

Lập trình hướng đối tượng

Lecturer: Trinh Thanh Trung, trungtt@soict.hust.edu.vn

Lab 03: Kỹ thuật hướng đối tượng cơ bản

Trong phòng thí nghiệm này, bạn sẽ thực hành với:

- Làm việc với quy trình phát hành
- Phương pháp quá tải
- Truyền tham số
- Thành viên trình phân loại so với thành viên phiên bản
- Thực hành quản lý bộ nhớ với String và StringBuffer và các trường hợp khác
- Gỡ lỗi với Eclipse
- Tổ chức lại dự án của bạn bằng cách tạo các gói để quản lý các lớp trong Eclipse

0. Nộp bài tập

Đối với lớp học phòng thí nghiệm này, bạn sẽ phải nộp công việc của mình hai lần, cụ thể:

- **Ngay sau giờ học:** đối với thời hạn này, bạn nên bao gồm bất kỳ công việc nào bạn đã làm trong lớp học phòng thí nghiệm .
- **10 giờ tối ba ngày sau lớp học:** đối với thời hạn này, bạn nên bao gồm **mã nguồn** của tất cả các phần của phòng thực hành này, vào một nhánh cụ thể là "**phát hành / lab03**" của kho lưu trữ hợp lệ.

Sau khi hoàn thành tất cả các bài tập trong phòng thí nghiệm, bạn phải cập nhật sơ đồ trường hợp sử dụng và sơ đồ lớp học của dự án AIMS.

Mỗi học sinh phải nộp bài làm của mình và không cho hoặc nhận viện trợ không được phép. Nếu không, chúng tôi sẽ áp dụng các phương pháp cực đoan để đo lường để ngăn chặn gian lận. P chothuê ghi lại câu trả lời cho tất cả các câu hỏi vào một tệp văn bản có tên "**câu trả lời.txt**" và gửi nó trong kho lưu trữ của bạn.

1. Phân nhánh kho lưu trữ của bạn

Ngày này qua ngày khác, kho lưu trữ của bạn ngày càng trở nên tinh vi hơn, điều này khiến mã của bạn khó quản lý hơn. May mắn thay, quy trình làm việc Git có thể giúp bạn giải quyết vấn đề này. Quy trình làm việc Git là một **công thức cho cách sử dụng Git** để kiểm soát mã nguồn một cách nhất quán và hiệu quả. Release Flow là một quy trình làm việc Git nhẹ nhưng hiệu quả giúp các nhóm hợp tác với quy mô lớn và bất kể chuyên môn kỹ thuật. Tham khảo tệp ¹**Release-Flow-Guidelines.pdf** để biết hướng dẫn chi tiết hơn.

Áp dụng Quy trình phát hành là bắt buộc từ phòng thí nghiệm này trở đi.

Tuy nhiên, chúng tôi sẽ sử dụng phiên bản sửa đổi của Luồng phát hành để đơn giản.

¹ <https://docs.microsoft.com/en-us/azure/devops/repos/git/git-branching-guidance?view=azure-devops>

- Chúng ta có thể tạo ra bao nhiêu chi nhánh tùy thích.
- Chúng tôi đặt tên cho các chi nhánh với những cái tên có ý nghĩa. Xem Bảng 1-Chính sách phân nhánh.
- Tốt hơn hết chúng tôi nên **giữ các chi nhánh càng gần** với chủ càng **tốt**; nếu không, chúng ta có thể phải đối mặt với địa ngục hợp nhất.
- Nói chung, khi chúng ta hợp nhất một nhánh với nguồn gốc của nó, nhánh đó đã trở thành lịch sử. Chúng tôi thường không chạm vào nó lần thứ hai.
- **Chúng ta phải tuân thủ nghiêm ngặt chính sách phát hành chi nhánh. Những người khác thì linh hoạt.**

| Nhánh | Đặt tên hiệp định | Nguồn gốc | Hợp nhất thành | Mục đích |
|--------------------------|---------------------------------------------------------------------------------|-----------|-----------------------|--------------------------------------------------------|
| Tính năng hoặc chủ đề | + feature/feature-name+ feature/ feature-area/feature-name + Chủ đề/Mô tả | chủ | chủ | Thêm tính năng hoặc chủ đề mới |
| Sửa lỗi | sửa lỗi / mô tả | chủ | chủ | Sửa lỗi |
| | | tính năng | tính năng | |
| hotfix | hotfix/mô tả | phát hành | Phát hành & Master[1] | Khắc phục lỗi trong bài tập đã gửi sau thời hạn |
| tái cấu trúc | tái cấu trúc / mô tả | chủ | chủ | Tái cấu trúc |
| | | tính năng | tính năng | |
| phát hành | phát hành / labXX | chủ | không ai | Gửi bài tập [2] |

Bảng 1: Chính sách phân nhánh

[1] Nếu chúng tôi muốn cập nhật các giải pháp của bạn trong vòng một tuần sau thời hạn, chúng tôi có thể tạo một chi nhánh hotfix mới (ví dụ: hotfix / stop-the-world). Sau đó, chúng tôi hợp nhất nhánh hotfix với nhánh chính và với nhánh phát hành cho bài tập được gửi lần cuối (ví dụ: release / lab05). Trong trường hợp chúng tôi đã tạo một nhánh phát hành cho nhiệm vụ tuần hiện tại (ví dụ: release/lab06), chúng tôi có thể hợp nhất nhánh hotfix với nhánh phát hành hiện tại **nếu cần** hoặc chúng tôi có thể xóa và sau đó tạo lại nhánh phát hành hiện tại.

[2] **Các phiên bản mới nhất của các dự án trong nhánh phát hành đóng vai trò là bài tập đã gửi**

Hãy sử dụng Release Flow làm quy trình làm việc Git của chúng ta và áp dụng nó để tái cấu trúc kho của chúng ta.

Bước 1: Tạo branch mới trong kho lưu trữ cục bộ của chúng ta. Chúng tôi tạo một luồng tái cấu trúc / áp dụng-phát hành nhánh mới từ nhánh chính của chúng tôi .

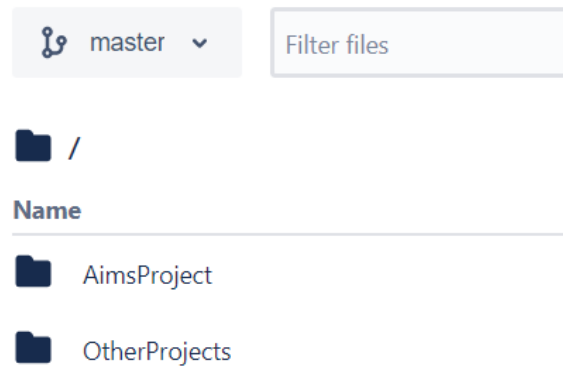
Bước 2: Thực hiện các thay đổi của chúng tôi, kiểm tra chúng và đẩy chúng. Chúng tôi di chuyển các phiên bản mới nhất của tất cả các tệp mới nhất của chúng tôi từ các phòng thí nghiệm trước đó sao cho chúng trực tiếp thuộc nhánh chính.

Xem <https://www.atlassian.com/git/tutorials/undoing-changes> để hoàn tác các thay đổi trong trường hợp có sự cố. Để cải thiện thông báo cam kết, hãy xem <https://thoughtbot.com/blog/5-useful-tips-for-a-better-commit-message>.

Bước 3: Thực hiện yêu cầu kéo để đánh giá từ đồng đội của chúng tôi. ²Chúng tôi bỏ qua bước này vì chúng tôi đang solo trong kho lưu trữ này. Tuy nhiên, tốt hơn hết là chúng tôi không bao giờ bỏ qua bước này khi chúng tôi làm việc theo nhóm.

Bước 4: Hợp nhất các chi nhánh. Hợp nhất nhánh mới refactor/apply-release-flow vào nhánh chính.

Kết quả được thể hiện trong hình dưới đây.



Hình 1. Kết quả hợp nhất

Gợi ý:

Các bước điển hình cho một chi nhánh mới:

- ☐ Tạo và chuyển sang một nhánh mới (ví dụ: abc) trong kho lưu trữ cục bộ: **git checkout -b abc**
- ☐ Thực hiện sửa đổi trong repo cục bộ
- ☐ Cam kết thay đổi trong repo cục bộ: **git commit -m "What you had change"**
- ☐ Tạo một branch mới (ví dụ: abc) trong repo từ xa (GitHub thông qua GUI)
- ☐ Đẩy nhánh cục bộ đến nhánh từ xa: **git push origin abc**
- ☐ Hợp nhất nhánh từ xa (ví dụ: abc) với nhánh chính (GitHub thông qua GUI)

Sau khi hoàn thành tất cả các nhiệm vụ của tuần đó và hợp nhất tất cả các nhánh thành nhánh chính, bạn nên tạo một nhánh release/labxx từ master trong repo từ xa (GitHub).

Ví dụ, trong phòng thí nghiệm3, có thể có 9 nhiệm vụ chính. Vì vậy, một cách khả thi để áp dụng luồng phát hành là tạo 9 nhánh:

- Tạo **refactor/apply-release-flow** nhánh để tái cấu trúc kho lưu trữ theo luồng phát hành
- Tạo một **chủ đề chi nhánh** / phương pháp quá tải cho bài tập về quá tải phương pháp
- Tạo một **topic nhánh/passing-parameter** cho bài tập mà bạn điều tra về việc truyền tham số của Java
- Tạo một **chủ đề chi nhánh/các thành viên trong lớp** cho bài tập mà bạn thực hành với thành viên trình phân loại và thành viên phiên bản
- Tạo một **tính năng chi nhánh** / giỏ in để thực hiện các mục in trong tính năng giỏ hàng
- Tạo một tính năng chi nhánh / giỏ hàng tìm kiếm để thực hiện các mặt hàng tìm kiếm trong tính năng giỏ hàng
- Tạo **branch topic/store** để triển khai class Store
- Tạo **refactor/packages** branch để tái cấu trúc các dự án trong kho lưu trữ của bạn bằng cách sử dụng các gói
- Tạo một **branch topic/memory-management-string** cho bài tập StringBuffer & StringBuilder

² <https://www.atlassian.com/git/tutorials/making-a-pull-request>

Tham khảo phần trình diễn của Luồng phát hành trong phần cuối của phòng thực hành này để biết thêm hướng dẫn chi tiết.

2. Làm việc với phương pháp quá tải

Quá tải phương pháp cho phép các phương thức khác nhau có **cùng tên** nhưng chữ ký khác nhau trong đó chữ ký có thể khác nhau theo số lượng tham số đầu vào hoặc loại (các) tham số đầu vào hoặc **cả hai**.

2.1 Quá tải bởi các loại tham số khác nhau

- **Mở Eclipse**
- **Mở JavaProject có tên "AimsProject" mà bạn đã tạo trong phòng thí nghiệm trước đó.**
- **Mở giỏ hàng lớp. java:** bạn sẽ quá tải phương thức **addDigitalVideoDisc** mà bạn đã tạo lần trước.
- + Phương thức hiện tại có một tham số đầu vào của lớp **DigitalVideoDisc**
- + Bạn sẽ tạo một phương thức mới có cùng tên nhưng với loại tham số khác nhau.

addDigitalVideoDisc(DigitalVideoDisc [] dvdList)

Phương pháp này sẽ thêm danh sách các đĩa DVD vào giỏ hàng hiện tại.

+ Cố gắng thêm một phương thức **addDigitalVideoDisc** cho phép vượt qua một số lượng tùy ý các đối số cho dvd. So sánh với một tham số mảng. Bạn thích gì trong trường hợp này?

2.2. Quá tải bằng cách thay đổi số lượng thông số

- **Tiếp tục tập trung vào lớp Cart**
- **Tạo phương thức mới có tên addDigitalVideoDisc**
- + Chữ ký của phương pháp này có hai tham số như sau:

addDigitalVideoDisc(DigitalVideoDisc dvd1,DigitalVideoDisc dvd2)

3. Truyền tham số

- Câu hỏi: **JAVA là ngôn ngữ lập trình Pass by Value hay Pass by Reference?**

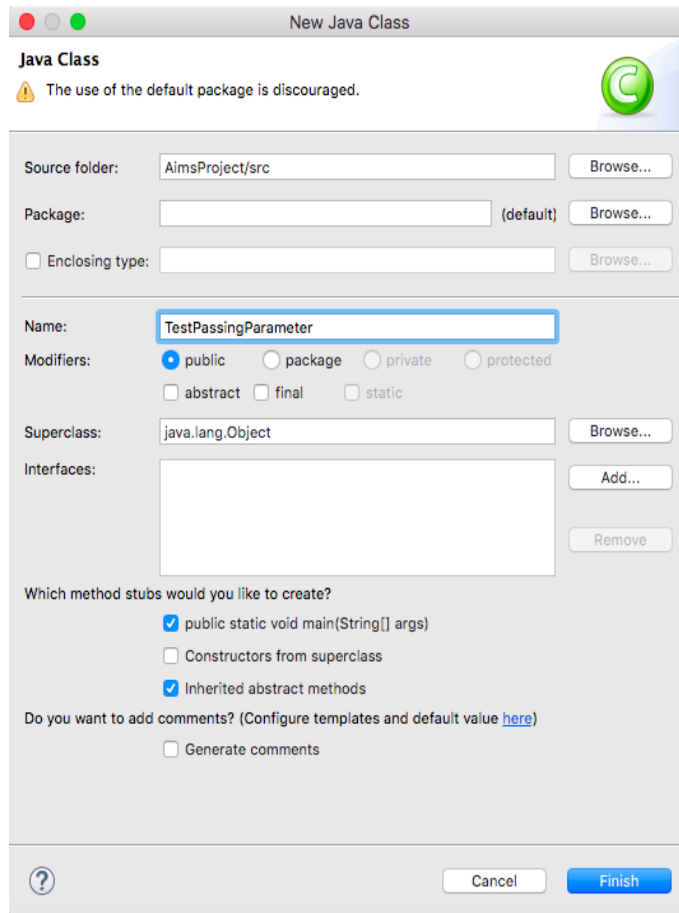
Trước hết, chúng tôi nhớ lại ý nghĩa của việc vượt qua **giá trị** hoặc **vượt qua bằng tham chiếu**.

- Pass by value: Các giá trị tham số phương thức được sao chép sang một biến khác và sau đó đối tượng được sao chép được truyền đến phương thức. Đó là lý do tại sao nó được gọi là vượt qua giá trị
- Chuyển qua tham chiếu: Một bí danh hoặc tham chiếu đến tham số thực tế được chuyển đến phương thức. Đó là lý do tại sao nó được gọi là vượt qua bằng cách tham khảo.

Bây giờ, bạn sẽ thực hành với lớp **DigitalVideoDisc** để kiểm tra cách JAVA truyền các tham số. Đối với bài tập này, bạn sẽ cần tạm thời thêm một setter cho tiêu đề thuộc tính của lớp DigitalVideoDisc.

Tạo một class mới có tên **TestPassingParameter** trong project hiện tại

- Kiểm tra tùy chọn để tạo phương thức chính trong lớp này như trong Hình 2



Hình 2. Tạo TestPassingParameter bằng Eclipse

Trong phương thức **main()** của lớp, gõ code dưới đây trong Hình 3:

```
public class TestPassingParameter {
    public static void main(String[] args) {
        // TODO Auto-generated method stub
        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);
    }
}
```

rừng

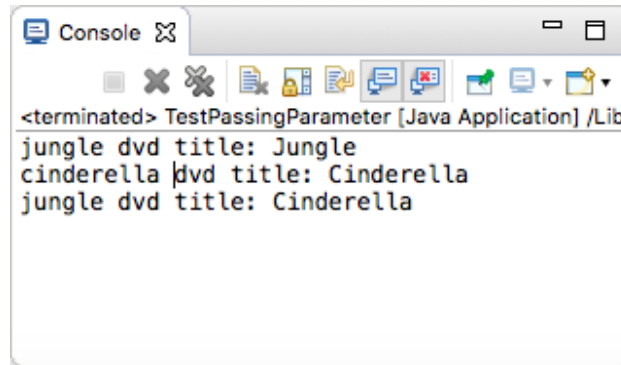
Đĩa DVD
("Rừng rậm")

lọ lemDVD

Đĩa DVD
("Cô bé lọ
lem")

Hình 3. Mã nguồn của TestPassingParameter.

Kết quả trong bảng điều khiển như sau:



Hình 4. Kết quả(1)

Để kiểm tra xem một ngôn ngữ lập trình đang truyền theo giá trị hay truyền qua tham chiếu, chúng tôi thường sử dụng phương thức **hoán đổi**. Phương pháp này nhằm mục đích hoán đổi một đối tượng sang một đối tượng khác.

- Sau cuộc gọi **hoán đổi (jungleDVD, cinderellaDVD)** tại sao tiêu đề của hai đối tượng này vẫn còn?
- Sau cuộc gọi của **changeTitle (jungleDVD, cinderellaDVD.getTitle ())** tại sao tiêu đề của JungleDVD lại thay đổi?

Sau khi tìm được câu trả lời cho những câu hỏi trên, các bạn sẽ hiểu rằng JAVA luôn là ngôn ngữ lập trình giá trị vượt qua.

Vui lòng viết một phương thức swap() có thể hoán đổi chính xác hai đối tượng.

4. Sử dụng tính năng gỡ lỗi chạy:

4.1. Gỡ lỗi Java trong Eclipse

Video: <https://www.youtube.com/watch?v=9gAjiQc4bPU&t=8s>

Gỡ lỗi là quá trình thường xuyên để xác định vị trí và loại bỏ các lỗi, lỗi hoặc bất thường khỏi chương trình. Đó là một kỹ năng bắt buộc phải có đối với bất kỳ nhà phát triển Java nào vì nó giúp tìm ra các lỗi tinh tế không thể nhìn thấy trong quá trình đánh giá mã hoặc chỉ xảy ra khi một điều kiện cụ thể xảy ra. Eclipse Java IDE cung cấp nhiều công cụ gỡ lỗi và chế độ xem được nhóm lại trong Debug Perspective để giúp bạn với tư cách là nhà phát triển gỡ lỗi một cách hiệu quả và hiệu quả.

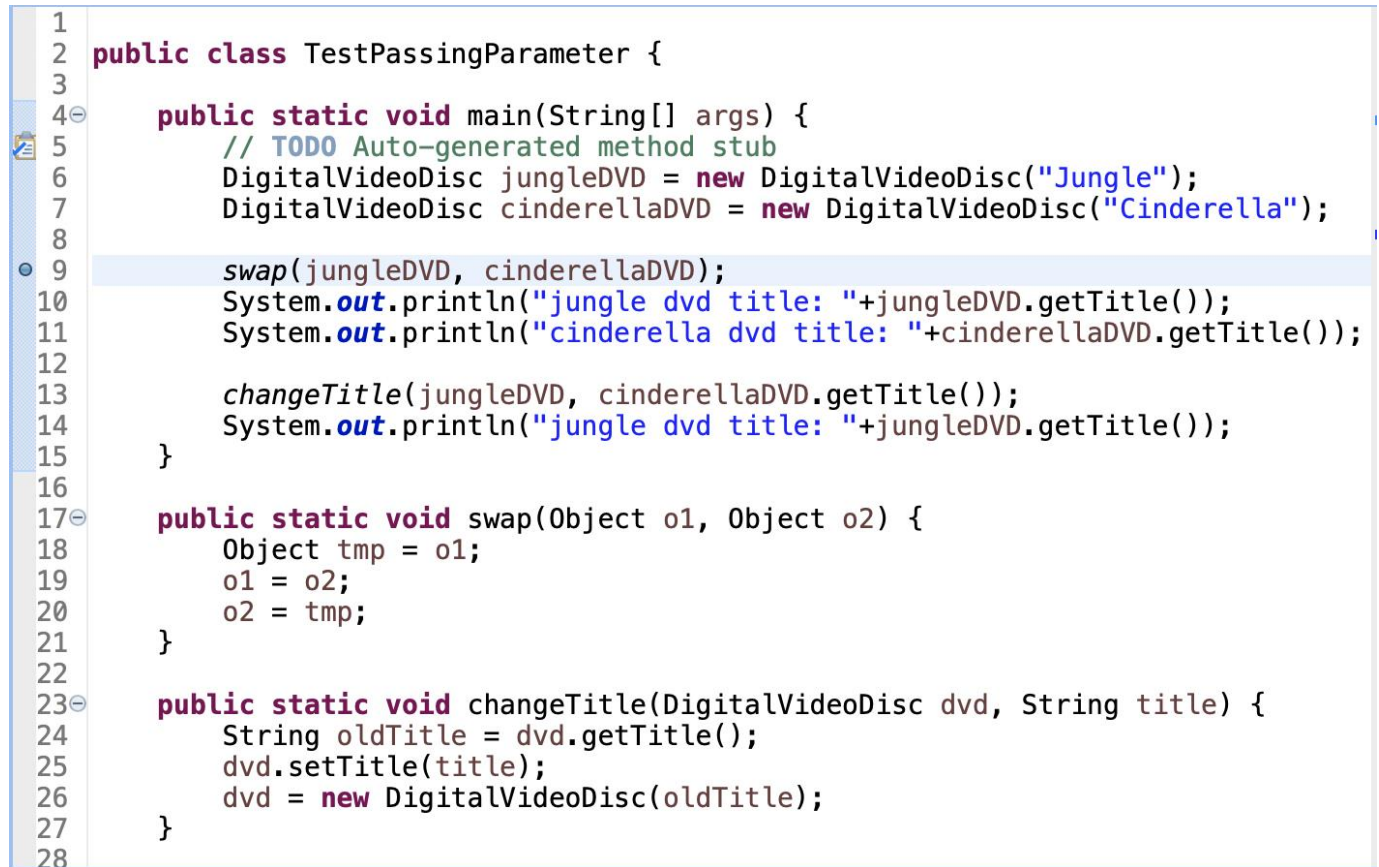
Chạy gỡ lỗi cho phép bạn chạy một chương trình tương tác trong khi xem mã nguồn và các biến trong quá trình thực thi. Một **điểm ngắt** trong mã nguồn chỉ định nơi thực thi chương trình sẽ dừng lại trong quá trình gỡ lỗi. **Khi chương trình bị dừng, bạn có thể điều tra các biến, thay đổi nội dung của chúng, v.v.**

4.2. Ví dụ về debug run cho phương thức swap của TestPassingParameter

4.2.1. Cài đặt, xóa và hủy kích hoạt điểm ngắt:

Để đặt điểm ngắt, hãy đặt con trỏ trên dòng cần gỡ lỗi, nhấn giữ Ctrl+Shift và nhấn B để bật điểm ngắt. Một chấm màu xanh lam ở phía trước dòng sẽ xuất hiện (Hình 5). Ngoài ra, bạn có thể nhấp chuột phải vào lề

trái của dòng trong trình chỉnh sửa Java và chọn Toggle Breakpoint. Điều này tương đương với việc nhấp đúp vào lề trái của dòng.



Hình 5. Một điểm ngắt được đặt

Để xóa điểm ngắt, hãy chuyển đổi điểm ngắt một lần nữa. Chấm màu xanh lam ở phía trước dòng sẽ biến mất (Hình 6).

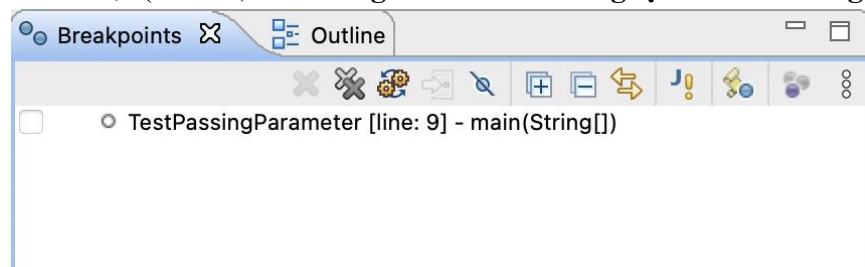

```

1 public class TestPassingParameter {
2
3
4 public static void main(String[] args) {
5     // TODO Auto-generated method stub
6     DigitalVideoDisc jungleDVD = new DigitalVideoDisc("Jungle");
7     DigitalVideoDisc cinderellaDVD = new DigitalVideoDisc("Cinderella");
8
9     swap(jungleDVD, cinderellaDVD);
10    System.out.println("jungle dvd title: "+jungleDVD.getTitle());
11    System.out.println("cinderella dvd title: "+cinderellaDVD.getTitle());
12
13    changeTitle(jungleDVD, cinderellaDVD.getTitle());
14    System.out.println("jungle dvd title: "+jungleDVD.getTitle());
15 }
16
17 public static void swap(Object o1, Object o2) {
18     Object tmp = o1;
19     o1 = o2;
20     o2 = tmp;
21 }
22
23 public static void changeTitle(DigitalVideoDisc dvd, String title) {
24     String oldTitle = dvd.getTitle();
25     dvd.setTitle(title);
26     dvd = new DigitalVideoDisc(oldTitle);
27 }
28

```

Hình 6. Điểm ngắt quãng bị xóa

Để hủy kích hoạt điểm ngắt, hãy điều hướng đến Chế độ xem điểm ngắt và bỏ chọn dấu kiểm bên cạnh điểm ngắt bạn muốn hủy kích hoạt (Hình 7). **Chương trình sẽ chỉ dừng lại ở các điểm ngắt được kích hoạt.**

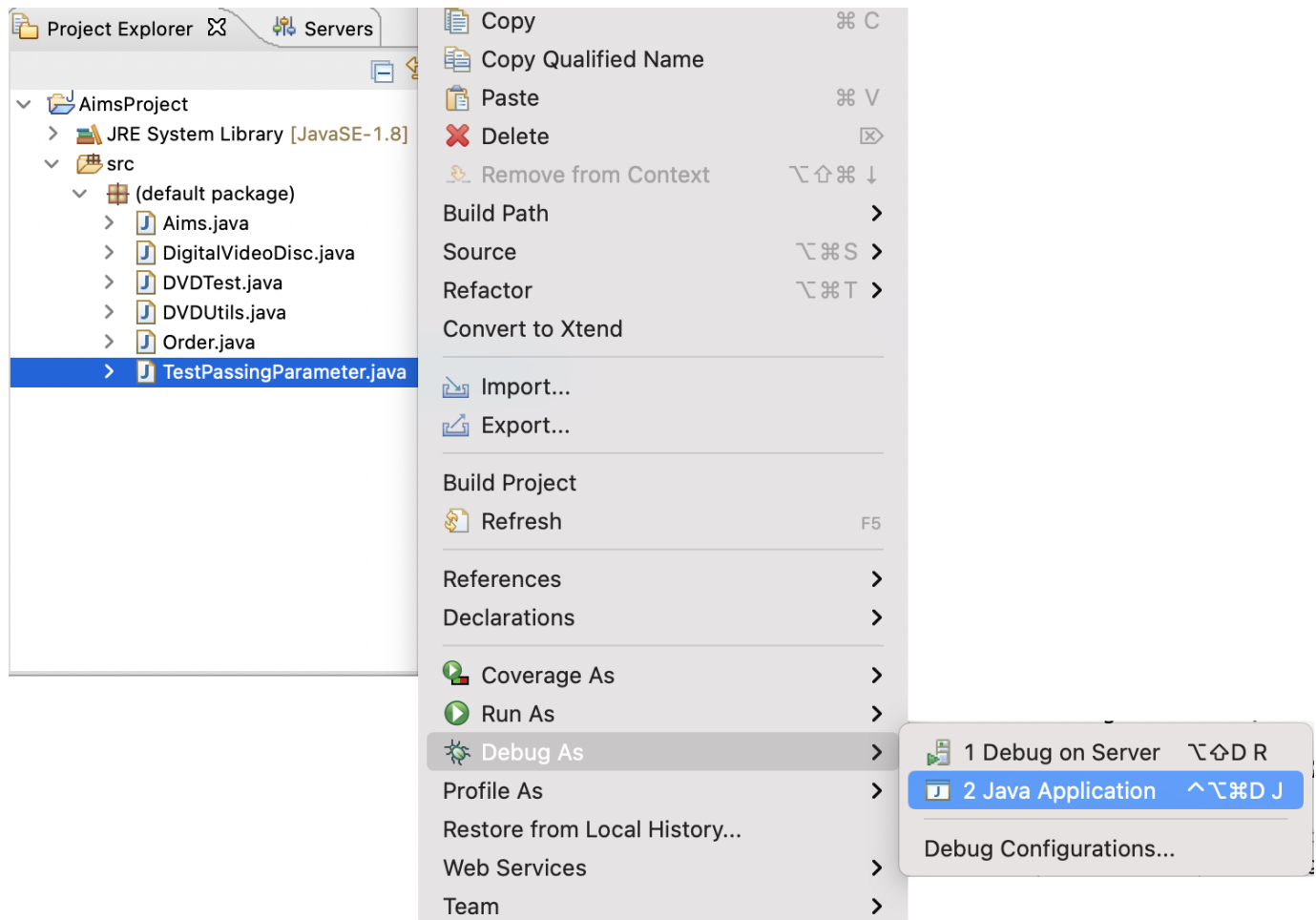


Hình 7. Điểm dừng đã hủy kích hoạt trong Chế độ xem điểm ngắt

Đối với ví dụ này, chúng ta sẽ cần giữ điểm ngắt này, vì vậy hãy đảm bảo đặt lại điểm ngắt sau khi thực hành xóa / hủy kích hoạt nó trước khi chuyển sang phần tiếp theo.

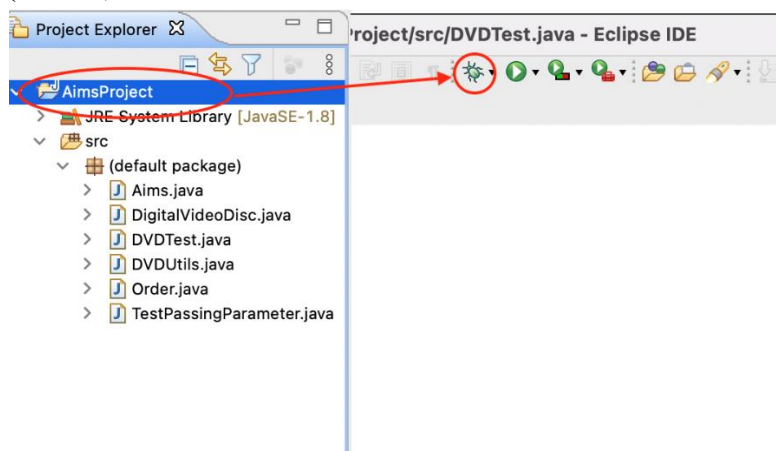
4.2.2. Chạy ở chế độ Debug:

Chọn một tệp Java với phương pháp chính có chứa mã mà bạn cần gỡ lỗi từ Project Explorer. Trong ví dụ này, chúng tôi chọn tệp **TestPassingParameter.java**. Nhấp chuột phải và chọn **Debug As > Java Application** (Hình 8).



Hình 8. Chạy gỡ lỗi từ một lớp

Ngoài ra, các bạn có thể chọn nút gốc của project trong Project Explorer và nhấn vào biểu tượng debug trên thanh công cụ Eclipse (Hình 9)

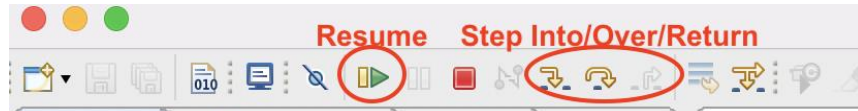


Hình 9. Chạy gỡ lỗi từ một dự án

Ứng dụng bây giờ sẽ được khởi động với Eclipse được đính kèm dưới dạng trình gỡ lỗi. Xác nhận để mở Phối cảnh gỡ lỗi.

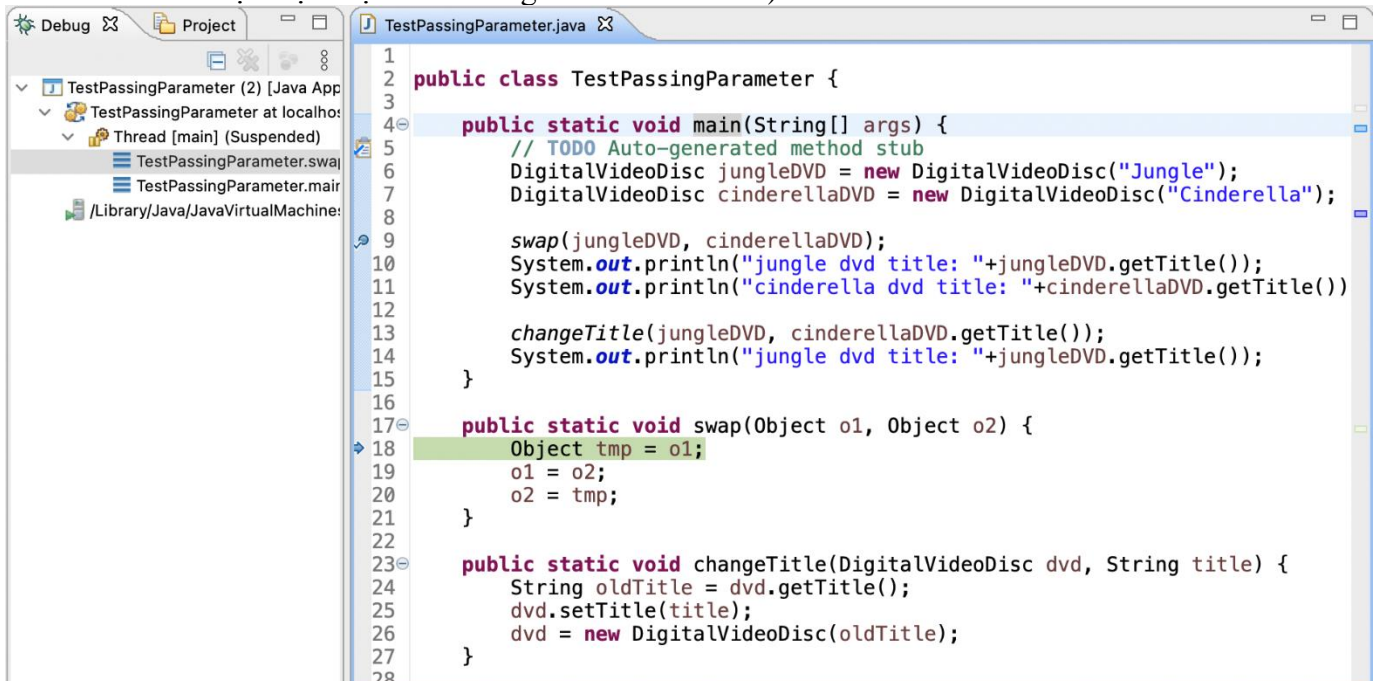
4.2.3. Bước vào, bước qua, bước trở lại, tiếp tục:

-Hình 10.



Hình 10. Bước lệnh trên thanh công cụ trong phối cảnh gỡ lỗi

- Với các tùy chọn trình gỡ lỗi, sự khác biệt giữa "Step into" và "Step over" chỉ đáng chú ý nếu bạn gặp phải một lệnh gọi hàm:
 - o "Bước vào" (F5) có nghĩa là trình gỡ lỗi bước vào chức năng
 - o "Step over" (F6) chỉ cần di chuyển trình gỡ lỗi đến dòng tiếp theo trong cùng một hành động Java
- Với "Step Return" (nhấn F7), bạn có thể hướng dẫn trình gỡ lỗi rời khỏi chức năng; điều này về cơ bản ngược lại với "Bước vào".
- Nhấp vào "Tiếp tục" (F8) hướng dẫn trình gỡ lỗi tiếp tục cho đến khi nó đạt đến điểm ngắt khác. Đối với ví dụ này, chúng ta cần xem việc thực thi hàm **hoán đổi**, vì vậy chúng ta chọn Step Into. Trình gỡ lỗi sẽ bước vào việc thực hiện chức năng **hoán đổi** (Hình 11).



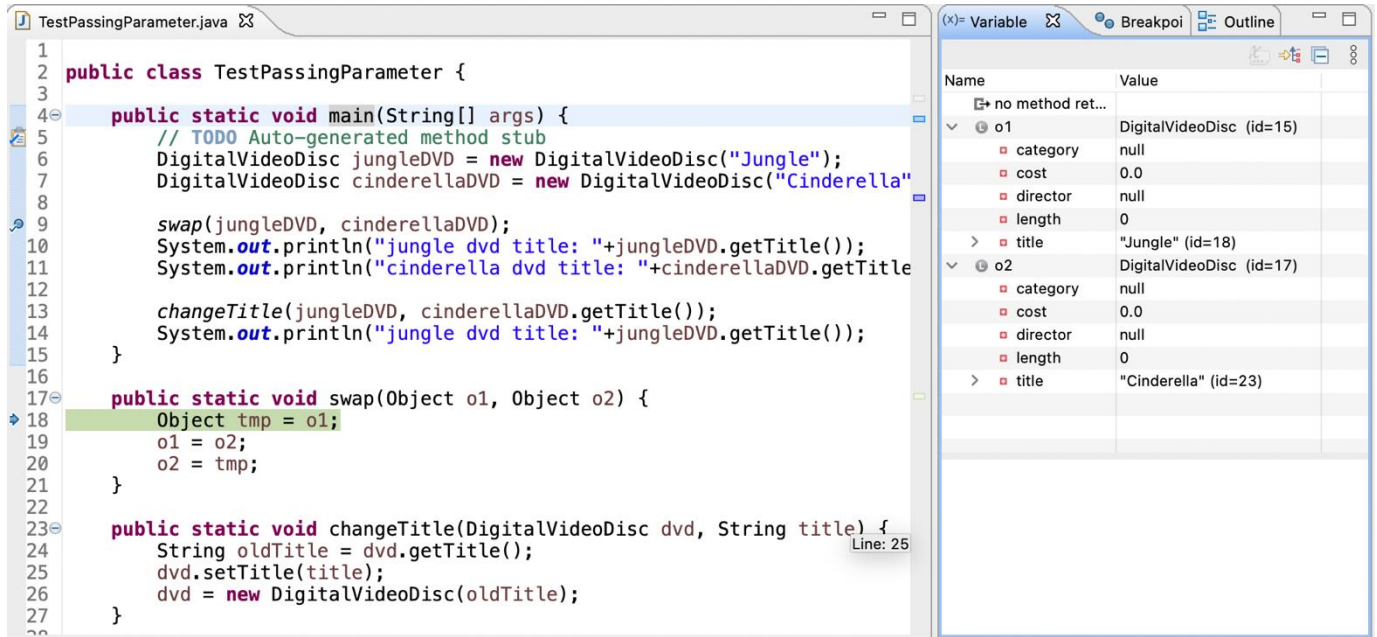
Hình 11. Bước vào chức năng hoán đổi

4.2.4. Điều tra giá trị của các biến:

Chúng ta có thể quan sát giá trị của các biến & biểu thức trong Variable / Expression View. Bạn cũng có thể thêm đồng hồ vĩnh viễn vào biểu thức/biến, sau đó sẽ được hiển thị trong chế độ xem Biểu thức khi tính năng gỡ lỗi được bật.

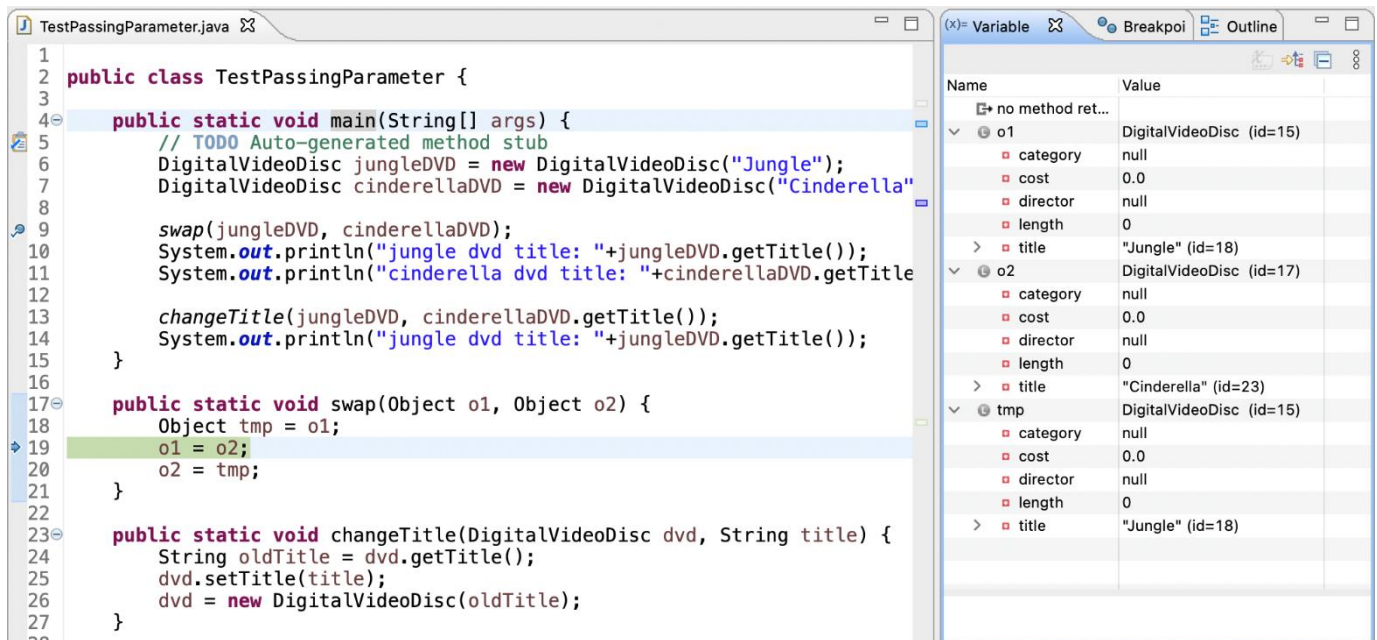
Ngoài ra, hãy buộc con trỏ của bạn vào bất kỳ biến nào trong hành động Java để xem giá trị của nó trong cửa sổ bật lên.

Mở Góc nhìn biến đổi và quan sát các giá trị của biến **o1** & **o2** (Hình 12). Bạn có thể nhấp vào mũi tên thả xuống để điều tra các thuộc tính của biến.

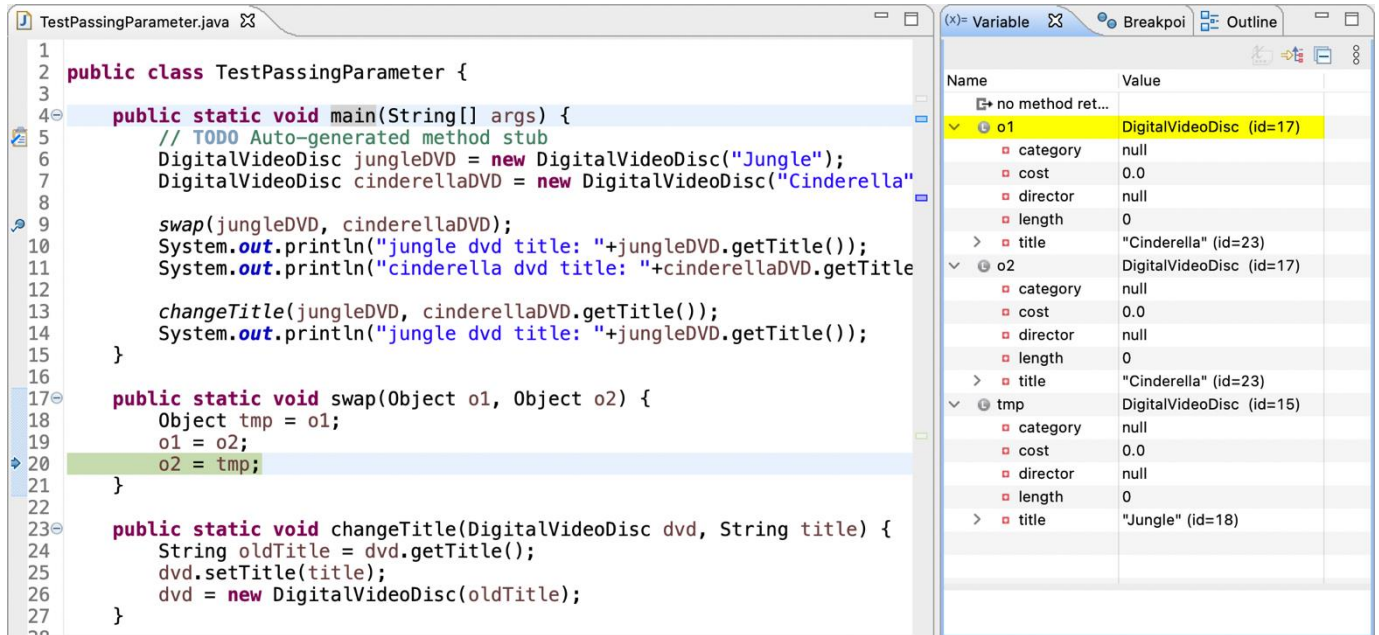


Hình 12. Các biến được hiển thị trong Chế độ xem biến

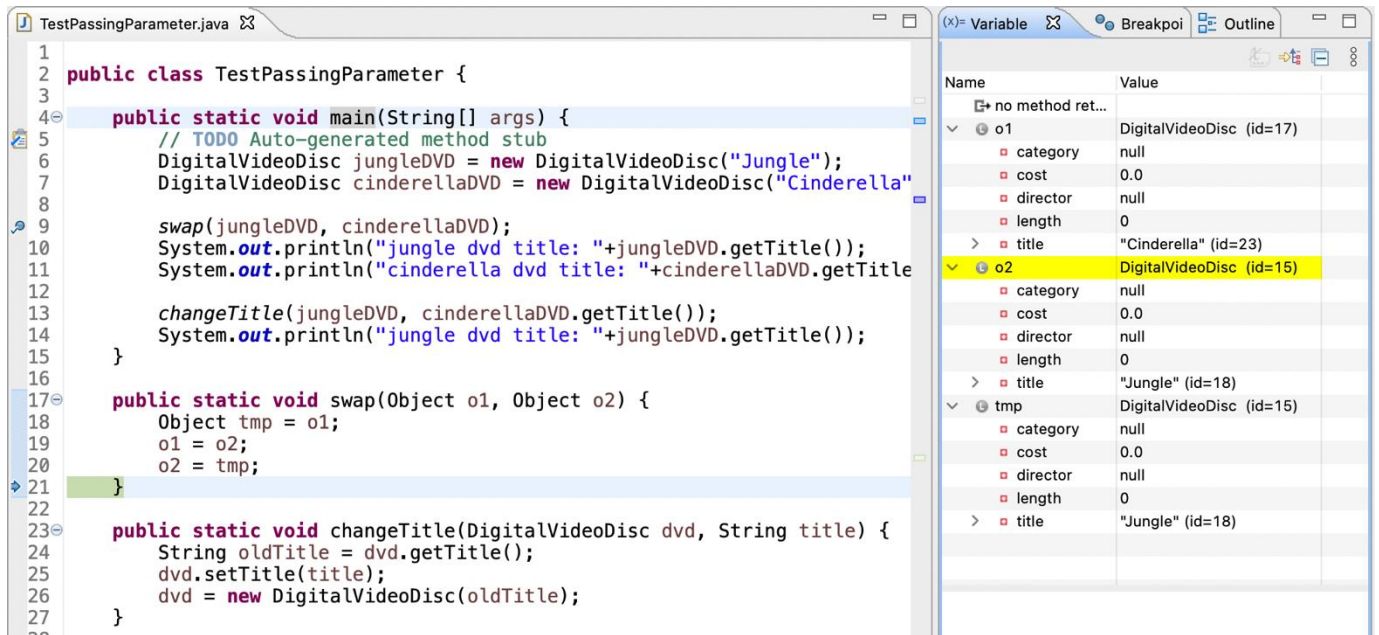
Nhấp vào Step Over và xem sự thay đổi giá trị của các biến **o1**, **o2** & **tmp**. Lặp lại điều này cho đến khi kết thúc hàm **hoán đổi** (Hình 13, Hình 14, Hình 15)



Hình 13. Bước qua dòng 18 của chức năng hoán đổi



Hình 14. Bước qua dòng 19 của chức năng hoán đổi

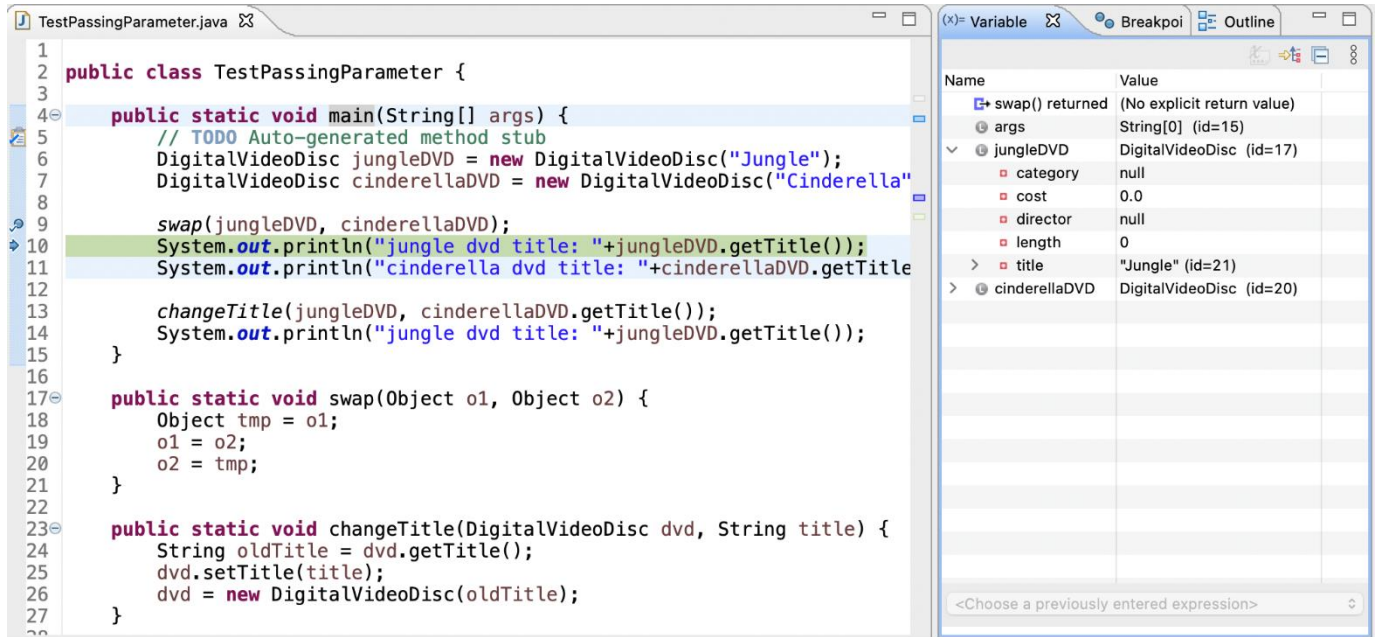


Hình 15. Bước qua dòng 20 của chức năng hoán đổi

4.2.4. Thay đổi giá trị của biến:

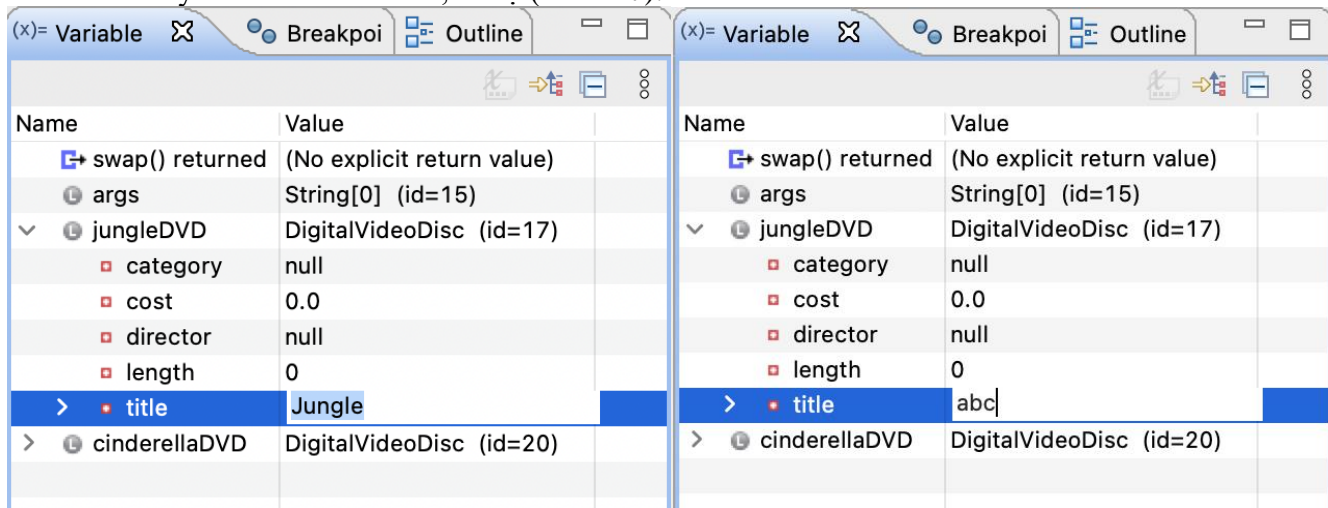
Trong Phối cảnh biến, bạn cũng có thể thay đổi giá trị của biến trong khi gỡ lỗi.

Nhấp vào Step Return để trình gỡ lỗi trở về từ hàm **hoán đổi** trở lại dòng sau khi gọi đến nó. (Hình 16)



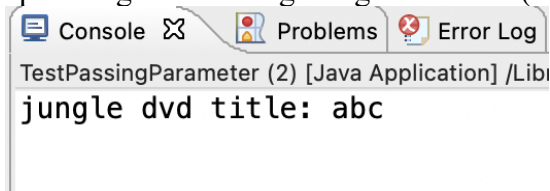
Hình 16. Bước trở lại chức năng chính

Biến **jungleDVD** vẫn có thuộc tính title với giá trị "Jungle". Bạn có thể thay đổi giá trị này bằng cách nhấp vào nó và thay đổi nó thành "abc", ví dụ (Hình 17).



Hình 17. Thay đổi tiêu đề của jungleDVD

Nhấp vào Step Over và xem kết quả trong đầu ra trong Bảng điều khiển (Biểu đồ 18)



Biểu đồ 18. Kết quả(2)

5. Thành viên trình phân loại và Thành viên phiên bản

- Trình phân loại/Thành viên lớp:
 - Được định nghĩa trong một lớp mà một bản sao tồn tại bất kể có bao nhiêu trường hợp của lớp tồn tại.
 - Mục tiêu: có các biến chung cho tất cả các đối tượng

- Bất kỳ đối tượng nào của lớp đều có thể thay đổi giá trị của một biến lớp; Đó là lý do tại sao bạn nên luôn cẩn thận với tác dụng phụ của thành viên trong lớp
- Các biến lớp có thể được thao tác mà không cần tạo một thể hiện của lớp
- Thành viên Phiên bản/Đối tượng:
 - Chỉ liên kết với các đối tượng
 - Được định nghĩa bên trong lớp nhưng bên ngoài bất kỳ phương thức nào
 - Chỉ khởi tạo khi phiên bản được tạo
 - Giá trị của chúng là duy nhất cho mỗi trường hợp của một lớp
 - Sống miễn là đối tượng sống

Mở lớp DigitalVideoDisc:

- Bạn cần lưu ý rằng lớp này chỉ có các biến instance: **title**, **category**, **director**, **length**, **cost**.
- Bây giờ, chúng ta biết rằng mỗi DVD có một id duy nhất được gán bởi hệ thống. Một cách đơn giản để quản lý tất cả các id là cung cấp chúng cho các đĩa DVD mới dưới dạng các giá trị tăng liên tiếp. Để làm điều này, chúng ta phải theo dõi số lượng DVD được tạo.
- Tạo một thuộc tính lớp có tên "**nbDigitalVideoDiscs**" trong lớp **DigitalVideoDisc**
- Tạo một thuộc tính instance có tên "**id**" trong class **DigitalVideoDisc**

```

tính riêng int nbDigitalVideoDiscs = 0;

```

- Mỗi khi một phiên bản của lớp DigitalVideoDisc được tạo, các **nbDigitalVideoDiscs** sẽ được cập nhật. Do đó, bạn nên cập nhật giá trị cho biến lớp này bên trong phương thức constructor và gán giá trị thích hợp cho id.

6. Mở lớp Cart

Viết các phương thức mới để thực hiện các chức năng sau:

- Tạo một phương pháp mới để in danh sách các mặt hàng đã đặt hàng của một giỏ hàng, giá của từng mặt hàng và tổng giá cả. Định dạng dàn ý như sau:

```

XE*****

```

Các mặt hàng đã đặt hàng:

1. DVD - [Tiêu đề] - [thể loại] - [Đạo diễn] - [Độ dài]: [Giá] \$

2. DVD - [Tiêu đề] - ...

Tổng chi phí: [tổng chi phí]

```

*****

```

Gợi ý: Viết một phương thức **toString()** cho lớp **DigitalVideoDisc**. Loại trả về của phương thức này là gì?

- Tìm kiếm DVD trong giỏ hàng theo ID và hiển thị kết quả tìm kiếm. Đảm bảo thông báo cho người dùng nếu không tìm thấy kết quả phù hợp.

- Search cho DVD trong giỏ hàng theo tiêu đề và in kết quả. Đảm bảo thông báo cho người dùng nếu không tìm thấy kết quả phù hợp. *Tham khảo câu lệnh sự cố trong Lab02 để biết quy tắc phù hợp.* **Gợi ý:** viết một **boolean isMatch (Tiêu đề chuỗi)** phương thức i n **DigitalVideoDisc** để tìm hiểu xem đĩa tương ứng có khớp với tiêu đề hay không.

- Trong lớp CartTest, hãy viết mã để kiểm tra tất cả các phương pháp bạn đã viết trong bài tập này. Bạn nên tạo DVD và giỏ hàng mẫu, như trong đoạn mã này:


```

public class CartTest {
    public static void main(String[] args) {
        //Create a new cart
        Cart cart = new Cart();

        //Create new dvd objects and add them to the 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);

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

        //Test the print method
        cart.print();
        //To-do: Test the search methods here
    }
}

```

Biểu đồ 19. Đoạn mã cho CartTest

7. Triển khai lớp Store

- Tạo một lớp Store, chứa one attribute **itemsInStore[]** – một mảng DVD có sẵn trong cửa hàng.
- Để thêm và xóa DVD khỏi cửa hàng, hãy thực hiện hai phương pháp được gọi là **addDVD** và **removeDVD**
- Kiểm tra hai phương pháp này trong lớp **Store Test**.

8. Tổ chức lại các dự án của bạn

- Đổi tên dự án, sử dụng các gói và tổ chức lại tất cả các phòng thí nghiệm và bài tập thực hành từ Lab01 đến nay.
- + Để đổi tên hoặc di chuyển một mục (tức là một dự án, một lớp, một biến...), nhấp chuột phải vào mục, chọn Refactor -> Rename/Move và làm theo các bước.

| | | | |
|------------------|---------------|----------------------|-------------|
| Refactor | Alt+Shift+T > | Rename... | Alt+Shift+R |
| Convert to Xtend | | Move... | Alt+Shift+V |
| Import... | | Extract Interface... | |

Biểu đồ 20. Tái cấu trúc

- + Để tạo package, click chuột phải vào project (hoặc vào menu File) và chọn New -> Package. Nhập đường dẫn đầy đủ của gói bao gồm các gói chính, được phân tách bằng dấu chấm.
- Giữ file văn bản để trả lời các câu hỏi trong phòng thí nghiệm, các thư mục **"Requirement"** & **"Design"** nên được di chuyển bên trong thư mục gốc của **AimsProject**, bên cạnh thư mục **src/** và **bin/**.

- **Cấu trúc phòng thí nghiệm của bạn** ít nhất phải như dưới đây. Bạn có thể tạo các gói phụ để tổ chức các lớp học của mình một cách hiệu quả hơn trong cả dự án và tất cả các gói danh sách. Tất cả các excercises của lab01 nên được đặt trong gói tương ứng của một dự án - dự án OtherProjects.

+ **Dự án AimsProject**

ho.soict.dsai.aims.disc.DigitalVideoDisc

hust.soict.dsai.aims.cart.Cart

hust.soict.dsai.aims.store.Store

hust.soict.dsai.aims.Aims

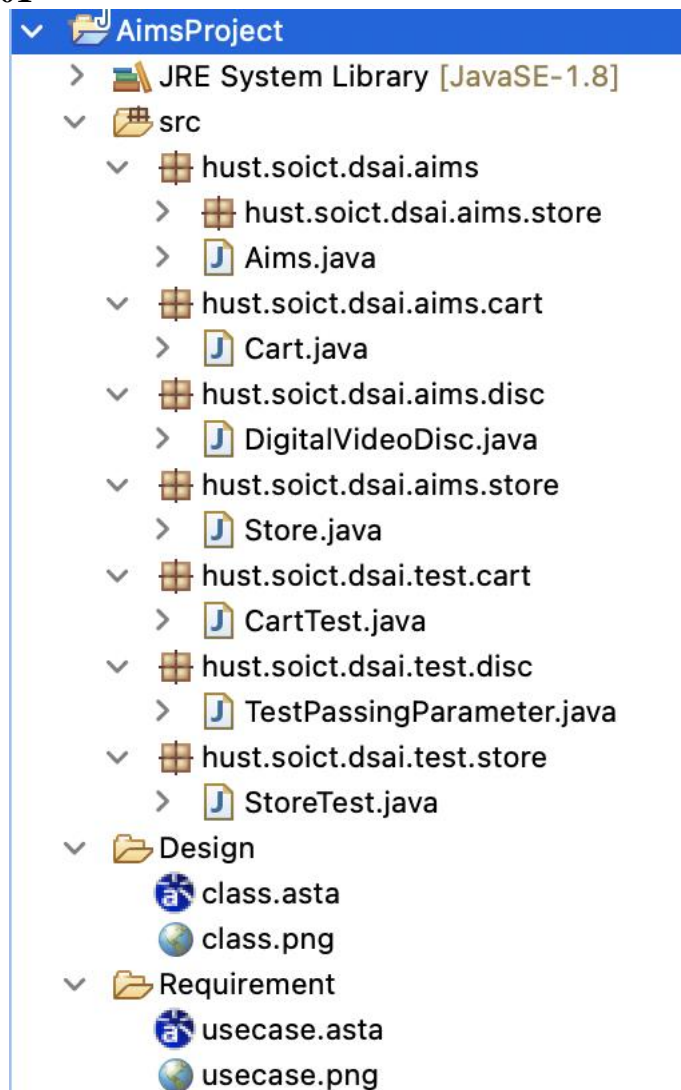
ho.soict.dsai.test.cart.CartTest

hust.soict.dsai.test.store.StoreTest

hust.soict.dsai.test.disc.TestPassingParameter

+ **Các dự án khác**

ho.soict.dsai.lab01



Biểu đồ 21. Cấu trúc được đề xuất cho DS-AI

9. String, StringBuilder và StringBuffer

- Trong dự án **OtherProjects**, tạo một package mới `hust.soict.globalict.garbage` cho ICT hoặc **hust.soict.dsai.garbage** cho DS-AI. Chúng tôi làm việc với gói này trong bài tập này.
- Tạo một lớp mới **ConcatenationInLoops** để kiểm tra thời gian xử lý để xây dựng String sử dụng toán tử `+`, **StringBuffer** và **StringBuilder**.

```
1 public class ConcatenationInLoops {
2     public static void main(String[] args) {
3         Random r = new Random(123);
4         long start = System.currentTimeMillis();
5         String s = "";
6         for (int i = 0; i < 65536; i++)
7             s += r.nextInt(2);
8         System.out.println(System.currentTimeMillis() - start); // This prints roughly 4500.
9
10        r = new Random(123);
11        start = System.currentTimeMillis();
12        StringBuilder sb = new StringBuilder();
13        for (int i = 0; i < 65536; i++)
14            sb.append(r.nextInt(2));
15        s = sb.toString();
16        System.out.println(System.currentTimeMillis() - start); // This prints 5.
17    }
18 }
```

Biểu đồ 22. ConcatenationInLoops

Để biết thêm thông tin về **nối chuỗi**, vui lòng tham khảo <https://redfin.engineering/java-string-concatenation-which-way-is-best-8f590a7d22a8>.

- Tạo một lớp mới **GarbageCreator**. Tạo "rác" càng nhiều càng tốt và quan sát khi bạn chạy một chương trình (nó sẽ để chương trình bị treo hoặc thậm chí ngừng hoạt động khi quá nhiều "rác"). Viết một lớp **NoGarbage** khác để giải quyết vấn đề.

Một số gợi ý:

- Đọc một tập tin văn bản/nhị phân vào một Chuỗi mà không cần sử dụng **StringBuffer** để nối Chuỗi (chỉ sử dụng **toán tử +**). Quan sát và chụp màn hình của bạn khi bạn chọn một tệp rất dài
- Cải thiện code sử dụng **StringBuffer**.

Đoạn mã sau đây là một gợi ý cho việc triển khai của bạn

```

8      String filename = "test.exe"; // test.exe is the name or path to an executable file
9      byte[] inputBytes = { 0 };
10     long startTime, endTime;
11
12     inputBytes = Files.readAllBytes(Paths.get(filename));
13     startTime = System.currentTimeMillis();
14     String outputString = "";
15     for (byte b : inputBytes) {
16         outputString += (char)b;
17     }
18     endTime = System.currentTimeMillis();
19     System.out.println(endTime - startTime);

```

Biểu đồ 23. Mã mẫu cho GarbageCreator

Thay đổi code ở dòng 14-17 ở trên để sử dụng `StringBuffer` thay vì toán tử "+" để build string và quan sát kết quả

```

14     StringBuilder outputStringBuilder = new StringBuilder();
15     for (byte b : inputBytes) {
16         outputStringBuilder.append((char)b);
17     }

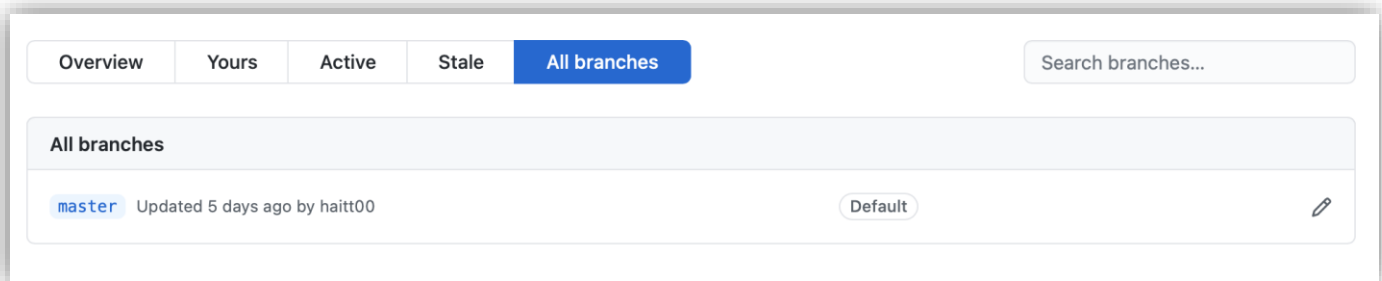
```

Biểu đồ 24. Mã mới sử dụng StringBuffer

10. Trình diễn quy trình phát hành

10.1. Giả thuyết

Biểu đồ 25 cho thấy các nhánh của kho lưu trữ từ xa hiện tại của chúng tôi.



Biểu đồ 25. Các nhánh của Remote Repository

Bây giờ chúng ta thêm một topic mới hoặc một feature mới vào ứng dụng của chúng ta. Phần tiếp theo hướng dẫn chúng ta cách áp dụng Release Flow trong giả thuyết này.

10.2. Trình diễn

Bước 1. Cập nhật kho lưu trữ cục bộ.

Đưa ra lệnh sau và giải quyết xung đột nếu có.

(chủ nhân) \$ git kéo

Bước 2. Tạo và chuyển sang một nhánh mới trong kho lưu trữ cục bộ.

(chủ nhân) \$ git checkout -b feature/demonstrate-release-flow

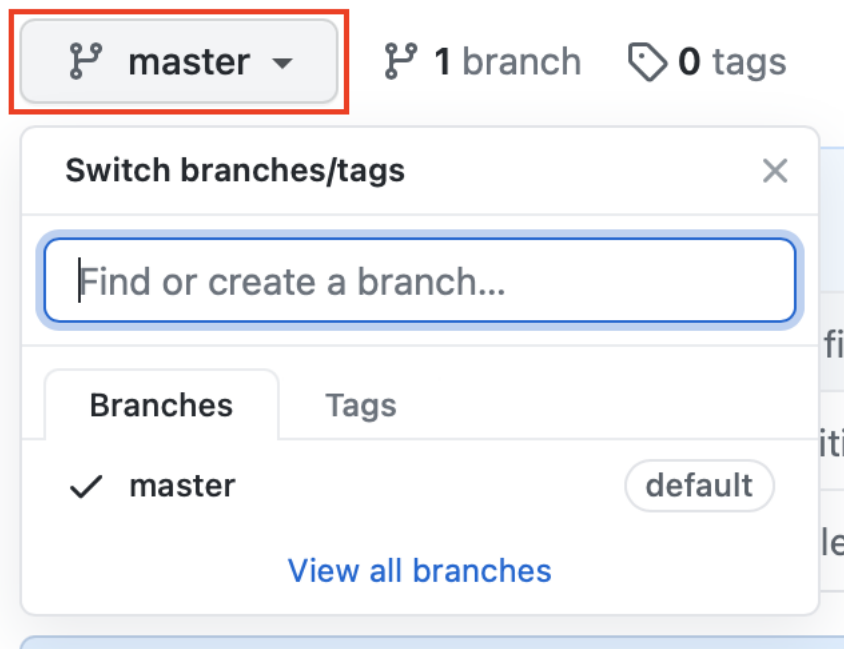
Bước 3. Thực hiện sửa đổi trong kho lưu trữ cục bộ.

Bước 4. Cam kết thay đổi trong kho lưu trữ cục bộ.

(tính năng/trình diễn-phát hành-luồng) \$ git cam kết -m "Thêm tính năng để trình diễn"

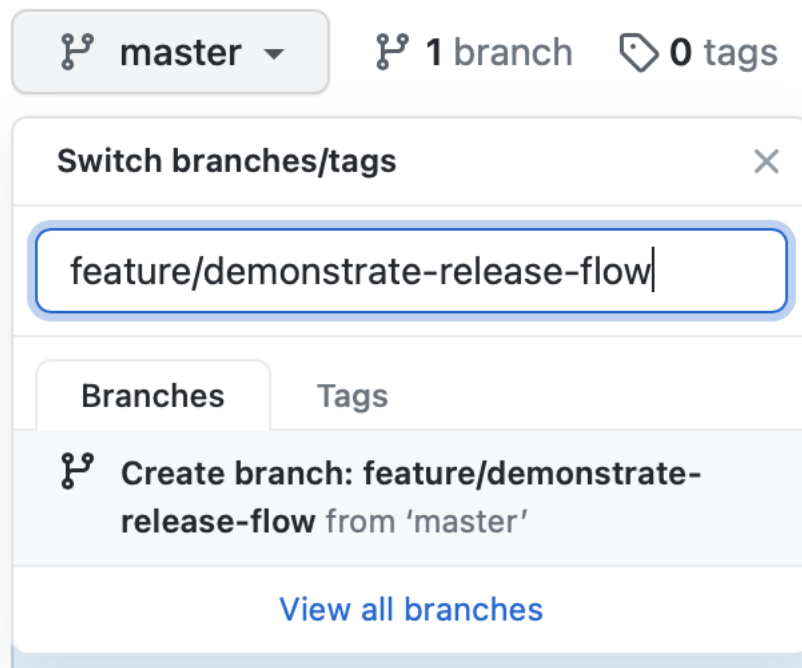
Bước 5. Tạo một branch mới trong remote repository (GitHub thông qua GUI).

- Đầu tiên, trong tab "Mã" của thanh điều hướng trên cùng, chọn nút thả xuống có tên nhánh (trong trường hợp này là "chính") ở trên cùng bên trái.



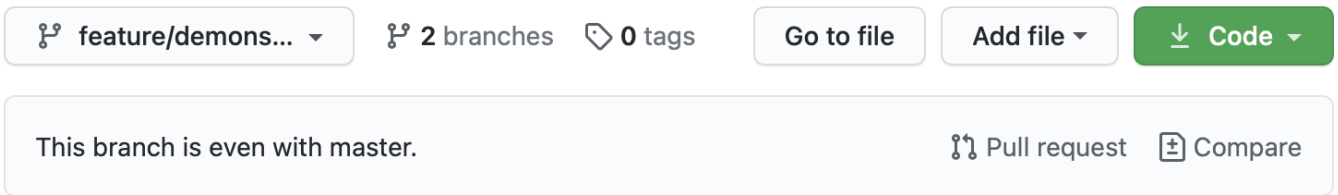
Biểu đồ 26. Tạo chi nhánh trong GitHub GUI (1/3)

- Thứ hai, nhập tên nhánh mới "feature/demonstrate-release-flow" vào trường văn bản và nhấp vào "Create branch: feature/demonstrate-release-flow from 'master'".



Biểu đồ 27. Tạo chi nhánh trong GitHub GUI (2/3)

- Hình dưới đây cho thấy kết quả của những nỗ lực của chúng tôi trong bước này.



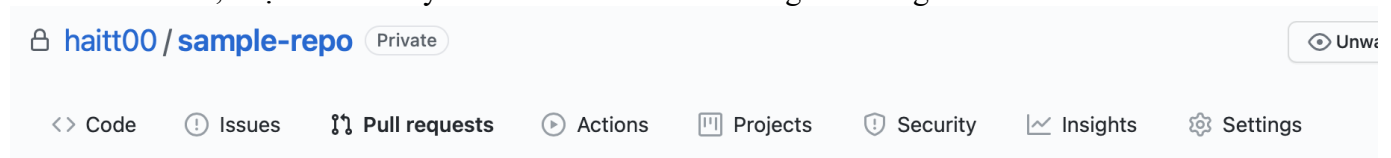
Biểu đồ 28. Tạo chi nhánh trong GitHub GUI (3/3)

Bước 6. Đẩy chi nhánh cục bộ đến chi nhánh từ xa

(tính năng/trình diễn-phát hành-luồng) \$ git tính năng nguồn gốc đẩy / trình diễn-phát hành-dòng chảy

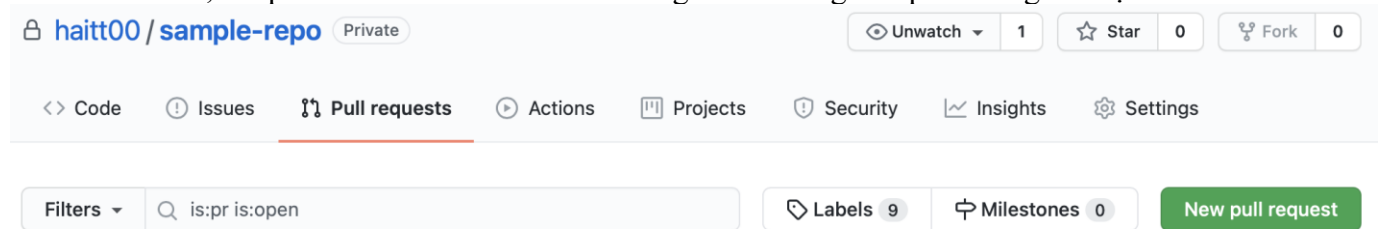
Bước 7. Tạo yêu cầu kéo trong GitHub GUI (chỉ dành cho làm việc trong nhóm)

- Đầu tiên, chọn tab "Kéo yêu cầu" từ thanh điều hướng trên cùng.



Biểu đồ 29. Tạo yêu cầu kéo trong Gui GitHub (1/4)

- Thứ hai, nhấp vào nút "Yêu cầu kéo mới" ở góc trên cùng bên phải của giao diện.

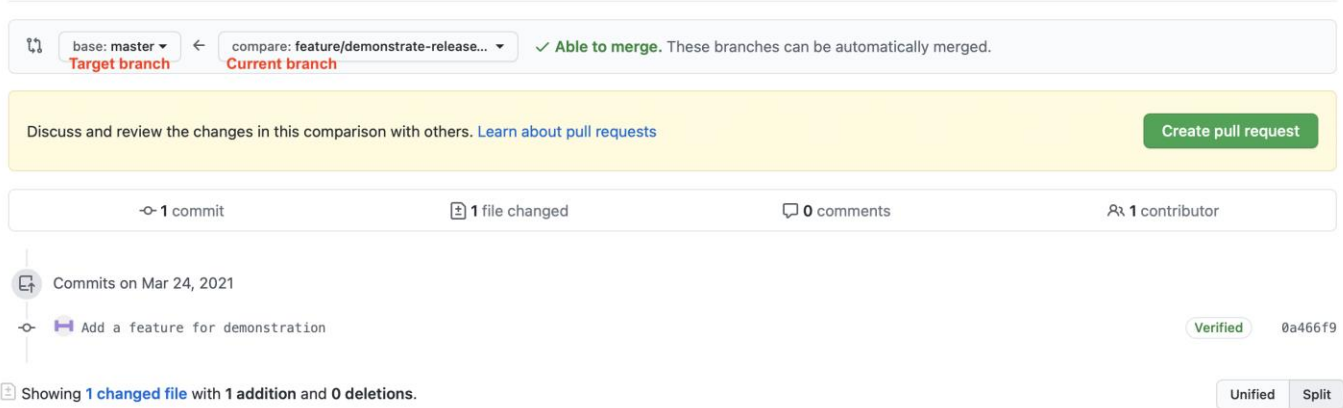


Biểu đồ 30. Tạo yêu cầu kéo trong GitHub GUI (2/4)

- Sau đó, chọn nhánh mục tiêu và nhánh hiện tại. Bên cạnh đó, ở cuối giao diện, chúng ta có thể thấy những thay đổi giữa nhánh hiện tại và nhánh mục tiêu. Chọn "Tạo pull request" ở trên cùng bên phải.
Lưu ý: nhánh đích sẽ ảnh hưởng đến nhánh đích mà chúng ta muốn hợp nhất nhánh của mình trong bước tiếp theo.

Comparing changes

Choose two branches to see what's changed or to start a new pull request. If you need to, you can also [compare across forks](#).

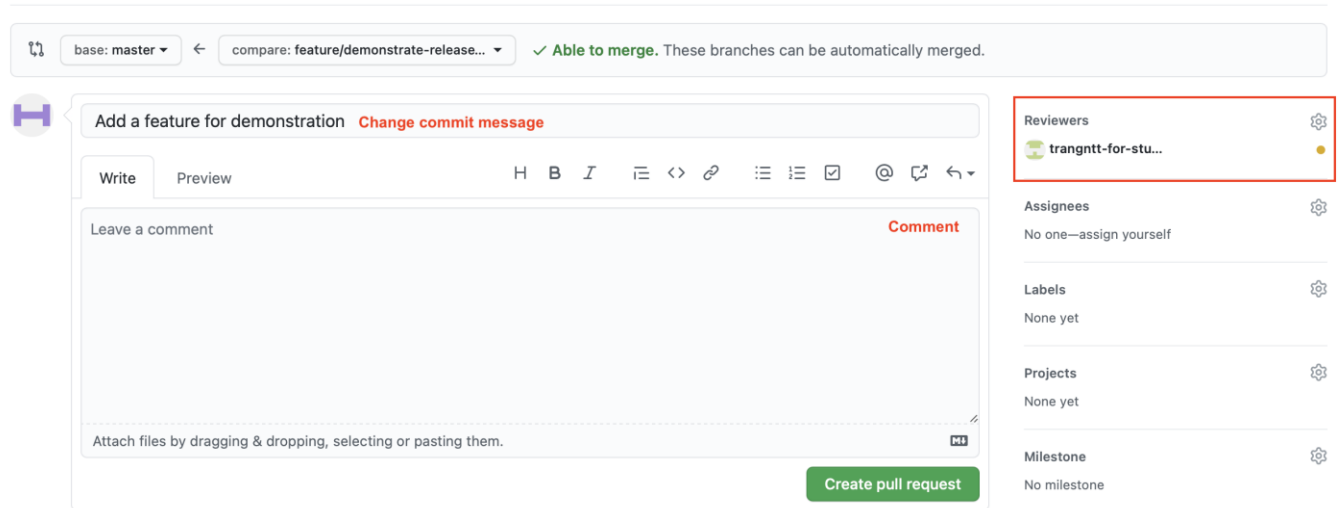


- Biểu đồ 31. Tạo yêu cầu kéo trong GitHub GUI (3/4)

- Cuối cùng, chọn người đánh giá cho yêu cầu kéo. Chúng tôi cũng có thể thay đổi thông điệp cam kết và thêm nhận xét như chúng tôi mong muốn. Chọn "Create pull request"

Open a pull request

Create a new pull request by comparing changes across two branches. If you need to, you can also [compare across forks](#).



- Hình dưới đây cho thấy kết quả của những nỗ lực của chúng tôi trong bảng điều khiển của GitHub. Những người đánh giá được thêm vào cũng có thể thấy các yêu cầu kéo trong trang tổng quan của họ. Khi các thay đổi được xem, chúng ta có thể hợp nhất các nhánh.

Recent activity



Add a feature for demonstration

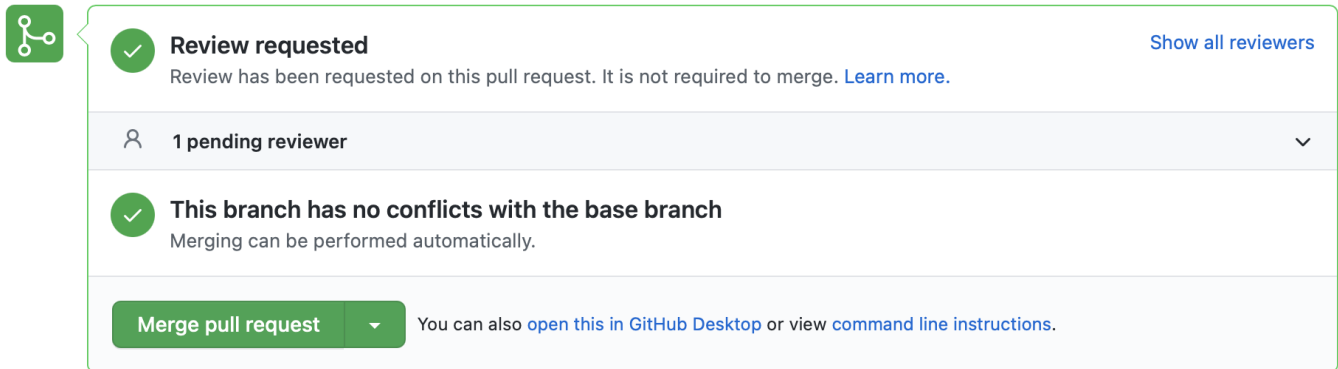


haitt00/sample-repo · Your review was requested 1 minute ago

Biểu đồ 32. Tạo yêu cầu kéo trong GitHub GUI (4/4)

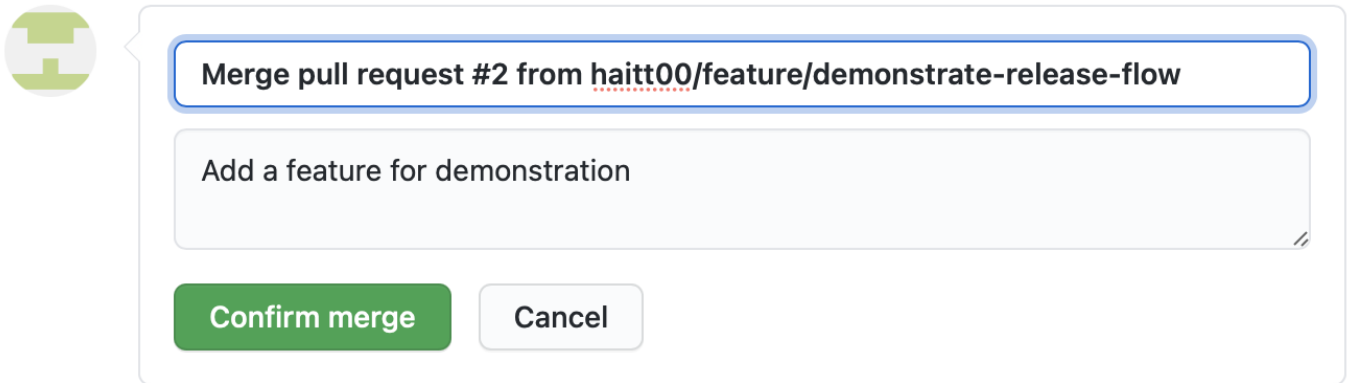
Bước 8. Hợp nhất nhánh từ xa mới vào nhánh chính.

- Mở yêu cầu kéo.
- Chọn "Hợp nhất yêu cầu kéo". Bạn có thể chọn một trong một số tùy chọn hợp nhất từ menu thả xuống



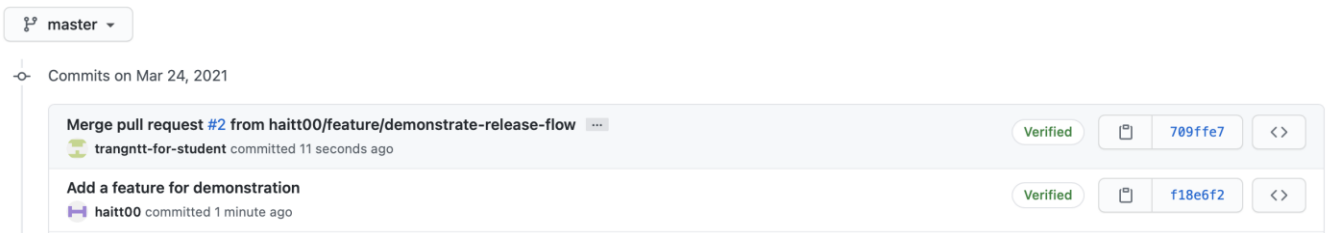
Biểu đồ 33. Sáp nhập chi nhánh (1/3)

- Biểu đồ 33)



Biểu đồ 34. Sáp nhập chi nhánh (2/3)

- Hình dưới đây cho thấy kết quả của những nỗ lực của chúng tôi. Các thay đổi từ nhánh mục tiêu đã được hợp nhất thành nhánh mục tiêu "chính".



Biểu đồ 35. Sáp nhập chi nhánh (3/3)