1.What are the advantages of Polymorphism?

Tính đa hình mang lại nhiều lợi ích quan trọng trong lập trình hướng đối tượng:

- **Tính linh hoạt và khả năng mở rộng (Flexibility and Extensibility):** Cho phép dễ dàng thêm các lớp con mới mà không cần thay đổi nhiều mã nguồn đã tồn tại.
- **Khả năng tái sử dụng mã (Code Reusability):** Cho phép viết mã chung có thể hoạt động trên nhiều loại đối tượng khác nhau thông qua lớp cha hoặc interface chung của chúng.
- **Giảm sự phụ thuộc (Reduced Coupling):** Các lớp không cần biết cụ thể về kiểu dữ liệu chính xác của các đối tượng mà chúng tương tác.
- **Tính trừu tượng hóa (Abstraction):** Ẩn đi sự phức tạp của các kiểu dữ liệu cụ thể và tập trung vào hành vi chung.
- **Dễ bảo trì và phát triển (Maintainability and Development):** Mã đa hình thường dễ đọc, dễ hiểu và dễ bảo trì hơn do tính nhất quán trong cách xử lý các đối tượng. Việc thêm chức năng mới thường chỉ yêu cầu tạo các lớp con mới mà không ảnh hưởng đến mã đã có.

2. How is Inheritance useful to achieve Polymorphism in Java?

Tính kế thừa đóng một vai trò then chốt trong việc đạt được tính đa hình ở Java, đặc biệt là đa hình thời gian chạy (method overriding):

- **Tạo ra mối quan hệ "is-a" (Creates "is-a" relationship):** Kế thừa cho phép một lớp con (subclass) kế thừa các thuộc tính và phương thức từ lớp cha (superclass). Điều này tạo ra mối quan hệ "là một" (ví dụ: DigitalVideoDisc là một Disc, Disc là một Media).
- **Cung cấp một kiểu chung (Provides a common type):** Các đối tượng của các lớp con khác nhau có thể được coi như là các đối tượng của lớp cha. Trong ví dụ, một ArrayList<Media> có thể chứa các đối tượng Book, DigitalVideoDisc, và CompactDisc.
- **Cho phép ghi đè phương thức (Enables method overriding):** Lớp con có thể ghi đè (override) các phương thức được kế thừa từ lớp cha để cung cấp một cách triển khai riêng phù hợp với đặc điểm của nó. Chính việc ghi đè này cho phép cùng một lời gọi phương thức (ví dụ: toString()) lại thực hiện các hành động khác nhau tùy thuộc vào kiểu thực tế của đối tượng tại thời điểm chạy.

3.What are the differences between Polymorphism and Inheritance in Java?

Tính đa hình và tính kế thừa là hai khái niệm riêng biệt nhưng có mối quan hệ chặt chẽ trong lập trình hướng đối tượng:

- **Tính kế thừa (Inheritance):** Là cơ chế cho phép một lớp (lớp con) kế thừa các thuộc tính và phương thức từ một lớp khác (lớp cha). Mục đích chính của kế thừa là tái sử dụng mã nguồn và thiết lập mối quan hệ "is-a" giữa các lớp. Nó tập trung vào việc tạo ra một hệ thống phân cấp các lớp.
- **Tính đa hình (Polymorphism):** Là khả năng một đối tượng có thể thể hiện nhiều hình thức khác nhau. Trong Java, điều này thường đạt được thông qua việc sử dụng các tham chiếu kiểu lớp cha để trỏ đến các đối tượng của lớp con, và phương thức được gọi sẽ phụ thuộc vào kiểu thực tế của đối tượng đó tại thời điểm chạy. Mục đích chính của đa hình là tạo ra sự linh hoạt và khả năng mở rộng cho mã nguồn.

Điểm khác biệt chính:

- **Mục đích chính:** Kế thừa tập trung vào việc tái sử dụng mã và xây dựng mối quan hệ giữa các lớp, trong khi đa hình tập trung vào việc cho phép các đối tượng khác nhau phản ứng khác nhau với cùng một hành động (gọi phương thức).
- **Cơ chế hoạt động:** Kế thừa hoạt động ở cấp độ định nghĩa lớp, tạo ra các lớp con dựa trên lớp cha. Đa hình hoạt động ở cấp độ đối tượng và thời gian chạy, quyết định phương thức nào được thực thi dựa trên kiểu thực tế của đối tượng.
- **Mối quan hệ:** Kế thừa thường là tiền đề để đạt được đa hình thời gian chạy. Nếu không có kế thừa và mối quan hệ "is-a", việc một tham chiếu kiểu lớp cha trỏ đến đối tượng lớp con sẽ không có ý nghĩa. Tuy nhiên, đa hình biên dịch (overloading) không nhất thiết yêu cầu kế thừa.