDVD(DigitalVideoDisc dvd) {
        if (numberOfItems < itemInStore.length) {
            itemInStore[numberOfItems] = dvd;
            numberOfItems++;
            System.out.println(dvd.getTitle() + " has been added!");
        } else {
            System.out.println(x:"Cannot add more DVD!");
        }
    }
    //Remove DVDs
    public void removeDVD(DigitalVideoDisc dvd) {
        boolean check = false; //Check if DVD already exists
        for (int i = 0; i < numberOfItems; i++) {
            if (itemInStore[i].equals(dvd)) {
                check = true;
                for (int j = i; j < numberOfItems - 1; j++) {
                    itemInStore[j] = itemInStore[j + 1];
                }
                numberOfItems--;
                System.out.println(dvd.getTitle() + " has been removed!");
                break;
            }
        }
        if (!check) System.out.println(dvd.getTitle() + " not found!");
    }
    //Display store interface
    public void displayStore() {
        System.out.println(x:"**********DVD STORE**********");
        for (int i = 0; i < numberOfItems; i++) {
            System.out.println(itemInStore[i].toString());
        }
        System.out.println(x:"*****************************");
    }
}
```

Figure 20: Store Class

```java
//Dang Hong Minh - 20225740
package lab3.AimsProject;
public class StoreTest {
    Run | Debug
    public static void main(String[] args) {
        //Create new store with a 5 DVDs max and add DVD to the store
        Store store = new Store(maxSize:5);
        //Create some DVDs to add to the store
        DigitalVideoDisc dvd1 = new DigitalVideoDisc(title:"Tabi no Tochuu",
        category:"Music", director:"Natsumi Kiyoura", length:5, cost:17.44f);
        store.addDVD(dvd1);
        DigitalVideoDisc dvd2 = new DigitalVideoDisc(title:"See Tinh",
        category:"Music", director:"Hoang Thuy Linh", length:6, cost:16.99f);
        store.addDVD(dvd2);
        DigitalVideoDisc dvd3 = new DigitalVideoDisc(title:"Shake it Off",
        category:"Music", director:"Taylor Swift", length:4, cost:20.23f);
        store.addDVD(dvd3);
        //Display the current state of the store
        store.displayStore();
        //Remove a DVD from the store (success)
        store.removeDVD(dvd2);
        // Remove failed
        DigitalVideoDisc dvd4 = new DigitalVideoDisc(title:"Harry Potter",
        category:"Fiction", cost:18.99f);
        store.removeDVD(dvd4);
        //Display the updated store
        store.displayStore();
    }
}
```

Figure 21: StoreTest class

Result:

```
Tabi no Tochuu has been added!
See Tinh has been added!
Shake it Off has been added!
**********DVD STORE**********
1. Tabi no Tochuu - Music - Natsumi Kiyoura - 5: 17.44$
2. See Tinh - Music - Hoang Thuy Linh - 6: 16.99$
3. Shake it Off - Music - Taylor Swift - 4: 20.23$
**************************
See Tinh has been removed!
Harry Potter not found!
**********DVD STORE**********
1. Tabi no Tochuu - Music - Natsumi Kiyoura - 5: 17.44$
3. Shake it Off - Music - Taylor Swift - 4: 20.23$
**************************
```

Figure 22: Result 8

6. String, StringBuilder and StringBuffer

Code:

```java
//Dang Hong Minh - 20225740
import java.util.Random;
public class ConcatenationInLoops {
    Run | Debug
    public static void main(String[] args) {
        Random r = new Random(seed:123);
        long start = System.currentTimeMillis();
        String s = "";
        for(int i = 0;i < 65536;i++)
            s += r.nextInt(bound:2);
        System.out.println(System.currentTimeMillis() - start);
        r = new Random(seed:123);
        start = System.currentTimeMillis();
        StringBuilder sb = new StringBuilder();
        for(int i = 0;i < 65536;i++)
            sb.append(r.nextInt(bound:2));
        s = sb.toString();
        System.out.println(System.currentTimeMillis()- start);
        System.out.println("Length of string s: " + s.length());
    }
}
```

Figure 23: ConcatenationInLoops class

```java
//Dang Hong Minh - 20225740
import java.nio.file.Files;
import java.nio.file.Paths;
public class GarbageCreator {
    Run | Debug
    public static void main(String[] args) {
        String filename = "sample.txt"; // ~ 80KB
        byte[] inputBytes = { 0 };
        long startTime, endTime;
        try {
            inputBytes = Files.readAllBytes(Paths.get(filename));
        } catch (Exception e) {
            e.printStackTrace();
        }
        startTime = System.currentTimeMillis();
        String outputString = "";
        for (byte b : inputBytes) {
            outputString += (char) b;
        }
        endTime = System.currentTimeMillis();
        System.out.println(endTime - startTime);
        System.out.println("Output string length: " + outputString.length());
    }
}
```

Figure 24: GarbageCreator class

```java
//Dang Hong Minh - 20225740
import java.nio.file.Files;
import java.nio.file.Paths;
public class NoGarbage {
    Run | Debug
    public static void main(String[] args) {
        String filename = "sample.txt";
        byte[] inputBytes = { 0 };
        long startTime, endTime;
        try {
            inputBytes = Files.readAllBytes(Paths.get(filename));
        } catch (Exception e) {
            e.printStackTrace();
        }
        startTime = System.currentTimeMillis();
        StringBuffer outputStringBuilder = new StringBuffer();
        for (byte b : inputBytes) {
            outputStringBuilder.append(Character.toString((char) b));
        }
        endTime = System.currentTimeMillis();
        System.out.println(endTime - startTime);
    }
}
```

Figure 25: NoGarbage class

Result:

```
40.DangHongMinh\lab3\OtherProjects> java ConcatenationInLoops
473
2
```

Figure 26: So sánh String và StringBuilder

```
.DangHongMinh\lab3\OtherProjects> java GarbageCreator
1149
Output string length: 102400
```

Figure 27: Result: program hangs

```
40.DangHongMinh\lab3\OtherProjects> java NoGarbage
6
```

Figure 28: Result: Program finishes quickly

- When concatenates, Java creates a new string that contains the combined contents of the two original strings. This process involves allocating memory for the new string, copying the characters from the original strings, and updating the reference to the new string.

- In contrast, appending to a StringBuilder does not create a new string . StringBuilder maintains a buffer that stores the characters of the string. When appends, the new characters are simply added to the end of the buffer.

- The first loop using string concatenation takes significantly longer time than the second loop using StringBuilder. This is because the first loop creates a new string for each iteration of the loop, whereas the second loop only appends to a single StringBuilder.

7. Updated Class Diagram

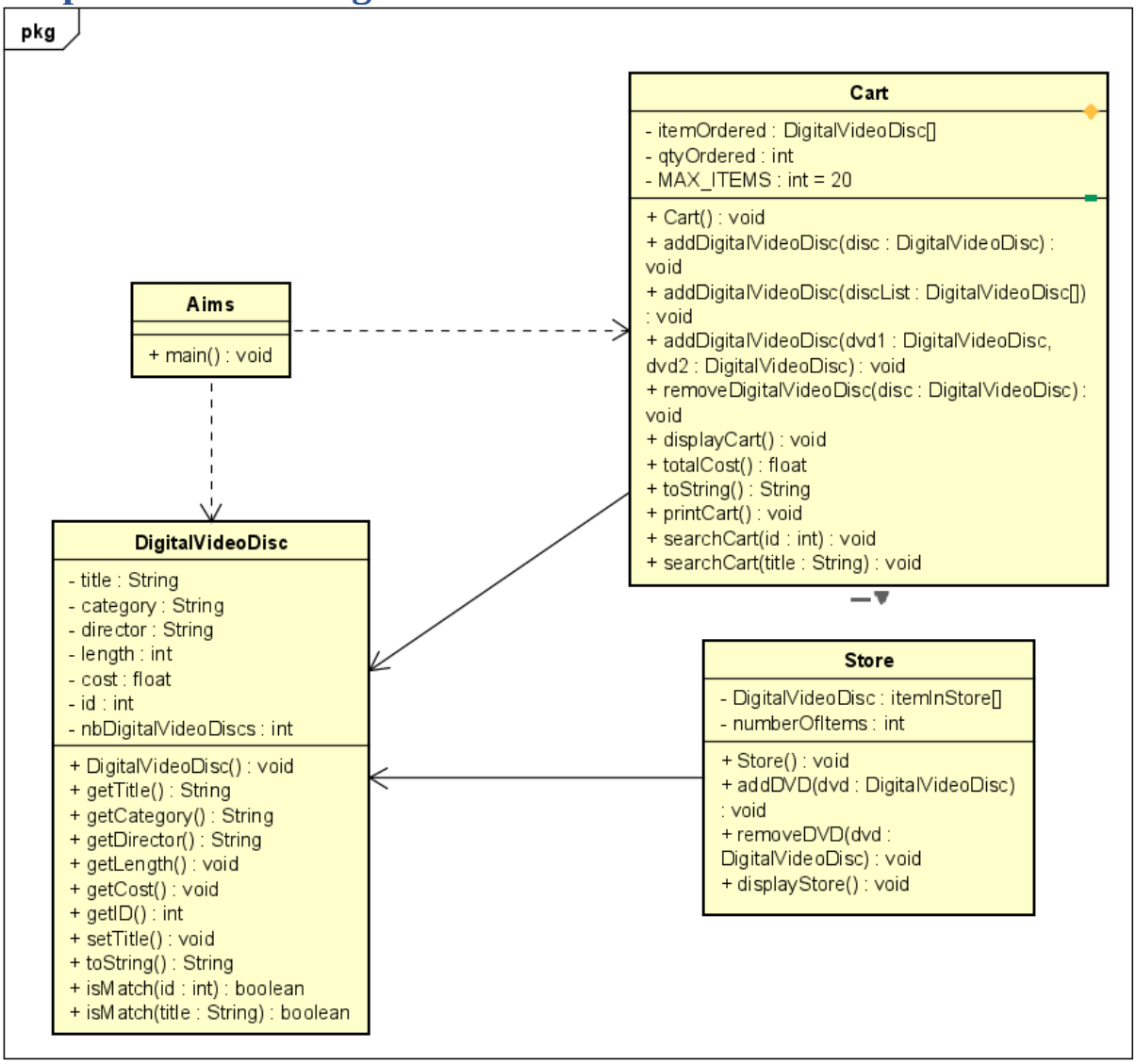

Figure 29: Updated Class Diagram