1. What are the advantages of Polymorphism?

Lợi ích của tính đa hình (Polymorphism):

- **Tăng tính linh hoạt:** Một đối tượng có thể biểu diễn dưới nhiều hình thức, giúp mã linh hoạt hơn trong xử lý các kiểu khác nhau của đối tượng cha (superclass).
- **Dễ mở rộng:** Khi cần thêm lớp mới (subclass), ta không cần thay đổi nhiều mã đã có; thay vào đó chỉ cần đảm bảo lớp mới ghi đè (override) đúng phương thức cần thiết.
- **Dễ bảo trì:** Việc sửa đổi hoặc mở rộng hành vi của chương trình dễ dàng hơn vì ta có thể thay thế đối tượng mà không cần thay đổi toàn bộ cấu trúc chương trình.
- **Tối ưu hóa tái sử dụng mã:** Các phương thức có thể dùng chung với nhiều lớp con mà không cần viết lại.
- **Hỗ trợ thiết kế hướng đối tượng hiệu quả:** Cho phép lập trình giao diện thay vì lập trình hiện thực cụ thể.

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

Kế thừa giúp đạt được đa hình trong Java như thế nào?

- Trong Java, kế thừa (inheritance) cho phép một lớp con (subclass) **kế thừa các thuộc tính và phương thức từ lớp cha** (superclass).
- Khi một lớp con **ghi đè phương thức (override)** từ lớp cha, ta có thể xử lý các đối tượng lớp con thông qua tham chiếu kiểu lớp cha.
- Từ đó, một phương thức có thể **gọi đến các đối tượng có hành vi khác nhau tùy vào lớp thực tế**, giúp hiện thực đa hình.

Ví dụ:

```
Media m;
m = new Book();          // m là kiểu Media, đối tượng là Book
m = new DigitalVideoDisc(); // m là kiểu Media, đối tượng là DVD
```

Khi gọi m.play() thì mỗi đối tượng sẽ thực hiện hành vi khác nhau tùy theo lớp con.

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

Sự khác biệt giữa Polymorphism và Inheritance trong Java:

Tiêu chí	Inheritance (Kế thừa)	Polymorphism (Đa hình)
Khái niệm	Cơ chế cho phép lớp con kế thừa thuộc tính và phương thức từ lớp cha.	Cơ chế cho phép đối tượng thể hiện nhiều hành vi khác nhau tùy vào lớp cụ thể của nó.
Mục tiêu chính	Tái sử dụng mã nguồn, tổ chức hệ thống phân cấp.	Tăng tính linh hoạt trong xử lý đối tượng.
Quan hệ giữa lớp	Tồn tại quan hệ "is-a" giữa subclass và superclass.	Xảy ra khi có phương thức ghi đè (override) hoặc giao diện (interface).
Yêu cầu về kế thừa	Bắt buộc có quan hệ kế thừa.	Có thể xảy ra thông qua kế thừa hoặc interface.
Ví dụ minh họa	class Book extends Media	Media m = new Book(); m.play(); → hành vi tùy vào đối tượng thực tế là Book.