Câu hỏi 1. What are the advantages of Polymorphism?

Đa hình trong Java mang lại nhiều lợi ích quan trọng trong các dự án

a. Tính linh hoạt và tái sử dụng mã:

- Đa hình cho phép chúng ta sử dụng một kiểu dữ liệu chung (ví dụ: Media) để tham chiếu đến các đối tượng của các lớp con (DigitalVideoDisc, Book, CompactDisc).

- VD: Aims.java (Sec 16), List<Media> để lưu trữ các đối tượng khác nhau và gọi toString() trên chúng mà không cần quan tâm đến kiểu cụ thể của từng đối tượng. Điều này giúp mã trở nên linh hoạt và dễ tái sử dụng.

- VD: Có thể thêm một loại media mới (như Magazine) mà không cần sửa đổi mã duyệt danh sách trong Aims.java.

b. Dễ dàng mở rộng: Khi cần thêm một lớp con mới, ta chỉ cần override các phương thức (như toString(), play()) mà không cần thay đổi logic chính.

c. Giảm độ phức tạp: Đa hình cho phép viết mã tổng quát hơn. Thay vì viết mã riêng để xử lý từng loại media (CD, DVD, Book), ta chỉ cần viết một vòng lặp duy nhất để gọi toString() từ đó giảm độ phức tạp, tăng tính dễ bảo trì của mã.

d. Tăng tính trừu tượng: Đa hình giúp làm việc ở mức trừu tượng cao hơn. Trong Cart và Store,
lưu trữ các đối tượng trong ArrayList<Media> và thao tác trên chúng (thêm, xóa, hiển thị) sẽ không cần biết đối tượng cụ thể là CD, DVD hay Book.

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

a. Tạo mối quan hệ is-a:

- Kế thừa thiết lập mối quan hệ giữa lớp cha và lớp con, cho phép lớp con được xem như một thể hiện

của lớp cha. Trong dự án, DigitalVideoDisc, Book, và CompactDisc kế thừa từ Media (trực tiếp hoặc

gián tiếp qua Disc), có nghĩa là một đối tượng DigitalVideoDisc cũng là một Media, cho phép ta lưu

trữ nó trong List<Media> hoặc biến kiểu Media.

b. Cho phép ghi đè phương thức (Method Overriding):

- Kế thừa cho phép các lớp con ghi đè (override) các phương thức của lớp cha, như toString() trong

DigitalVideoDisc, Book, và CompactDisc. Khi chúng ta gọi m.toString() trong vòng lặp for (Media m : mediae),

Java sử dụng cơ chế dynamic method dispatch để gọi phương thức toString() của lớp con tương ứng.

c. Hỗ trợ tham chiếu đa hình:

- Nhờ kế thừa ta có thể gán một đối tượng lớp con cho biến kiểu lớp cha. Trong Cart và Store, em sử dụng ArrayList<Media>

để lưu trữ các đối tượng thuộc nhiều lớp con khác nhau. Khi gọi các phương thức như toString() hoặc equals() (Sec 15), Java

tự động gọi phiên bản của lớp con, thể hiện đa hình.

d. Tăng tính tái sử dụng và trừu tượng:

- Kế thừa cho phép định nghĩa các thuộc tính và phương thức chung trong lớp cha Media (như id, title, category, cost, toString()).

Các lớp con không cần định nghĩa lại những phần chung này, mà chỉ cần mở rộng hoặc ghi đè khi cần. Điều này giúp đạt được đa hình một

cách hiệu quả và gọn gàng.

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

a.Định nghĩa:

- Kế thừa (Inheritance): Là cơ chế cho phép một lớp (lớp con) thừa hưởng các thuộc tính và phương thức của một lớp khác (lớp cha).

Nó thiết lập mối quan hệ "là một" (is-a).

- Đa hình (Polymorphism): Là khả năng của một đối tượng có thể được xem và xử lý như một đối tượng của lớp cha, nhưng vẫn thực thi

hành vii của lớp con. Đa hình thường được thực hiện thông qua ghi đè phương thức (method overriding) và tham chiếu kiểu lớp cha.

b. Mục đích:

- Kế thừa: Dùng để tái sử dụng mã và thiết lập mối quan hệ phân cấp giữa các lớp, giúp tránh lặp mã và tạo cấu trúc rõ ràng.

- Đa hình: Dùng để tăng tính linh hoạt và trừu tượng, cho phép mã hoạt động với các đối tượng khác nhau thông qua một giao diện chung.

c. Cách hoạt động:

- Kế thừa: Xảy ra ở thời điểm biên dịch (compile-time), khi ta định nghĩa mối quan hệ giữa các lớp bằng từ khóa extends.

VD: class DigitalVideoDisc extends Disc thiết lập mối quan hệ kế thừa ngay khi viết mã.

- Đa hình: Xảy ra ở thời điểm chạy (runtime), khi Java quyết định phương thức nào sẽ được gọi dựa trên kiểu thực tế của đối tượng (dynamic method dispatch).

Ví dụ: Khi gọi m.toString() trong Aims.java, Java xác định kiểu thực tế của m (có thể là CompactDisc, DigitalVideoDisc, hoặc Book) và gọi

phương thức toString() tương ứng.

d. Mối quan hệ:

- Kế thừa là điều kiện tiên quyết để đạt được đa hình trong hầu hết các trường hợp. Nếu không có kế thừa, các lớp con không thể được xem như

một thể hiện của lớp cha, và đa hình không thể xảy ra.

- Tuy nhiên, không phải mọi trường hợp kế thừa đều dẫn đến đa hình. Đa hình chỉ xảy ra khi có ghi đè phương thức và tham chiếu kiểu lớp cha được sử dụng.

e. Khác biệt:

- Kế thừa là cơ chế tái sử dụng mã và thiết lập phân cấp (compile-time).

- Đa hình là khả năng linh hoạt trong hành vi (runtime), dựa trên kế thừa và ghi đè phương thức.