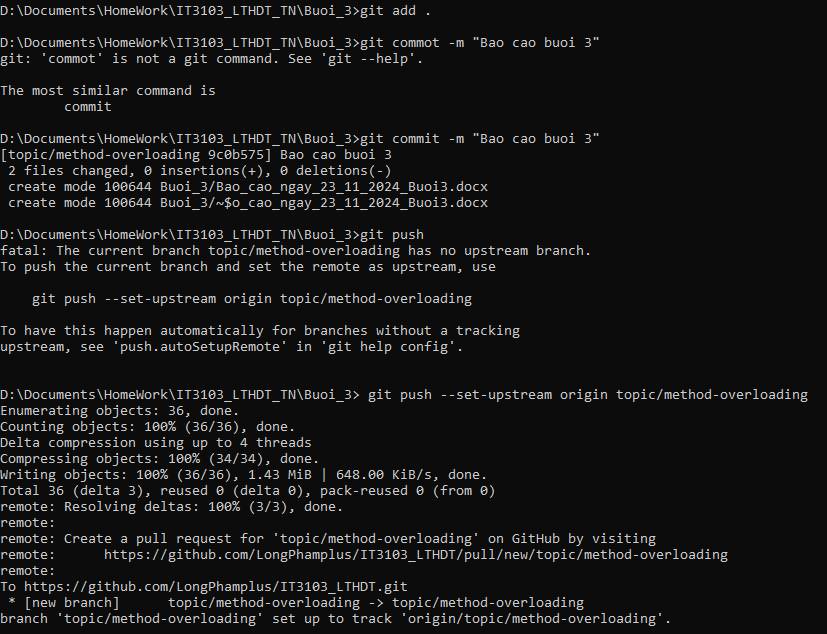
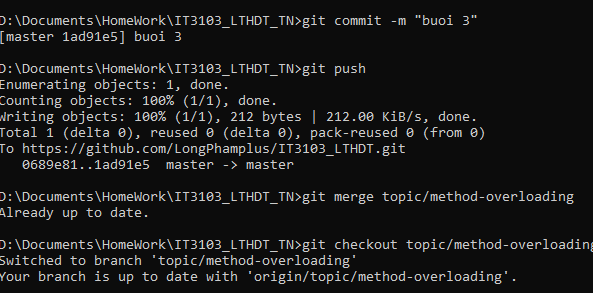
Báo cáo Lab3

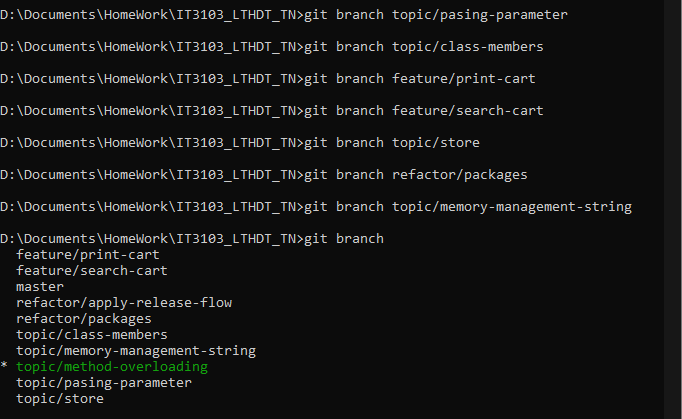
1. Branch your repository.

* Đoạn mã tải dữ liệu lên github và merge nhánh method-overloading





* Đoạn mã tạo branch kèm kết quả:



1. Working with method overloading.
   1. Overloading by differing types of parameter.

* Phương thức addDigitalVideoDisc(DigitalVideoDisc [] dvdList):

public void addDigitalVideoDisc (DigitalVideoDisc[] dvdDisc) {

int dvdLength = dvdDisc.length;

if (qtyOrdered + dvdLength <= ***MAX\_NUMBER\_ORDERED***) {

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

itemOrdered[qtyOrdered] = dvdDisc[i];

++qtyOrdered;

}

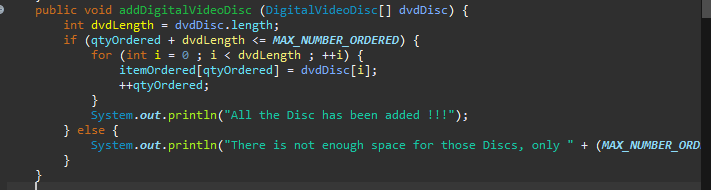
System.***out***.println("All the Disc has been added !!!");

} else {

System.***out***.println("There is not enough space for those Discs, only " + (***MAX\_NUMBER\_ORDERED*** - qtyOrdered) + " left.");

}

}



* Phương thức addDigitalVideoDisc(DigitalVideoDisc …dvdList):

public void addDigitalVideoDisc (DigitalVideoDisc ...dvdList) {

for (DigitalVideoDisc disc: dvdList) {

if (qtyOrdered < ***MAX\_NUMBER\_ORDERED***) {

itemOrdered[qtyOrdered] = disc;

++qtyOrdered;

System.***out***.println("The Disc has been added !!!");

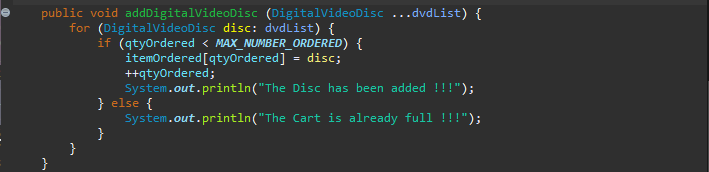
} else {

System.***out***.println("The Cart is already full !!!");

}

}

}



|  |  |  |
| --- | --- | --- |
| Phương thức | DigitalVideoDisc[] | DigitalVideoDisc… |
| Yêu cầu trước khi gọi | Cần tạo mảng trước | Truyền trực tiếp các tham số hoặc mảng |
| Giá trị truyền | Chỉ nhận mảng | Có thể nhận mảng hoặc tham số |
| Tính linh hoạt | Kém hơn, cú pháp rườm rà | Linh hoạt, cú pháp gọn gàng |

* 1. Overloading by differing the number of parameters.
* Phương thức addDigitalVideoDisc(DigitalVideoDisc dvd1,DigitalVideoDisc dvd2)

public void addDigitalVideoDisc(DigitalVideoDisc dvd1,DigitalVideoDisc dvd2) {

if (qtyOrdered + 1 < ***MAX\_NUMBER\_ORDERED***) {

itemOrdered[qtyOrdered] = dvd1;

++qtyOrdered;

itemOrdered[qtyOrdered] = dvd2;

++qtyOrdered;

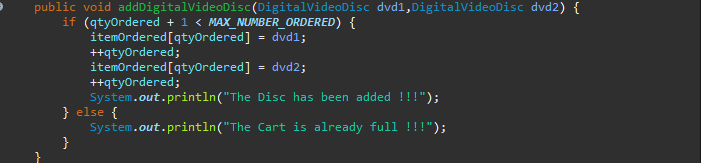
System.***out***.println("The Disc has been added !!!");

} else {

System.***out***.println("The Cart is already full !!!");

}

}



1. Passing parameter.

* Question: Is JAVA a Pass by Value or a Pass by Reference programming language?
* Answer: Trong JAVA mọi thứ đều được truyền theo chuyển tham trị (Pass by Reference).
* Hàm swap ban đầu không làm thay đổi giá trị vì khi gọi hàm swap và truyền vào o1 và o2, Java sẽ tạo 1 bản sao tham chiếu của 2 giá trị và chỉ thay đổi giá trị của 2 bản sao đó.
* Hàm changeTitle làm thay đổi giá trị vì khi gọi hàm cả 2 giá trị dvd và giá trị truyền vào đều được tham chiếu tới giá trị ban đầu nên khi thay đổi sẽ làm đổi giá trị của cả 2 đối tượng đó.
* Đoạn mã của phương thức swap() để có thể chuyển đổi 2 đối tượng:

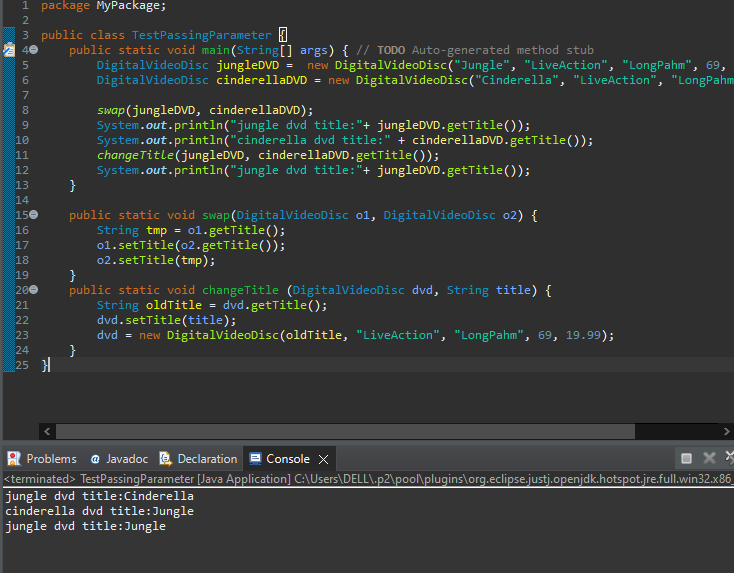
public static void swap(DigitalVideoDisc o1, DigitalVideoDisc o2) {

String tmp = o1.getTitle();

o1.setTitle(o2.getTitle());

o2.setTitle(tmp);

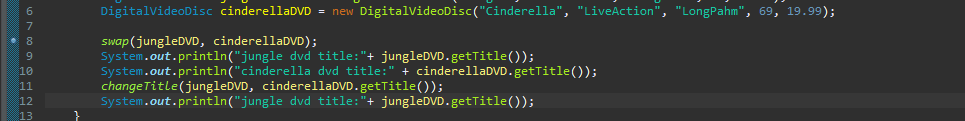
}



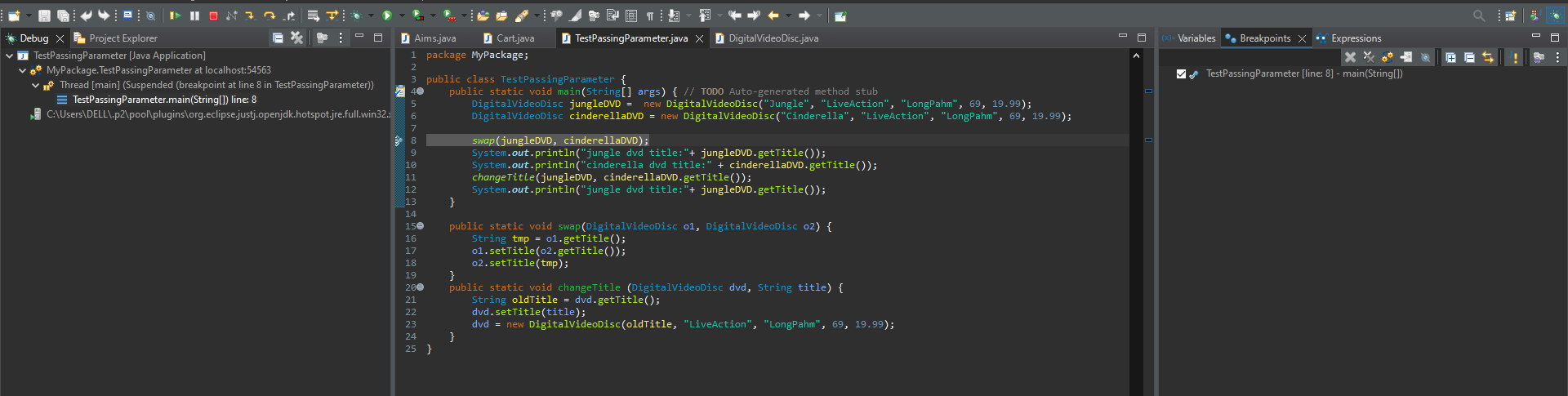
1. Use debug run:

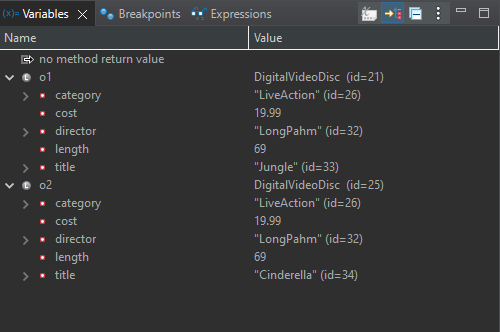
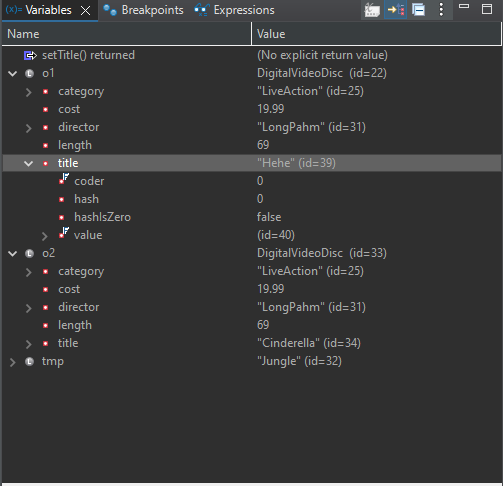
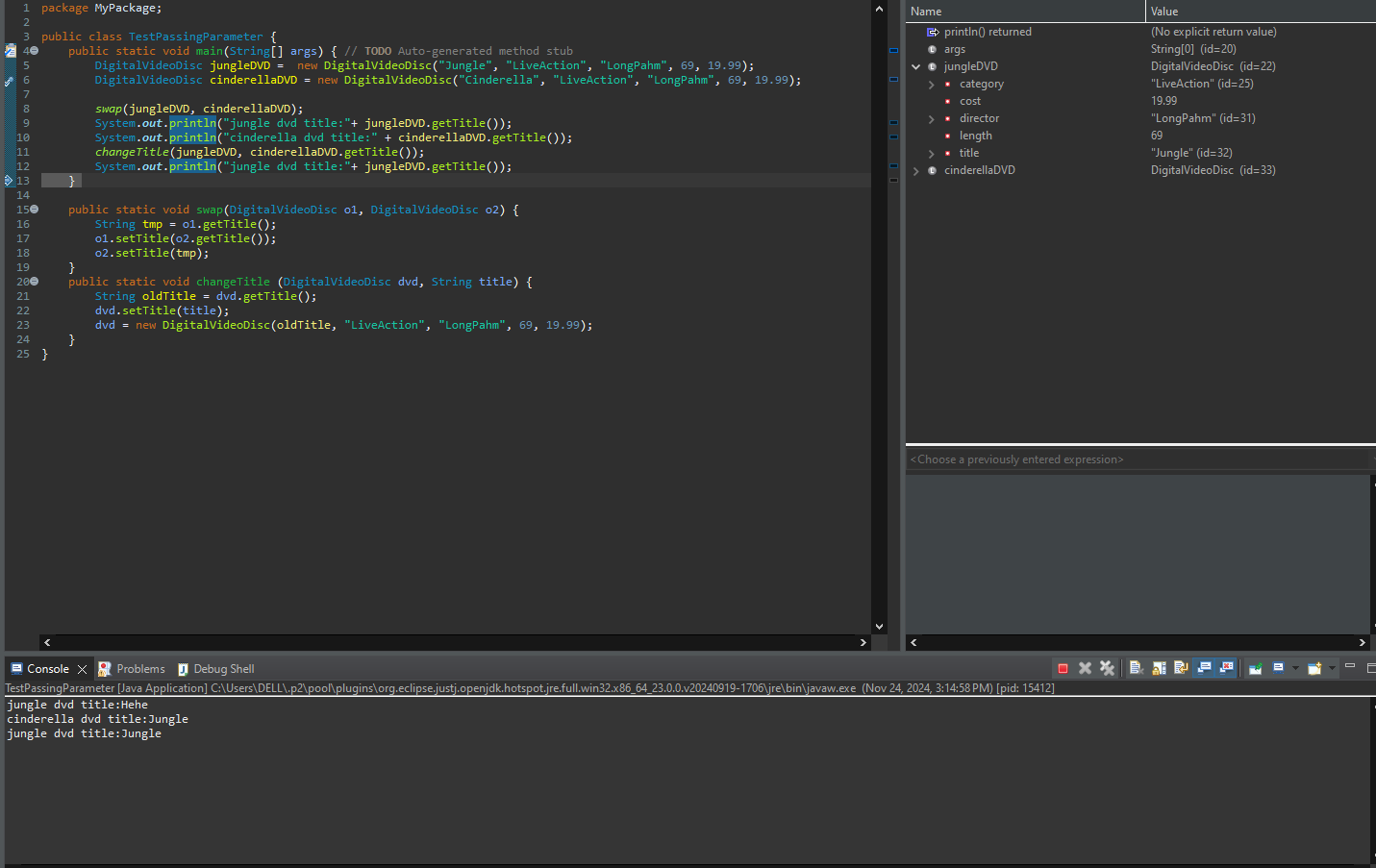
4.2. Example of debug run for the swap method of TestPassingParameter .

4.2.1. Setting, deleting & deactivate breakpoints:



4.2.2. Run in Debug mode:



* Giá trị các biến trước khi thay đổi
* 
* Giá trị các biến sau khi thay đổi:
* 
* Out put:
* 

1. Classifier Member and Instance Member

* Đoạn mã được sử dụng:

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

super();

this.id = id;

this.title = title;

this.category = category;

this.director = director;

this.length = length;

this.cost = cost;

*nbDigitalVideoDiscs*++;

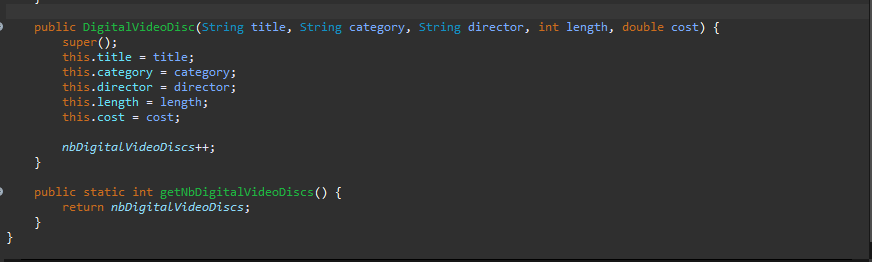
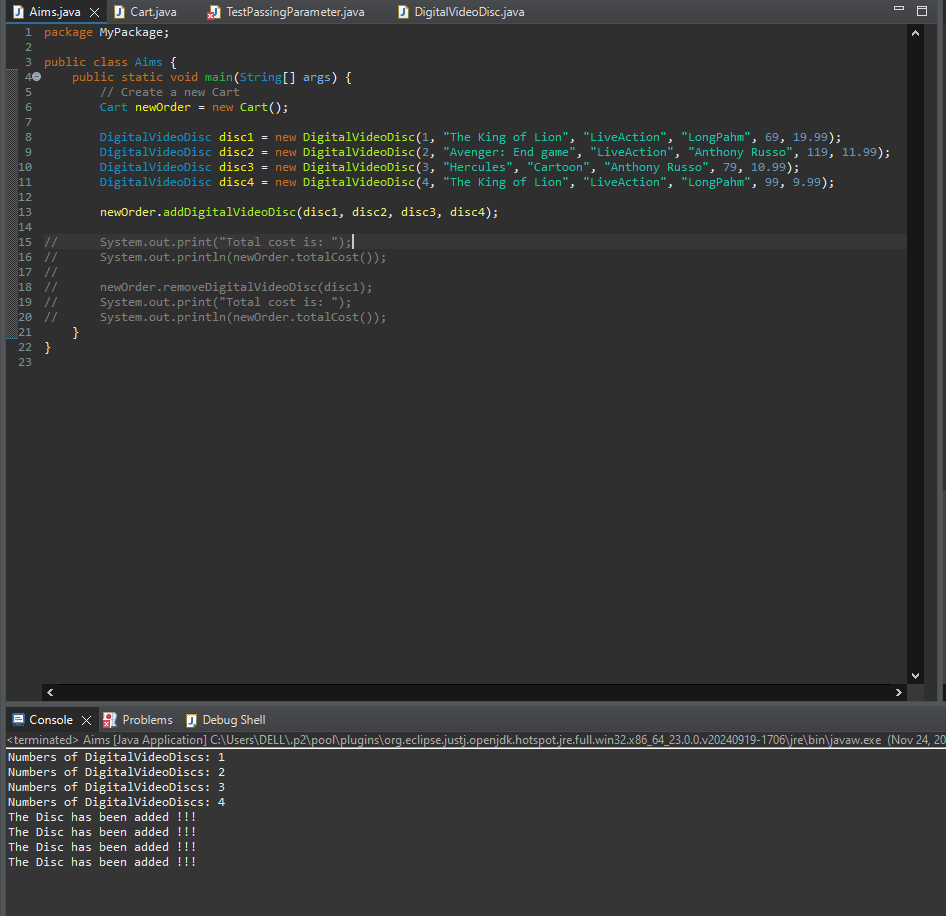
System.***out***.println("Numbers of DigitalVideoDiscs: " + DigitalVideoDisc.*getNbDigitalVideoDiscs*());

}

public static int getNbDigitalVideoDiscs() {

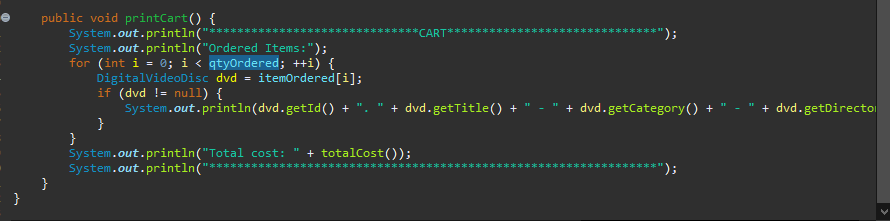
return *nbDigitalVideoDiscs*;

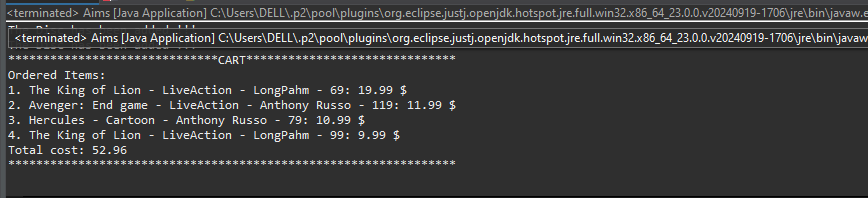
}

* 
* Ví dụ
* 

1. Open the Cart class

* Đoạn mã gọi hàm printCart:
* public void printCart() {
* System.***out***.println("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*CART\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");
* System.***out***.println("Ordered Items:");
* for (int i = 0; i < qtyOrdered; ++i) {
* DigitalVideoDisc dvd = itemOrdered[i];
* if (dvd != null) {
* System.***out***.println(dvd.getId() + ". " + dvd.getTitle() + " - " + dvd.getCategory() + " - " + dvd.getDirector() + " - " + dvd.getLength() + ": " + dvd.getCost() + " $ ");
* }
* }
* System.***out***.println("Total cost: " + totalCost());
* System.***out***.println("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");
* }



* Output: 
* Đoạn mã của 2 hàm search:

public void searchDVDById(int id) {

boolean check = false;

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

DigitalVideoDisc dvd = itemOrdered[i];

if (dvd != null && id == dvd.getId()) {

System.***out***.println(dvd.getId() + ". " + dvd.getTitle() + " - " + dvd.getCategory() + " - " + dvd.getDirector() + " - " + dvd.getLength() + ": " + dvd.getCost() + " $ ");

check = true;

}

}

if (!check) {

System.***out***.println("DVD id " + id + " not found.");

}

}

public void searchDVDByTitle(String title) {

boolean check = false;

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

DigitalVideoDisc dvd = itemOrdered[i];

if (dvd != null && title.equals(dvd.getTitle())) {

System.***out***.println(dvd.getId() + ". " + dvd.getTitle() + " - " + dvd.getCategory() + " - " + dvd.getDirector() + " - " + dvd.getLength() + ": " + dvd.getCost() + " $ ");

check = true;

}

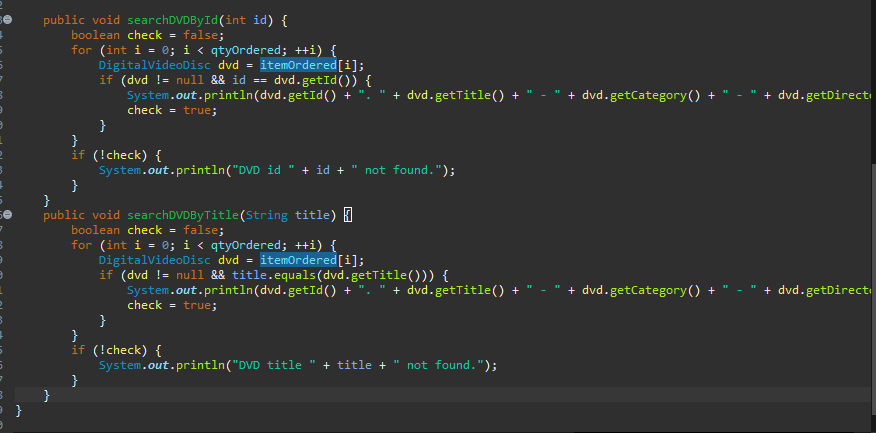
}

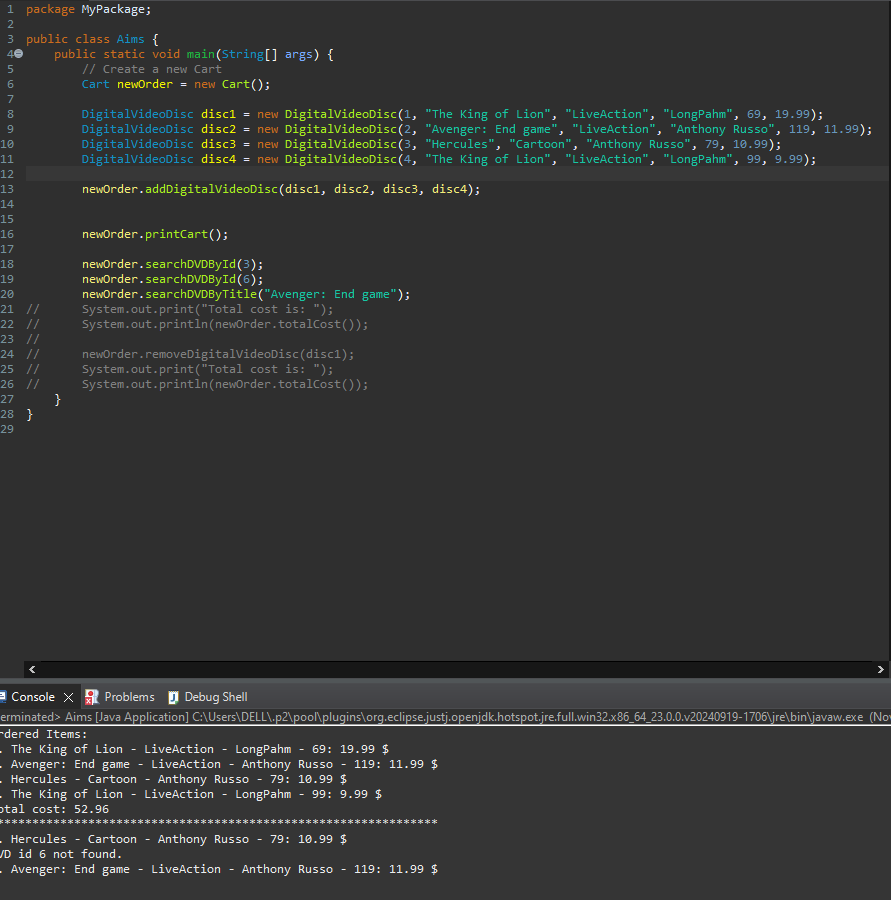
if (!check) {

System.***out***.println("DVD title " + title + " not found.");

}

}



* Output: 

1. Implement the Store class

* Đoạn mã của Store class:

package MyPackage;

public class Store {

public static final int ***MAX\_NUMBER\_ORDERED*** = 20;

private DigitalVideoDisc itemOrdered[] = new DigitalVideoDisc[***MAX\_NUMBER\_ORDERED***];

private int qtyOrdered = 0;

public void addDVD (DigitalVideoDisc ...dvdList) {

for (DigitalVideoDisc disc: dvdList) {

if (qtyOrdered < ***MAX\_NUMBER\_ORDERED***) {

itemOrdered[qtyOrdered] = disc;

++qtyOrdered;

System.***out***.println("The Disc has been added !!!");

} else {

System.***out***.println("The Cart is already full !!!");

}

}

}

public void removeDVD (DigitalVideoDisc disc) {

boolean found = false;

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

if (itemOrdered[i].equals(disc)) {

found = true;

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

itemOrdered[j] = itemOrdered[j + 1];

}

itemOrdered[qtyOrdered] = null;

--qtyOrdered;

System.***out***.println("The Disc has been removed successfully !!!");

break;

}

}

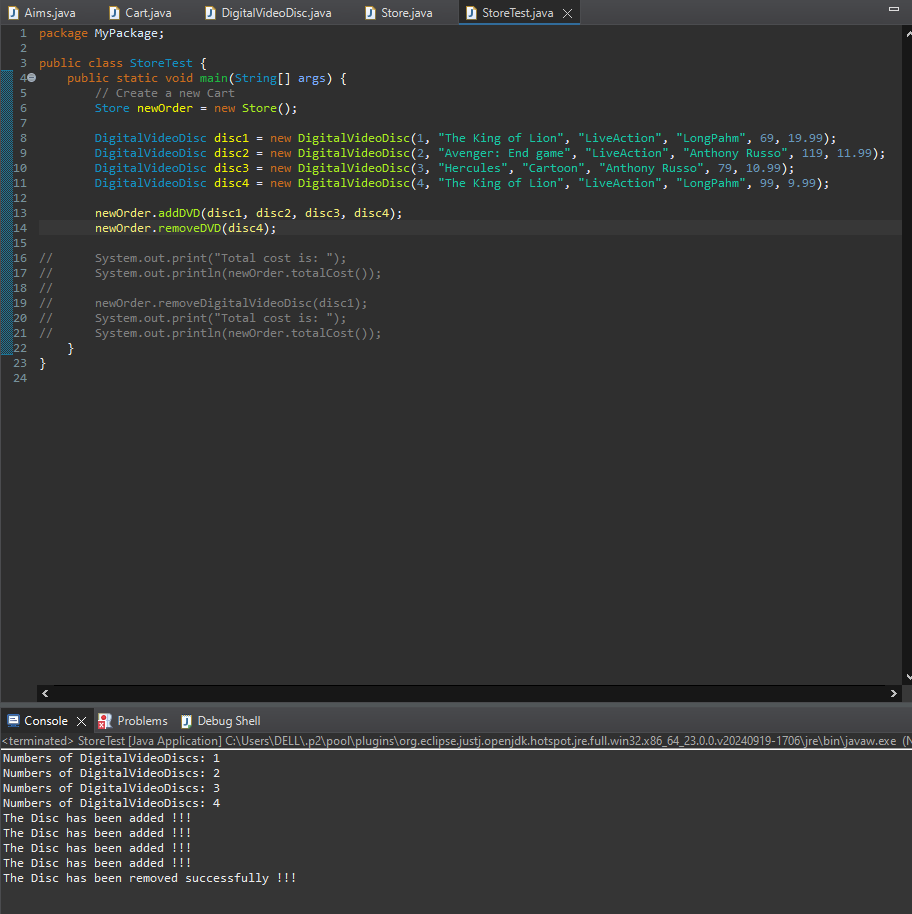
if (!found) {

System.***out***.println("Disk not found !!!");

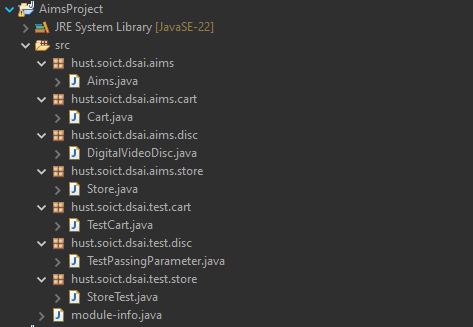
}

}

}

* Output:
* 

1. Re-organize your projects

* 

1. String, StringBuilder and StringBuffer

* Đoạn mã của GabageCreator:

package hust.soict.globalict.garbage;

public class GabageCreator {

public static void main(String[] args) {

System.***out***.println("Starting Garbage Creator...");

String s = "";

try {

for (int i = 0; i < 1\_000\_000; i++) {

// Inefficient string concatenation

s += "A";

// Optional: Print memory usage periodically

if (i % 100\_000 == 0) {

System.***out***.println("Iteration: " + i + " - Free memory: " + Runtime.*getRuntime*().freeMemory());

}

}

} catch (OutOfMemoryError e) {

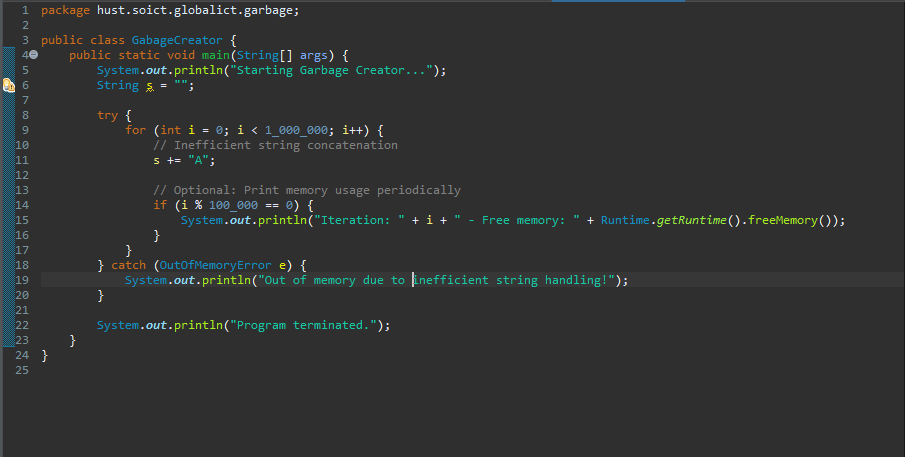
System.***out***.println("Out of memory due to inefficient string handling!");

}

System.***out***.println("Program terminated.");

}

}

* 
* Đoạn mã của NoGabage:

package hust.soict.globalict.garbage;

public class NoGarbage {

public static void main(String[] args) {

System.***out***.println("Starting No Garbage...");

StringBuilder sb = new StringBuilder();

try {

for (int i = 0; i < 1\_000\_000; i++) {

// Efficient string manipulation using StringBuilder

sb.append("A");

// Optional: Print memory usage periodically

if (i % 100\_000 == 0) {

System.***out***.println("Iteration: " + i + " - Free memory: " + Runtime.*getRuntime*().freeMemory());

}

}

} catch (OutOfMemoryError e) {

System.***out***.println("Out of memory! But this is unlikely with StringBuilder.");

}

System.***out***.println("Final string length: " + sb.length());

System.***out***.println("Program terminated gracefully.");

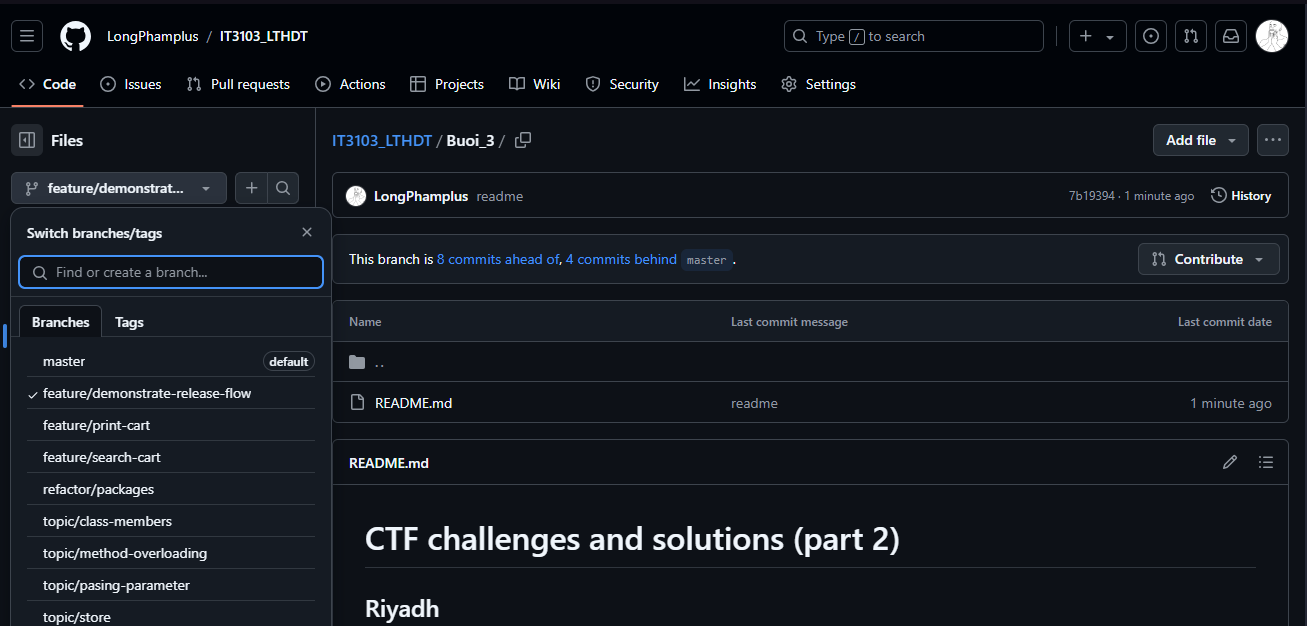
}

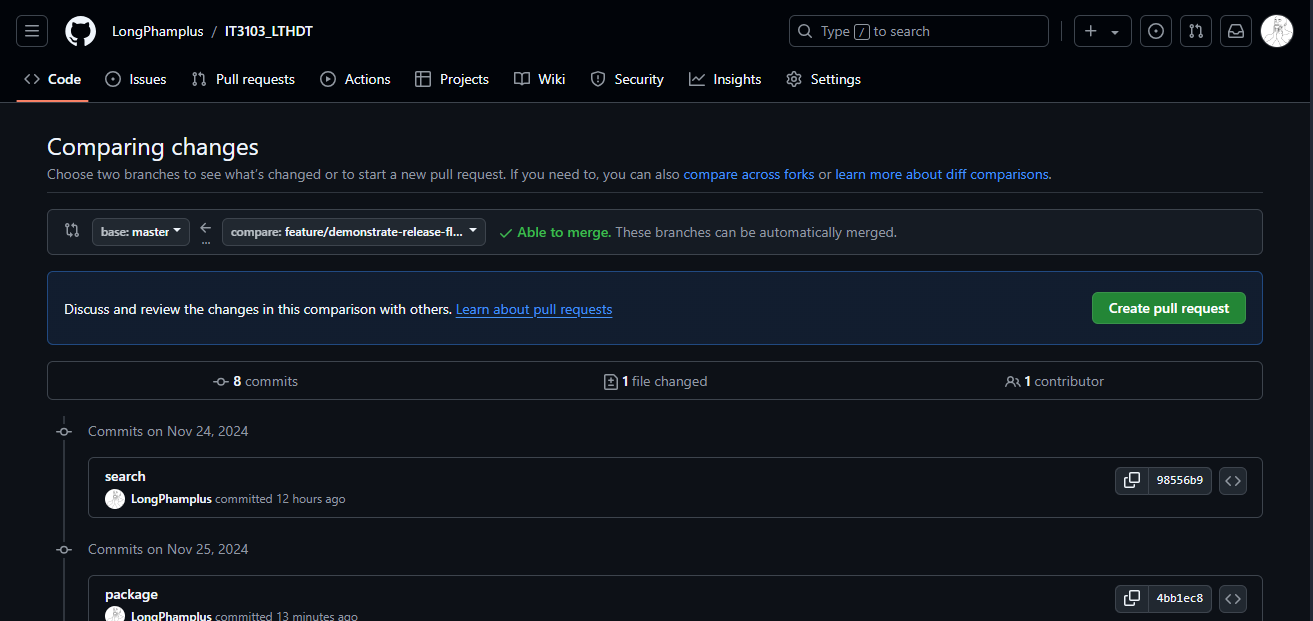
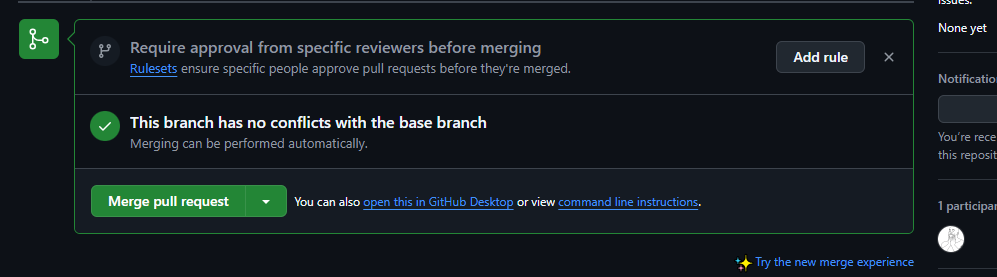
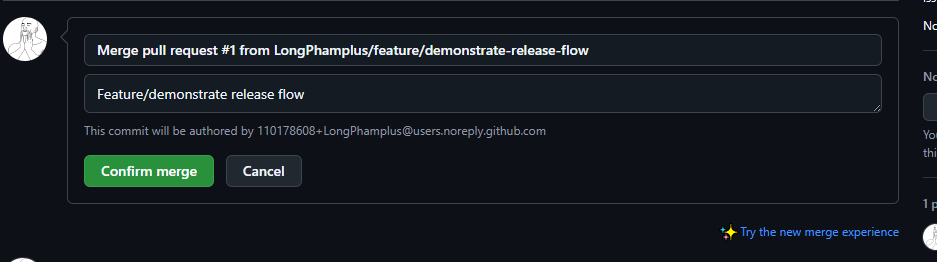
}

* 

1. Release flow demonstration

* Ảnh tạo branch feature/demonstrate-release-flow và push 1 file readme.



* Ảnh tạo pull request:
* 
* Merge request:
* 
* Confirm merge:
* 
* Merge successfully