BÁO CÁO THỰC HÀNH LAP 3  
LẬP TRÌNH HƯỚNG ĐỐI TƯỢNG

Contents

[1. Branch your repository 2](#_Toc150790081)

[2. Working with method overloading 3](#_Toc150790082)

[2.1 Overloading by differing types of parameter 3](#_Toc150790083)

[2.2. Overloading by differing the number of parameters 4](#_Toc150790084)

[3. Passing parameter 5](#_Toc150790085)

[4. Use debug run: 7](#_Toc150790086)

[4.1. Debugging Java in Eclipse 7](#_Toc150790087)

[4.2. Example of debug run for the **swap** method of **TestPassingParameter** 7](#_Toc150790088)

[4.2.1. Setting, deleting & deactivate breakpoints 7](#_Toc150790089)

[4.2.2. Run in Debug mode 7](#_Toc150790090)

[4.2.3. Step Into, Step Over, Step Return, Resume 7](#_Toc150790091)

[4.2.4. Investigate value of variables 8](#_Toc150790092)

[4.2.5. Change value of variables 8](#_Toc150790093)

[5. Classifier Member and Instance Member 8](#_Toc150790094)

[6. Open the **Cart** class 8](#_Toc150790095)

[7. Implement the **Store** class 11](#_Toc150790096)

[8. Re-organize your projects 12](#_Toc150790097)

[9. **String**, **StringBuilder** and **StringBuffer** 13](#_Toc150790098)

[10. Release flow demonstration 15](#_Toc150790099)

[10.1. Hypothesis 15](#_Toc150790100)

[10.2. Demonstration 15](#_Toc150790101)

[Hình 1 Branch your repository 3](#_Toc150790184)

[Hình 2. Truyền mảng 3](#_Toc150790185)

[Hình 3 Truyền số lượng tham số tùy ý 4](#_Toc150790186)

[Hình 4 Overloading by differing the number of parameters 4](#_Toc150790187)

[Hình 5 TestPassingParameter.java 5](#_Toc150790188)

[Hình 6 Viết lại swap() 6](#_Toc150790189)

[Hình 7 Breakpoint 7](#_Toc150790190)

[Hình 8 Step Into, Step Over, Step Return, Resume 7](#_Toc150790191)

[Hình 9 Investigate value of variables 8](#_Toc150790192)

[Hình 10 Change value of variables 8](#_Toc150790193)

[Hình 11 Classifier Member and Instance Member 8](#_Toc150790194)

[Hình 12 Show Cart 8](#_Toc150790195)

[Hình 13 toString() 9](#_Toc150790196)

[Hình 14 isMatch() 9](#_Toc150790197)

[Hình 15 CartTest.java 10](#_Toc150790198)

[Hình 16 Store.java 11](#_Toc150790199)

[Hình 17 StoreTest.java 11](#_Toc150790200)

[Hình 18 Re-organize your projects 12](#_Toc150790201)

[Hình 19 ConcatenationInLoops.java 13](#_Toc150790202)

[Hình 20 GarbageCreator.java 14](#_Toc150790203)

[Hình 21 Sử dụng StringBuiler 14](#_Toc150790204)

[Hình 22 15](#_Toc150790205)

[Hình 23 Merge branch 15](#_Toc150790206)

# 1. Branch your repository

Đã tạo 9 branch như hình 1.

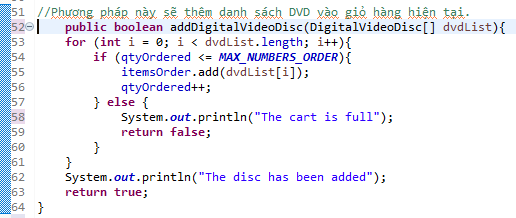
A screenshot of a computer

Description automatically generated

Hình 1 Branch your repository

# 2. Working with method overloading

## 2.1 Overloading by differing types of parameter



Hình 2. Truyền mảng

A screen shot of a computer program

Description automatically generated

Hình 3 Truyền số lượng tham số tùy ý

* 1 phương thức truyền vào tham số mảng, 1 phương thức truyền số lượng tham số tùy ý (ở đây là 2).
* Các phương thức thuộc kiểu dữ liệu boolean sẽ trả về kết quả thêm vào giỏ hàng được hay không. Sẽ thêm được nếu giỏ hàng còn chỗ và không thêm được nếu giỏ hàng hết chỗ.
* So sánh 2 phương thức :
  + Dùng mảng:
    - Ưu điểm: Rõ ràng, dễ đọc, và linh hoạt khi cần thực hiện các thao tác trên mảng.
    - Nhược điểm: Cần tạo và truyền một mảng, điều này có thể làm tăng độ phức tạp của mã khi chỉ cần truyền một loạt các giá trị
  + Dùng số lượng đối số tùy ý:
    - Ưu điểm: Đơn giản hóa quá trình gọi hàm với số lượng đối số không xác định
    - Nhược điểm: Không linh hoạt như mảng khi cần thực hiện các thao tác phức tạp trên danh sách đối số
* Trong trường hợp này, em thích sử dụng truyền vào mảng hơn

## 2.2. Overloading by differing the number of parameters

A screenshot of a computer code

Description automatically generated

Hình 4 Overloading by differing the number of parameters

# 3. Passing parameter

* Tạo class **TestPassingParameter.**

A screenshot of a computer program

Description automatically generated

Hình 5 TestPassingParameter.java

**Giải thích:**

Điều này xảy ra do cách Java xử lý tham số khi truyền vào các phương thức.

* Với phương thức swap:
  + Trong phương thức swap, ta đang thay đổi giá trị của các tham số cục bộ ‘o1’ và ‘o2’. Khi phương thức kết thúc, các tham số này mất đi và không ảnh hưởng đến giá trị của ‘jungleDVD’ và ‘cinderellaDVD’ trong ‘main’ vì chúng chỉ là các bản sao của tham số được truyền vào.
* Với phương thức ‘changeTitle’:
  + Trong phương thức ‘changeTitle’, ta đang thay đổi trạng thái của đối tượng ‘dvd’ bằng cách gọi ‘setTitle(title)’. Điều này ảnh hưởng trực tiếp đến đối tượng được truyền vào phương thức (ở đây là ‘jungleDVD’). Tuy nhiên, sau đó, ta tạo một đối tượng mới ‘DigitalVideoDisc’ và gán cho ‘dvd’, điều này không ảnh hưởng đến ‘jungleDVD’ trong ‘main’.

Vì vậy, sau khi gọi ‘swap’, ‘jungleDVD’ và ‘cinderellaDVD’ không thay đổi vì ta chỉ đang thay đổi giá trị của các tham số cục bộ. Ngược lại, sau khi gọi ‘changeTitle’, ‘jungleDVD’ thay đổi vì ta đã thay đổi trạng thái của đối tượng được truyền vào.

**Viết lại phương thức swap() và kết quả:**

A screenshot of a computer program

Description automatically generatedHình 6 Viết lại swap()

# 4. Use debug run:

## Debugging Java in Eclipse

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

### Setting, deleting & deactivate breakpoints

A screenshot of a computer program

Description automatically generated

Hình 7 Breakpoint

### Run in Debug mode

### Step Into, Step Over, Step Return, Resume

A screenshot of a computer program

Description automatically generated

Hình 8 Step Into, Step Over, Step Return, Resume

### Investigate value of variables

A screenshot of a computer

Description automatically generated

Hình 9 Investigate value of variables

### Change value of variables

A screenshot of a computer

Description automatically generated

Hình 10 Change value of variables

# Classifier Member and Instance Member



Hình 11 Classifier Member and Instance Member

# Open the **Cart** class

A screenshot of a computer code

Description automatically generated

Hình 12 Show Cart

A screen shot of a computer program

Description automatically generated

Hình 13 toString()

A screenshot of a computer code

Description automatically generated

Hình 14 isMatch()

A screenshot of a computer program

Description automatically generated

Hình 15 CartTest.java

# Implement the **Store** class

A screenshot of a computer program

Description automatically generated

Hình 16 Store.java



A screenshot of a computer program

Description automatically generated

Hình 17 StoreTest.java

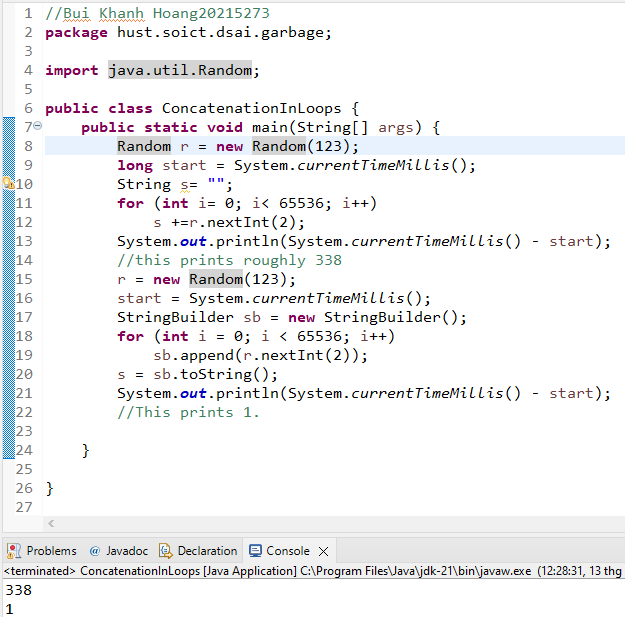
# Re-organize your projects

A screenshot of a computer

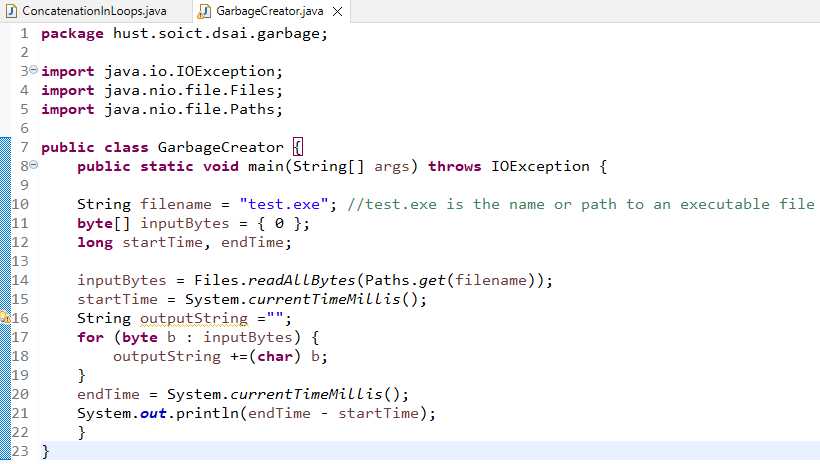
Description automatically generated

Hình 18 Re-organize your projects

# **String**, **StringBuilder** and **StringBuffer**



Hình 19 ConcatenationInLoops.java



Hình 20 GarbageCreator.java

A screenshot of a computer program

Description automatically generated

Hình 21 Sử dụng StringBuiler

# Release flow demonstration

## Hypothesis

A screenshot of a computer

Description automatically generated

Hình 22

## 10.2. Demonstration

A screenshot of a computer

Description automatically generated

Hình 23 Merge branch