Reading Assignment

1. **What are the advantages of Polymorphism?**
   - Tái sử dụng mã (Code Reusability): có thể viết mã tổng quát hoạt động với nhiều đối tượng khác nhau nếu chúng cùng kế thừa từ một lớp cha hoặc giao diện.
   - Dễ bảo trì (Maintainability): Vì mã được viết một cách tổng quát nên dễ chỉnh sửa, mở rộng hoặc cập nhật mà không phải đụng đến nhiều nơi trong hệ thống.
   - Dễ mở rộng: có thể thêm lớp mới kế thừa lớp cha hoặc cài đặt giao diện mà không cần thay đổi logic cũ.
   - Tính linh hoạt thông qua Interface: Lập trình dựa trên giao diện thay vì cài đặt cụ thể giúp dễ dàng thay thế hoặc mở rộng hệ thống.
   - Ràng buộc động (Dynamic Method Binding): Phương thức được chọn để thực thi được quyết định trong lúc chạy (runtime) dựa vào kiểu thực tế của đối tượng, không phải kiểu tham chiếu.
   - Cải thiện khả năng đọc mã nguồn: Mã sử dụng đa hình thường ngắn gọn
   - và dễ hiểu hơn.
2. **How is Inheritance useful to achieve Polymorphism in Java?**
   Kế thừa cho phép một lớp con kế thừa các thuộc tính và phương thức từ lớp cha, giúp tái sử dụng mã và xây dựng cấu trúc phân cấp rõ ràng.
   - Lớp con có thể ghi đè (override) các phương thức được định nghĩa trong lớp cha (ví dụ như toString()).
   - Khi tham chiếu một đối tượng của lớp con bằng kiểu dữ liệu của lớp cha, Java sẽ sử dụng cơ chế ràng buộc động (dynamic binding) để xác định phương thức cần gọi tại thời điểm chạy. Đây chính là đa hình động (runtime polymorphism)
3. **What are the differences between Polymorphism and Inheritance in Java?**
   - Định nghĩa:
     + Kế thừa (Inheritance): Là cơ chế trong lập trình hướng đối tượng cho phép một lớp con kế thừa các thuộc tính (biến) và hành vi (phương thức) từ một lớp cha. Nhờ đó, lớp con có thể sử dụng lại mã của lớp cha mà không cần viết lại.
     + Đa hình (Polymorphism): Là khả năng một đối tượng có thể mang nhiều hình thức khác nhau. Trong Java, điều này có nghĩa là cùng một phương thức có thể hoạt động khác nhau tùy vào đối tượng cụ thể đang thực hiện nó.
   - Mục đích:
     + Kế thừa: Giúp tái sử dụng mã nguồn, giảm thiểu việc trùng lặp mã, đồng thời giúp xây dựng quan hệ rõ ràng giữa các lớp theo mô hình phân cấp.
     + Đa hình: Giúp tăng tính linh hoạt và mở rộng trong lập trình. Khi sử dụng đa hình, bạn có thể viết mã chung cho lớp cha và để Java tự quyết định hành vi thực tế của lớp con tại thời điểm chạy, nhờ đó dễ bảo trì và mở rộng.

- Thời điểm xảy ra:
  + Kế thừa: Xảy ra tại thời điểm biên dịch (compile time), vì quan hệ kế thừa được xác định khi định nghĩa lớp.
  + Đa hình:
    - Compile time polymorphism: xảy ra khi nạp chồng (method overloading), tức là nhiều phương thức cùng tên nhưng khác tham số.
    - Runtime polymorphism: xảy ra khi ghi đè (method overriding), tức là phương thức ở lớp con thay thế phương thức lớp cha và được gọi dựa trên đối tượng thực tế khi chạy chương trình.
- Mối quan hệ:
  + Kế thừa: Tạo ra quan hệ "is-a" giữa lớp con và lớp cha (ví dụ: Dog là một loại Animal). Nhờ đó, lớp con có thể sử dụng tất cả thành phần không private của lớp cha.
  + Đa hình: Cho phép một phương thức hoặc đối tượng thể hiện hành vi khác nhau tùy vào ngữ cảnh, dù được gọi thông qua kiểu lớp cha hoặc giao diện. Điều này làm cho chương trình dễ mở rộng hơn khi thêm lớp mới.