**READING ASSIGNMENT**

Họ và tên: Trịnh Đức Phương

Lớp: IT3103 – OOP Lab

Ngày: 28 tháng 4 năm 2025

**1. Ưu điểm của Tính Đa Hình (Polymorphism) là gì?**

Linh hoạt: Bạn có thể viết code xử lý chung cho một kiểu cha (ví dụ: Media) và nó sẽ tự động hoạt động đúng với các kiểu con khác nhau (như Book, CD, DVD). Khi bạn thêm một loại Media mới, code cũ thường không cần sửa nhiều.

Dễ mở rộng: Việc thêm các lớp con mới với hành vi riêng trở nên dễ dàng hơn mà ít ảnh hưởng đến các phần khác của chương trình.

Code gọn hơn, dễ đọc hơn: Thay vì phải viết nhiều câu lệnh if-else hoặc switch để kiểm tra xem đối tượng thuộc loại nào rồi mới gọi hành động tương ứng, bạn chỉ cần gọi một phương thức chung (ví dụ: media.getInfo()). Java sẽ tự động gọi đúng phiên bản của phương thức đó cho đối tượng cụ thể tại thời điểm chạy.

**2. Kế thừa (Inheritance) hữu ích như thế nào để đạt được Đa hình (Polymorphism) trong Java?**

Kế thừa là cách chính để tạo ra đa hình (loại đa hình lúc chạy - runtime polymorphism) trong Java:

Tạo quan hệ "IS-A": Khi lớp con extends lớp cha (ví dụ: DVD extends Media), nó tạo ra mối quan hệ "DVD là một Media". Điều này cho phép bạn dùng một biến kiểu Media để lưu trữ một đối tượng DVD. Đây là điều kiện cần để đa hình hoạt động.

Cho phép Ghi đè phương thức (Method Overriding): Lớp con có thể định nghĩa lại phương thức của lớp cha. Ví dụ, DVD có thể có cách play() riêng, khác với CD.

Hoạt động lúc chạy: Khi bạn gọi phương thức qua biến kiểu lớp cha (ví dụ: media.play()), Java sẽ nhìn xem đối tượng thực sự mà biến media đang giữ là gì (là DVD hay CD?) và gọi phương thức play() tương ứng của đối tượng đó. Quá trình này (kết hợp kế thừa, ghi đè và quyết định lúc chạy) chính là cách đa hình được thể hiện.

**3. Sự khác biệt giữa Đa hình (Polymorphism) và Kế thừa (Inheritance) trong Java?**

Kế thừa (Inheritance): Là việc một lớp con "thừa hưởng" các đặc điểm (thuộc tính, phương thức) từ lớp cha. Nó tập trung vào việc tái sử dụng code và tạo ra mối quan hệ phân cấp (cha-con). Giống như con cái thừa hưởng đặc điểm từ bố mẹ.

Đa hình (Polymorphism): Là khả năng một đối tượng thể hiện nhiều "hình dạng" hoặc hành vi khác nhau thông qua một giao diện (lớp cha hoặc interface) chung. Nó tập trung vào tính linh hoạt trong cách các đối tượng hoạt động. Giống như một người có thể đóng nhiều vai trò khác nhau (học sinh, con cái, bạn bè...).

Mối liên hệ: Kế thừa là một cơ chế/cách thức quan trọng để đạt được tính đa hình trong Java. Bạn dùng kế thừa để tạo ra cấu trúc lớp và khả năng ghi đè phương thức, từ đó tính đa hình mới có thể được thể hiện khi chương trình chạy.

**4. So sánh 3 cách nối chuỗi**

***Cấu hình máy chạy:***

JDK: 17

IDE: Visual Studio Code

Máy: Macbook Air M1

***Số vòng lặp thử nghiệm:***

100.000 lần

***Kết quả chạy:***

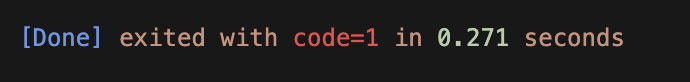
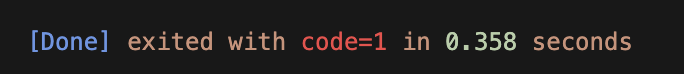
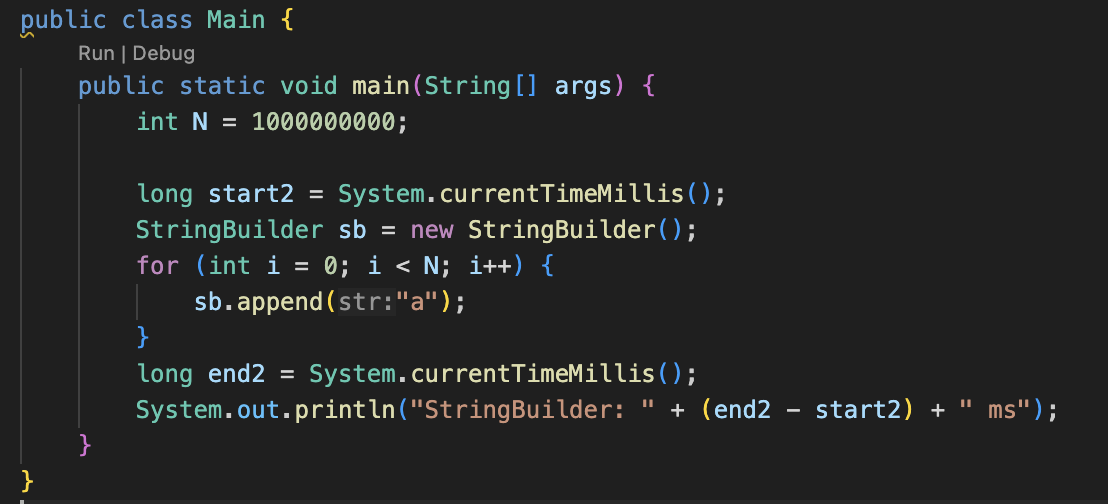
String +: 0.401s (Lần 1); 0.408s (Lần 2); 0.277s (Lần 3)A computer screen shot of a program code

AI-generated content may be incorrect.A black background with red text

AI-generated content may be incorrect.A black background with red text

AI-generated content may be incorrect.

StringBuilder: 0.358s (Lần 1); 0.271s (Lần 2); 0.343s (Lần 3)

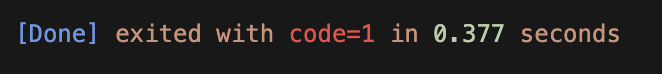
A black background with red text

AI-generated content may be incorrect.

StringBuffer: 0.355s (Lần 1); 0.377s (Lần 2); 0.343s (Lần 3)

A computer screen with colorful text

AI-generated content may be incorrect.A red text on a black background

AI-generated content may be incorrect.

***Nhận xét:***

Cách dùng + trên String (trung bình 0.362s) chậm nhất do tạo ra object mới mỗi lần nối.

StringBuilder (trung bình 0.324s) nhanh nhất vì không đồng bộ, dùng trong môi trường đơn luồng.

StringBuffer hơi chậm hơn (trung bình 0.358s) vì đồng bộ thread-safe, phù hợp khi làm đa luồng.

***Kết luận:***

Trong vòng lặp lớn, nên dùng StringBuilder để nối chuỗi hiệu quả.

Tránh dùng + trong vòng lặp để tránh hiệu suất kém.

**Nguồn tham khảo:**

Oracle Java Tutorials - Polymorphism: <https://docs.oracle.com/javase/tutorial/java/IandI/polymorphism.html>

Oracle Java Tutorials - Inheritance: <https://docs.oracle.com/javase/tutorial/java/IandI/subclasses.html>

GeeksforGeeks - Polymorphism in Java:   
<https://www.geeksforgeeks.org/polymorphism-in-java/>