1. What are the advantages of Polymorphism?

Polymorphism (đa hình) trong Java là một trong những tính chất quan trọng của lập trình hướng đối tượng và mang lại nhiều lợi ích:

- **Tăng tính linh hoạt của mã nguồn:**
  + Đa hình cho phép sử dụng chung một interface hoặc class cha để làm việc với nhiều kiểu đối tượng khác nhau. Nhờ đó, chương trình dễ mở rộng mà không cần sửa đổi mã hiện tại.
- **Giảm sự phụ thuộc giữa các lớp:**
  + Nhờ đa hình, các lớp không cần biết chính xác kiểu đối tượng mà chúng đang làm việc, chỉ cần biết rằng đối tượng đó tuân theo một interface hay kế thừa từ một lớp cha nhất định.
- **Tăng khả năng mở rộng (Extensibility):**
  + Khi muốn thêm một chức năng mới, bạn chỉ cần tạo một lớp mới kế thừa từ lớp cha hoặc implement interface — không cần thay đổi mã hiện có, giảm rủi ro lỗi.
- **Dễ bảo trì và tái sử dụng mã:**
  + Khi logic xử lý được tách biệt thông qua các class con, việc bảo trì trở nên dễ hơn. Có thể sửa lỗi hoặc thay đổi trong từng class riêng biệt mà không ảnh hưởng toàn hệ thống.

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

Kế thừa và đa hình là hai tính chất quan trọng trong lập trình hướng đối tượng, và kế thừa là nền tảng giúp đạt được đa hình trong Java:

- **Cung cấp một lớp cha chung:**
  + Khi một lớp kế thừa từ lớp khác, nó sẽ kế thừa các phương thức và thuộc tính của lớp cha. Điều này tạo ra sự chung nhau giữa các lớp và giúp thực hiện đa hình.
- **Hỗ trợ ghi đè phương thức:**
  + Nhờ kế thừa, các lớp con có thể **ghi đè** phương thức của lớp cha. Điều này cho phép **dispatch động**, tức là phương thức thực tế được gọi sẽ được xác định tại thời gian chạy dựa trên kiểu đối tượng thực tế.
  + Ví dụ:
  Animal myAnimal = new Dog(); // Tham chiếu kiểu Animal, đối tượng kiểu Dog
  myAnimal.makeSound(); // Kết quả: "Woof!" (chứ không phải phương thức của lớp cha)
- **Cho phép tổng quát hóa:**
  + Kế thừa giúp bạn viết mã sử dụng kiểu của lớp cha, và mã này sẽ làm việc với bất kỳ lớp con nào, tạo ra sự linh hoạt và tái sử dụng mã.

+ Ví dụ:

```
public void playAnimalSound(Animal a) {
    a.makeSound();  // Có thể truyền vào Dog, Cat, v.v.
}
```

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

Polymorphism và Kế thừa là hai khái niệm quan trọng trong lập trình hướng đối tượng, và mặc dù chúng liên quan chặt chẽ với nhau, nhưng chúng có những điểm khác biệt rõ rệt trong Java:

	Kế thừa	Đa hình
Khái niệm	Một lớp kế thừa các thuộc tính và phương thức từ lớp khác	Khả năng một phương thức có nhiều hành vi khác nhau
Mục đích	Tái sử dụng và mở rộng chức năng	Tập trung vào hành vi tại runtime
Cách hoạt động	Lớp con kế thừa và có thể ghi đè phương thức	Ghi đè phương thức để thay đổi hành vi tại runtime
Quan hệ	Nền tảng để đa hình hoạt động	Dừa vào kế thừa và ghi đè phương thức