**BÁO CÁO THỰC HÀNH TUẦN 3**

**Lập trình hướng đối tượng**

Dương Văn Nhất – 20215106

Table of Contents

[1. Branch your repository 1](#_Toc151136692)

[2. Working with method overloading 1](#_Toc151136693)

[3. Passing parameter 1](#_Toc151136694)

[4. Use debug run 1](#_Toc151136695)

[5. Classifier Member and Instance Member 1](#_Toc151136696)

[6. Open the Cart class 1](#_Toc151136697)

[7. Implement the Store class 1](#_Toc151136698)

[8. Re-organize your projects 1](#_Toc151136699)

[9. String, StringBuilder and StringBuffer 1](#_Toc151136700)

[10. Release flow demonstration 1](#_Toc151136701)

# Branch your repository

# Working with method overloading

Method overloading allows different methods to have the same name but different signatures where signature can differ by number of input parameters or type of input parameter(s) or both.

2.1 Overloading by differing types of parameter

- Open Eclipse

- Open the JavaProject named "AimsProject" that you have created in the previous lab.

- Open the class Cart.java: you will overload the method addDigitalVideoDisc you created last time.

+ The current method has one input parameter of class DigitalVideoDisc

+ You will create a new method that has the same name but with different type of parameter.

addDigitalVideoDisc(DigitalVideoDisc [] dvdList)

This method will add a list of DVDs to the current cart.

+ 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?

Code:

//Truyền tham số theo mảng biến số (variable arguments array)

    public void addDigitalVideoDisc(DigitalVideoDisc... dvdList){

        for (int i = 0 ; i < dvdList.length ; i++){

            this.addDigitalVideoDisc(dvdList[i]);

        }

    }

    // Truyền tham số là mảng

    public void addDigitalVideoDisc(DigitalVideoDisc[] list){

        for (int i = 0 ; i < list.length ; i++){

            this.addDigitalVideoDisc(list[i]);

        }

    }

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)

code:

// Truyền tham số khác với các hàm trên

    public void addDigitalVideoDisc(DigitalVideoDisc dvd1,DigitalVideoDisc dvd2){

        this.addDigitalVideoDisc(dvd1);

        this.addDigitalVideoDisc(dvd2);

    }

A screenshot of a computer program

Description automatically generated

# Passing parameter

\*JAVA là ngôn ngữ lập trình Pass by Value hay Pass by Reference?

Java là ngôn ngữ lập trình hướng đối tượng dựa trên kiểu truyền tham số theo giá trị (pass-by-value).

Cụ thể:

Trong Java, khi truyền tham số cho phương thức hoặc hàm, giá trị tham chiếu của biến được sao chép vào bộ nhớ của phương thức. Nó không truyền trực tiếp địa chỉ bộ nhớ của biến.

Nếu tham số là kiểu nguyên thủy (primitive type) thì giá trị của nó sẽ được sao chép. Nếu là kiểu tham chiếu (reference type) thì giá trị tham chiếu sẽ được sao chép chứ không phải đối tượng được truyền.

Trong phương thức, nếu thay đổi giá trị của biến tham số thì không ảnh hưởng đến biến gọi hàm bên ngoài.

Do đó, ta kết luận Java là ngôn ngữ lập trình hỗ trợ kiểu truyền tham số theo giá trị (pass-by-value).

Sau khi gọi phương thức swap(jungleDVD, cinderellaDVD), tại sao tiêu đề của hai đối tượng này vẫn còn không đổi?

-Lý do là trong Java, các tham số luôn được truyền giá trị (pass by value). Nghĩa là giá trị của đối tượng được sao chép rồi mới truyền vào phương thức. Trong phương thức swap, việc hoán đổi chỉ diễn ra với các biến sao chép chứ không phải đối tượng ban đầu. Do đó tiêu đề của hai đối tượng vẫn giữ nguyên.

Sau khi gọi phương thức changeTitle(jungleDVD, cinderellaDVD.getTitle()), tại sao tiêu đề của JungleDVD lại thay đổi?

-Lý do là trong phương thức này, tiêu đề của JungleDVD được gán lại bằng giá trị trả về từ cinderellaDVD.getTitle(). Do đó tiêu đề của nó mới thay đổi.

phương thức swap() có thể hoán đổi chính xác hai đối tượng:

public static void SetValue(DigitalVideoDisc dvd1, DigitalVideoDisc dvd2) {

        dvd1.setTitle(dvd2.getTitle());

        dvd1.setCategory(dvd2.getCategory());

        dvd1.setDirector(dvd2.getDirector());

        dvd1.setCost(dvd2.getCost());

        dvd1.setLength(dvd2.getLength());

    }

    public static void swap2(DigitalVideoDisc dvd1, DigitalVideoDisc dvd2) {

        DigitalVideoDisc tmp = new DigitalVideoDisc();

        SetValue(tmp, dvd1);

        SetValue(dvd1, dvd2);

        SetValue(dvd2, tmp);

    }

Full 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: " + jungleDVD.getTitle()));

        System.out.println("---------------" );

        swap2(jungleDVD, cinderellaDVD);

        System.out.println("jungle dvd title: " + jungleDVD.getTitle());

        System.out.println("cinderella dvd title: " + cinderellaDVD.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);

    }

    // viết hàm hoán đổi 2 đối tượng

    public static void SetValue(DigitalVideoDisc dvd1, DigitalVideoDisc dvd2) {

        dvd1.setTitle(dvd2.getTitle());

        dvd1.setCategory(dvd2.getCategory());

        dvd1.setDirector(dvd2.getDirector());

        dvd1.setCost(dvd2.getCost());

        dvd1.setLength(dvd2.getLength());

    }

    public static void swap2(DigitalVideoDisc dvd1, DigitalVideoDisc dvd2) {

        DigitalVideoDisc tmp = new DigitalVideoDisc();

        SetValue(tmp, dvd1);

        SetValue(dvd1, dvd2);

        SetValue(dvd2, tmp);

    }

}

Kết quả:

A screenshot of a computer

Description automatically generated

# Use debug run

4.1. Debugging Java in Eclipse Video: https://www.youtube.com/watch?v=9gAjIQc4bPU&t=8s

Debugging is the routine process of locating and removing bugs, errors or abnormalities from programs. It’s a must have skill for any Java developer because it helps to find subtle bugs that are not visible during code reviews or that only happen when a specific condition occurs. The Eclipse Java IDE provides many debugging tools and views grouped in the Debug Perspective to help you as a developer debug effectively and efficiently.

Debug run allows you to run a program interactively while watching the source code and the variables during the execution. A breakpoint in the source code specifies where the execution of the program should stop during debugging. Once the program is stopped you can investigate variables, change their content, etc.

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

4.2.1. Setting, deleting & deactivate breakpoints:

To set a breakpoint, place the cursor on the line that needs debugging, hold down Ctrl+Shift, and press B to enable a breakpoint. A blue dot in front of the line will appear (Figure 5). Alternatively, you can right-click in the left margin of the line in the Java editor and select Toggle Breakpoint. This is equivalent to double-clicking in the left margin of the line.

A screenshot of a computer program

Description automatically generated

Figure 5. A breakpoint is set

To delete a breakpoint, toggle the breakpoint one more time. The blue dot in front of the line will disappear (Figure 6).

A screen shot of a computer program

Description automatically generated

Figure 6. The breakpoint is deleted

To deactivate the breakpoint, navigate to the Breakpoints View and uncheck the tick mark next to the breakpoint you want to deactivate (Figure 7). The program will only stop at activated breakpoints.

A screenshot of a computer

Description automatically generated

Figure 7. Deactivated breakpoint in Breakpoints View

For this example, we will need to keep this breakpoint, so make sure to set the breakpoint again after practicing with deleting/deactivating it before moving to the next section.

4.2.2. Run in Debug mode:

Select a Java file with a main method that contains the code that you need to debug from the Project Explorer. In this example, we choose the TestPassingParameter.java file. Right click and choose Debug As > Java Application (Figure 8).

A screenshot of a computer program

Description automatically generated

Figure 8. Run Debug from a class

Alternatively, you can select the project root node in the Project Explorer and click the debug icon in the Eclipse toolbar (Figure 9)

A screenshot of a computer

Description automatically generated

Figure 9. Run debug from a project

The application will now be started with Eclipse attached as debugger. Confirm to open the Debug Perspective.

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

- In the Debug Perspective, you can observe the Step Into/Over/Return & Resume/Terminate buttons on the toolbar as in Figure 10.

A screenshot of a music player

Description automatically generated

Figure 10. Stepping Commands on the Toolbar in Debug Perspective

- With debugger options, the difference between "Step into" and "Step over" is only noticeable if you run into a function call:

o "Step into" (F5) means that the debugger steps into the function

o "Step over" (F6) just moves the debugger to the next line in the same Java action

- With "Step Return" (pressing F7), you can instruct the debugger to leave the function; this is basically the opposite of "Step into."

- Clicking "Resume" (F8) instructs the debugger to continue until it reaches another breakpoint.

For this example, we need to see the execution of the swap function, so we choose Step Into. The debugger will step into the implementation of the swap function in line 18 (Figure 11).

A screenshot of a computer program

Description automatically generated

Figure 11. Step into swap function

4.2.4. Investigate value of variables:

We can observe the value of variables & expression in the Variables/Expression View. You can also add a permanent watch on an expression/variable that will then be shown in the Expressions view when debugging is on.

Alternatively, place your cursor on any of the variables in the Java action to see its value in a pop-up window. Open the Variable Perspective and observe the values of variables o1 & o2 (Figure 12). You can click the drop-down arrow to investigate attributes of variables.

A screenshot of a computer

Description automatically generated

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 13, Figure 14, Figure 15).

A screenshot of a computer

Description automatically generated

Figure 13. Step over line 18 of swap function

A screenshot of a computer program

Description automatically generated

Figure 14. Step over line 19 of swap function

A screenshot of a computer program

Description automatically generated

Figure 15. Step over line 20 of swap function

4.2.4. Change value of variables:

In the Variable Perspective, you can also change the value of variable while debugging. Click Step Return so the debugger returns from the swap function back to the line after the call to it. (Figure 16)

A screenshot of a computer program

Description automatically generated

Figure 16. Step return to main function

The variable jungleDVD still has a title attribute with value “Jungle”. You can change this value by clicking on it and change it to “abc”, for example (Figure 17).

A screenshot of a computer

Description automatically generated

Figure 17. Change title of jungleDVD Click Step Over and see the result in the output in the Console (Figure 18)

A screenshot of a computer

Description automatically generated

Figure 18. Results(2)

# Classifier Member and Instance Member

Classifier/Class member:

o Defined in a class of which a single copy exists regardless of how many instances of the class exist.

o Objective: to have variables that are common to all objects

o Any object of class can change the value of a class variable; that's why you should always be careful with the side effect of class member

o Class variables can be manipulated without creating an instance of the class

● Instance/Object member:

o Associated with only objects

o Defined inside the class but outside of any method

o Only initialized when the instance is created

o Their values are unique to each instance of a class

o Lives as long as the object does

Open the DigitalVideoDisc class:

● You should note that this class only has instance variables: title, category, director, length, cost.

- Now, we know that each DVD has a unique id assigned by the system. One simple way to manage all the ids is to give them out to new DVDs as consecutively incremented values. In order to do this, we must keep track of the number of DVDs created.

- Create a class attribute named "nbDigitalVideoDiscs" in the class DigitalVideoDisc

- Create an instance attribute named "id" in the class DigitalVideoDisc

- Each time an instance of the DigitalVideoDisc class is created, the nbDigitalVideoDiscs should be updated. Therefore, you should update the value for this class variable inside the constructor method and assign the appropriate value for the id.

Full code:

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 int getId (){

        return id;

    }

    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 DigitalVideoDisc(){

    }

    public DigitalVideoDisc(String title){

        this.id =Update();

        this.title = title;

    }

    public DigitalVideoDisc(String title , String category, float cost){

        this(title);

        this.category = category;

        this.cost = cost;

    }

    public DigitalVideoDisc(String title, String category, String director , float cost){

        this(title, category, cost);

        this.director = director;

    }

    public DigitalVideoDisc(String title, String category, String director , int length, float cost){

        this(title, category, director, cost);

        this.length = length;

    }

    public void setTitle(String title) {

        this.title = title;

    }

    public void setCategory(String category) {

        this.category = category;

    }

    public void setDirector(String director) {

        this.director = director;

    }

    public void setCost (float cost){

        this.cost = cost;

    }

    public void setLength (int length){

        this.length = length;

    }

    public boolean equals(DigitalVideoDisc disc1){

        if (disc1 == null || this == null)

        return false;

        if (!disc1.title.equals(this.title))

        return false;

        return true;

    }

    private int Update(){

        nbDigitalVideoDiscs ++;

        return nbDigitalVideoDiscs;

    }

}

# Open the Cart class

Write new methods to implement the following functions:

- Create a new method to print the list of ordered items of a cart, the price of each item, and the total price. Format the outline as below:

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*CART\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Ordered Items:

1. DVD - [Title] - [category] - [Director] - [Length]: [Price] $

2. DVD - [Title] - ...

Total cost: [total cost]

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

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

- Search for DVDs in the cart by ID and display the search results. Make sure to notify the user if no match is found.

- Search for DVDs in the cart by title and print the results. Make sure to notify the user if no match is found. Refer to problem statement in Lab02 for the matching rule. Suggestion: write a boolean isMatch(String title) method in the DigitalVideoDisc which finds out if the corresponding disk is a match given the title.

- In the CartTest class, write codes to test all methods you have written in this exercise. You should create sample DVDs and carts, like in this code snippet:

A computer screen shot of a program

Description automatically generated

    // code

    public String toString() {

        float sum = 0;

        String result = "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*CART\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n"

                + "Ordered Items:\n";

        for (int i = 0; i < qtyOrdered; i++) {

            DigitalVideoDisc dvd = itemOrdered[i];

            result += (i + 1) + ". DVD - " + dvd.getTitle() + " - "

                    + dvd.getCategory() + " - " + dvd.getDirector() + " - "

                    + dvd.getLength() + ": " + dvd.getCost() + "$\n";

            sum += itemOrdered[i].getCost();

        }

        result += "Total cost: " + sum + "\n";

        result += "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*";

        return result;

    }

    public DigitalVideoDisc findById(int id) {

        for (DigitalVideoDisc dvd : itemOrdered) {

            if (dvd.getId() == id) {

                return dvd;

            }

        }

        System.out.println("khong tim thay " + id);

        return null;

    }

    public DigitalVideoDisc findByTitle(String title) {

        for (DigitalVideoDisc dvd : itemOrdered) {

            if (dvd.getTitle().equals(title)) {

                return dvd;

            }

        }

        System.out.println("khong tim thay " + title);

        return null;

    }

    public void printDVD(DigitalVideoDisc dvd) {

        System.out.println("DVD - " + dvd.getTitle() + " - "

                + dvd.getCategory() + " - " + dvd.getDirector() + " - "

                + dvd.getLength() + ": " + dvd.getCost() + "$\n");

    }

// Test Cart:

public class testCart {

  public static void main(String[] args) {

    Cart cart = new Cart();

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

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

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

    cart.addDigitalVideoDisc(dvd1);

    cart.addDigitalVideoDisc(dvd2);

    cart.addDigitalVideoDisc(dvd3);

    cart.totalCost();

    System.out.println(cart.toString());

    cart.printDVD(cart.findById(2));

    cart.printDVD(cart.findByTitle("The Lion King"));

  }

}

Kết quả:

A screenshot of a computer

Description automatically generated

# Implement the Store class

Create a Store class, which contains one attribute itemsInStore[] – an array of DVDs available in the store.

- To add and remove DVDs from the store, implement two methods called addDVD and removeDVD

- Test these two methods in StoreTest class.

public class Store {

    private DigitalVideoDisc[] itemsInStore;

    public Store() {

        itemsInStore = new DigitalVideoDisc[100];

    }

    public void addDVD(DigitalVideoDisc dvd) {

        for (int i = 0; i < itemsInStore.length; i++) {

            if (itemsInStore[i] == null) {

                itemsInStore[i] = dvd;

                System.out.println("The disc with name " + dvd.getTitle() + " has been added");

                break;

            }

        }

    }

    public void removeDVD(DigitalVideoDisc dvd) {

        for (int i = 0; i < itemsInStore.length; i++) {

            if (itemsInStore[i] != null && itemsInStore[i].equals(dvd)) {

                itemsInStore[i] = null;

                System.out.println( dvd.getTitle() + "has been deleted");

                break;

            }

        }

    }

}

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);

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

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

    store.addDVD(dvd1);

    store.addDVD(dvd2);

    store.removeDVD(dvd1);

  }

}

Kết quả:

A screen shot of a computer

Description automatically generated

# Re-organize your projects

A screenshot of a computer

Description automatically generated

# String, StringBuilder and StringBuffer

# Release flow demonstration