1. What are the advantages of Polymorphism?

Polymorphism (đa hình) mang lại nhiều lợi ích trong lập trình hướng đối tượng:

- **Tính linh hoạt và mở rộng**: Đa hình cho phép xử lý các đối tượng thuộc các lớp khác nhau thông qua một giao diện hoặc lớp cha chung. Ví dụ, trong hệ thống này, ArrayList<Media> có thể chứa DigitalVideoDisc, CompactDisc, và Book, và chúng ta có thể gọi toString() hoặc play() mà không cần quan tâm đến loại cụ thể của đối tượng.
- **Giảm sự dư thừa mã**: Thay vì viết mã riêng cho từng loại đối tượng, ta có thể viết mã chung cho lớp cha hoặc giao diện (Media hoặc Playable), và các lớp con sẽ tự động triển khai chi tiết.
- **Dễ bảo trì và mở rộng**: Khi thêm một loại media mới (ví dụ: Magazine), chỉ cần kế thừa Media và ghi đè các phương thức như toString(), hệ thống sẽ tự động hỗ trợ mà không cần sửa đổi mã hiện có.

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

Kế thừa (inheritance) là nền tảng để đạt được đa hình trong Java:

- **Kế thừa tạo mối quan hệ "is-a"**: Các lớp con (DigitalVideoDisc, CompactDisc, Book) kế thừa từ Media, nên chúng có thể được coi là Media. Điều này cho phép lưu trữ các đối tượng của lớp con trong một biến hoặc danh sách kiểu Media.
- **Ghi đè phương thức (method overriding)**: Kế thừa cho phép các lớp con ghi đè các phương thức trừu tượng hoặc phương thức của lớp cha (như toString(), play()). Khi gọi phương thức trên biến kiểu Media, Java sẽ gọi phiên bản của lớp con tương ứng (đa hình tại runtime).
- **Ví dụ trong hệ thống**:
    - Media là lớp cha trừu tượng với phương thức trừu tượng toString().
    - Các lớp con ghi đè toString() để cung cấp triển khai cụ thể.
    - Khi gọi media.toString() trong ArrayList<Media>, Java sẽ gọi đúng phương thức toString() của DigitalVideoDisc, CompactDisc, hoặc Book.

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

- **Inheritance (Kế thừa)**:
    - Là cơ chế cho phép một lớp con kế thừa các thuộc tính và phương thức từ lớp cha.

    - Tạo mối quan hệ "is-a" (ví dụ: DigitalVideoDisc is-a Disc, Disc is-a Media).
    - Mục đích chính là tái sử dụng mã và thiết lập hệ thống phân cấp lớp.
    - Ví dụ: DigitalVideoDisc kế thừa id, title, category, cost từ Media thông qua Disc.
- **Polymorphism (Đa hình)**:
    - Là khả năng một đối tượng có thể được xử lý dưới dạng lớp cha, nhưng hành vi cụ thể được xác định bởi lớp con tại runtime.
    - Có 2 loại chính trong Java:
        - **Compile-time polymorphism (đa hình tại compile-time)**: Thông qua nạp chồng phương thức (method overloading).
        - **Runtime polymorphism (đa hình tại runtime)**: Thông qua ghi đè phương thức (method overriding), như trong trường hợp toString() ở đây.
    - Mục đích chính là tăng tính linh hoạt và cho phép xử lý các đối tượng khác nhau thông qua một giao diện chung.
    - Ví dụ: Gọi media.toString() trên ArrayList<Media> sẽ gọi đúng phiên bản toString() của lớp con (DigitalVideoDisc, CompactDisc, hoặc Book).
- **Mối quan hệ giữa Kế thừa và Đa hình**:
    - Kế thừa là một cơ chế cấu trúc (tạo mối quan hệ giữa các lớp), còn đa hình là một cơ chế hành vi (cho phép hành vi khác nhau dựa trên đối tượng thực tế).
    - Kế thừa hỗ trợ đa hình bằng cách cho phép các lớp con ghi đè phương thức của lớp cha, từ đó tạo ra hành vi đa dạng khi gọi phương thức trên biến kiểu lớp cha.