1. What are the advantages of Polymorphism?
(Ưu điểm của Tính đa hình là gì?)

Tính đa hình là một trong những trụ cột của lập trình hướng đối tượng, mang lại nhiều lợi ích quan trọng:

- **Tính linh hoạt (Flexibility):** Cho phép bạn viết code có thể làm việc với các đối tượng thuộc nhiều kiểu khác nhau thông qua một giao diện (interface) hoặc lớp cha chung. Ví dụ, bạn có thể có một danh sách List<Media> chứa cả Book, DVD, CD và xử lý chúng một cách thống nhất mà không cần biết chính xác kiểu cụ thể của từng đối tượng tại thời điểm biên dịch.

- **Tính mở rộng (Extensibility):** Dễ dàng thêm các lớp con mới (ví dụ: thêm loại BluRay kế thừa từ Disc) mà không cần sửa đổi nhiều hoặc không cần sửa đổi mã nguồn hiện có đang sử dụng các đối tượng thông qua lớp cha hoặc interface chung. Chỉ cần lớp mới tuân thủ "hợp đồng" (ví dụ: implement phương thức play() nếu nó implements Playable), mã cũ vẫn hoạt động tốt.

- **Mã nguồn đơn giản và dễ đọc hơn (Simplicity and Readability):** Thay vì phải dùng nhiều câu lệnh if-else hoặc switch-case để kiểm tra kiểu đối tượng và gọi phương thức tương ứng, bạn chỉ cần gọi phương thức chung trên tham chiếu của lớp cha/interface. Ví dụ, thay vì kiểm tra if (media instanceof DVD) ... else if (media instanceof CD) ..., bạn chỉ cần gọi media.play() (nếu media là Playable), và Java sẽ tự động gọi đúng phương thức play() của đối tượng cụ thể tại thời điểm chạy.

- **Tăng khả năng tái sử dụng mã (Code Reusability):** Mã được viết cho lớp cha hoặc interface có thể được tái sử dụng cho tất cả các lớp con hiện tại và tương lai.

2. How is Inheritance useful to achieve Polymorphism in Java?
(Tính kế thừa hữu ích như thế nào để đạt được Tính đa hình trong Java?)

Tính kế thừa là một cơ chế *nền tảng* để đạt được **Tính đa hình thời gian chạy (Runtime Polymorphism)**, còn được gọi là **Ghi đè phương thức (Method Overriding)**, trong Java:

- **Thiết lập mối quan hệ "IS-A":** Kế thừa tạo ra mối quan hệ phân cấp "là một loại của" (IS-A). Ví dụ, Book IS-A Media, DigitalVideoDisc IS-A Disc, Disc IS-A Media. Điều này cho phép một đối tượng của lớp con được coi như một đối tượng của lớp cha (upcasting). Ví dụ: Media myMedia = new Book(...);.

- **Cho phép Ghi đè phương thức (Method Overriding):** Lớp con có thể cung cấp một triển khai (implementation) cụ thể riêng cho một phương thức đã được định nghĩa ở lớp cha (với cùng tên, cùng tham số, cùng kiểu trả về hoặc kiểu trả về con). Ví dụ,
  lớp Book, CompactDisc, DigitalVideoDisc đều có thể ghi đè phương thức toString() kế thừa từ Object (hoặc từ Media nếu Media ghi đè nó) để cung cấp cách hiển thị thông tin riêng. Tương tự, CompactDisc và DigitalVideoDisc ghi đè (hoặc triển khai từ interface) phương thức play().

- **Liên kết động (Dynamic Binding) / Quyết định tại Runtime:** Khi bạn gọi một phương thức bị ghi đè thông qua một tham chiếu của lớp cha, Java sẽ xác định *tại thời điểm chạy (runtime)* dựa trên kiểu đối tượng *thực tế* mà tham chiếu đang trỏ tới để quyết định phiên bản phương thức nào (của lớp cha hay của lớp con) sẽ được thực thi. Ví dụ:
- Media media1 = new Book("Dune");
- Media media2 = new CompactDisc("Thriller", ...);
- System.out.println(media1.toString()); // Gọi toString() của Book

System.out.println(media2.toString()); // Gọi toString() của CompactDisc

content_copydownload

Use code with caution.Java

Đây chính là biểu hiện của tính đa hình – cùng một lời gọi phương thức (toString()) nhưng thực thi các hành vi khác nhau dựa trên đối tượng thực tế – và nó được hiện thực hóa nhờ cơ chế kế thừa và ghi đè phương thức.

3. What are the differences between Polymorphism and Inheritance in Java?
(Sự khác biệt giữa Tính đa hình và Tính kế thừa trong Java là gì?)

Mặc dù liên quan chặt chẽ (đặc biệt là trong việc đạt được đa hình thời gian chạy), Kế thừa và Đa hình là hai khái niệm riêng biệt:

Đặc điểm	**Tính Kế thừa (Inheritance)**	**Tính Đa hình (Polymorphism)**
Khái niệm cốt lõi	Cơ chế cho phép một lớp (con) thừa hưởng thuộc tính và phương thức từ lớp khác (cha).	Khả năng của một đối tượng/tham chiếu thể hiện nhiều hình thái/hành vi khác nhau.
Mục đích chính	Tái sử dụng mã, tạo cấu trúc phân cấp lớp, định nghĩa mối quan hệ "IS-A".	Cung cấp tính linh hoạt, mở rộng, cho phép xử lý các đối tượng khác nhau một cách thống nhất.
Bản chất	Là một **cơ chế** (mechanism) của ngôn ngữ lập trình.	Là một **nguyên lý/khái niệm** (principle/concept) của OOP.
Mối quan	"IS-A" (Là một loại của). Ví	"Many Forms" (Nhiều hình thái). Một hành động

hệ	dụ: Book IS-A Media.	(phương thức) có nhiều cách thực hiện.
Cách đạt được	Sử dụng từ khóa extends.	- **Runtime:** Thông qua kế thừa và ghi đè phương thức (Method Overriding). - **Compile-time:** Thông qua nạp chồng phương thức (Method Overloading). - Cũng có thể đạt được qua interface.
Ví dụ	Lớp Book kế thừa title, cost từ Media.	Gọi media.toString() sẽ thực thi toString() của Book, CD, hoặc DVD tùy thuộc media là đối tượng nào. Gọi cùng tên phương thức play() trên các đối tượng Playable khác nhau.

Tóm lại:

- **Kế thừa** tập trung vào việc **chia sẻ mã và cấu trúc** giữa các lớp có liên quan (quan hệ cha-con).
- **Đa hình** tập trung vào việc **hành xử khác nhau** của các đối tượng thông qua một giao diện chung, mang lại sự linh hoạt và khả năng mở rộng cho hệ thống. Kế thừa là một trong những cách chính để hiện thực hóa tính đa hình.