**Task5: Tìm hiểu về lập trình hướng đối tượng Java**

**1. Khái niệm về lập trình hướng đối tượng**

**OOP** là viết tắt của Object-Oriented Programming, hay còn được gọi lập trình hướng đối tượng.

Lập trình thủ tục là viết các thủ tục hoặc phương thức thực hiện các hoạt động trên dữ liệu. Còn**lập trình hướng đối tượng** là tạo các đối tượng chứa cả dữ liệu và phương thức. OOP giúp tăng năng suất, dễ dàng mở rộng trong thiết kế phần mềm, đơn giản hóa việc bảo trì.

Lập trình hướng đối tượng có một số ưu điểm so với lập trình thủ tục:

* OOP nhanh hơn và dễ thực thi hơn
* OOP cung cấp một cấu trúc rõ ràng cho các chương trình
* OOP giúp giữ cho mã Java DRY “Don’t Repeat Yourself” và làm cho mã dễ bảo trì, sửa đổi và gỡ lỗi hơn
* OOP giúp bạn có thể tạo các ứng dụng có thể tái sử dụng đầy đủ với ít mã hơn và thời gian phát triển ngắn hơn

.

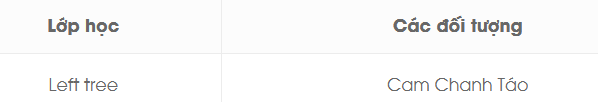
**2. Lớp và đối tượng trong Java**

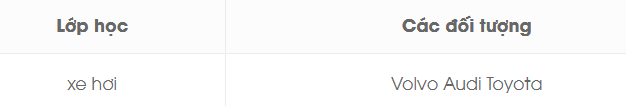
Lớp (Class) và đối tượng (Objects) là hai khía cạnh chính của lập trình hướng đối tượng java.

Lớp là một nhóm đối tượng có các thuộc tính chung, là một thực thể logic. Đối tượng là một thể hiện kết quả, thực thể của một lớp. Đối tượng có thể là đối tượng vật lý hoặc là đối tượng logic.

Một lớp trong java có thể chứa:

* Thành viên dữ liệu
* Constructor
* Phương thức
* Khối lệnh
* Lớp và Interface

Xem ví dụ vào minh họa sau để thấy sự khác biệt giữa lớp và các đối tượng:



**3. Thuộc tính của OOP**

* Tính trừu tượng

Trừu tượng nghĩa là sử dụng những thứ đơn giản để thể hiện sự phức tạp. Hay còn được hiểu đơn giản hơn là tổng quát hóa các thuộc tính, phương thức của thuộc tính. Ví dụ như để bật được tivi, chúng ta chỉ cần bấm vào nút điều khiển ở remote tivi, còn nguyên lý hay cách thức hoạt động của remote sau khi ấn nút để mở tivi thì chúng ta không cần quan tâm.

Tính trừu tượng trong Java cũng vậy. Những thứ đơn giản như đối tượng (Object), lớp (Class) và biến (Variable) sẽ đại diện cho code và dữ liệu phức tạp hơn.

Tính trừu tượng giúp bạn tập trung vào những yếu tố cốt lõi của đối tượng. Điều này cho phép tránh lặp lại cùng một công việc nhiều lần.

* Tính đóng gói

Đây là cách thực hiện giữa các trường trong một lớp ở chế độ riêng tư. Sau đó cung cấp quyền truy cập vào chúng thông qua các phương thức công khai.

Đó là một hàng rào bảo vệ giữ cho dữ liệu và code an toàn trong chính lớp đó. Bằng cách này, chúng ta có thể sử dụng lại các đối tượng như thành phần code hoặc biến mà không cho phép truy cập mở vào toàn bộ hệ thống dữ liệu.

* Tính di sản

Đây là một tính năng đặc biệt của **lập trình hướng đối tượng trong Java**. Nó cho phép các lập trình viên tạo các lớp mới chia sẻ một số thuộc tính của các lớp hiện có. Điều này cho phép người dùng có thể sử dụng các thuộc tính trên các lớp trước đó mà không cần viết lại.

* Tính đa hình

Khái niệm tính đa hình của **Java OOP** này, cho phép các lập trình viên sử dụng cùng đối tượng để thực hiện những tác vụ khác nhau – một dạng đa hình trong Java. Đây là phương thức nạp chồng khi một lớp có nhiều thuộc tính, phương thức cùng tên nhưng khác nhau về 1 số tham số khác.

Hình thức khác là phương thức ghi đè. Đây là hai phương thức cùng tên, cùng tham số, cùng kiểu trả về. Nó xuất hiện ở lớp cha và tiếp tục xuất hiện ở cả lớp con.

**4. Cách thức hoạt động của OOP**

1. Kế thừa (Inheritance)

Kế thừa là một trong 4 cách thức hoạt động của Java OOP với mục đích tránh lặp lại các đoạn code, tiết kiệm được thời gian cho người dùng.

Nó hoạt động bằng cách cho phép một lớp mới áp dụng các thuộc tính của lớp khác. Chúng ta gọi lớp kế thừa là lớp con. Lớp ban đầu thường được gọi là lớp cha. Chúng tôi sử dụng từ khóa expand để định nghĩa một lớp mới kế thừa các thuộc tính từ một lớp cũ.

1. Đóng gói (Encapsulation)

Tính năng đóng gói cho phép sử dụng lại chức năng có sẵn, nghĩa là mỗi “gói” được xây dựng để thực hiện một chức năng đặc biệt của nó. Bên cạnh đó chức năng đóng gói còn để bảo mật một số hàm, phương thức hoặc giá trị của đối tượng, không để bên ngoài nhìn thấy cũng như không thể tác động vào.

Đó là một cách thức hoạt động mạnh mẽ  của OOP trong Java. Và nó giúp chúng ta tiết kiệm rất nhiều thời gian.

*Ví dụ*: Chúng tôi có thể tạo một đoạn code gọi dữ liệu cụ thể từ cơ sở dữ liệu. Tính năng đóng gói sẽ rất hữu ích khi cho phép sử dụng lại code đó với cơ sở dữ liệu hoặc quy trình khác, trong khi vẫn giữ dữ liệu ban đầu  ở chế độ riêng tư. Ngoài ra, nó cũng cho phép chúng tôi thay đổi code gốc của mình mà không ảnh hưởng đến những người dùng khác.

1. Trừu tượng (Abstraction)

Tính trừu tượng cũng là một trong những thuộc tính OOP trong Java, hoạt động bằng cách cho phép các lập trình viên tạo ra các công cụ hữu ích và có thể tái sử dụng.

*Ví dụ:* Một lập trình viên có thể tạo một số loại đối tượng khác nhau (có thể là các biến, hàm hoặc cấu trúc dữ liệu). Người lập trình cũng có thể tạo các lớp đối tượng khác nhau (những cách để xác định các đối tượng).

Một ví dụ điểm hình, “lớp” (class) biến có thể là “địa chỉ”. Lớp có thể chỉ định rằng, mỗi đối tượng “địa chỉ” có  số nhà, tên đường, quận/huyện, thành phố và mã zip.  Đối tượng, trong trường hợp này có thể là địa chỉ nhân viên, địa chỉ khách hàng hoặc địa chỉ nhà cung cấp.

1. Đa hình (Polymorphism)

Tính đa hình trong Java hoạt động bằng cách sử dụng một tham chiếu đến lớp cha để ảnh hưởng đến một đối tượng trong lớp con. Chúng ta có thể tạo một lớp gọi là “ngựa” bằng cách mở rộng lớp “động vật”. Lớp đó cũng có thể triển khai lớp “đua xe chuyên nghiệp”. Lớp “ngựa” là “đa hình”, vì nó kế thừa các thuộc tính của cả lớp “động vật” và “đua chuyên nghiệp”.

Hai ví dụ khác về tính đa hình trong Java là ghi đè và nạp chồng (xếp chồng).

Trong **phương thức ghi đè**, lớp con có thể sử dụng khái niệm đa hình OOP để ghi đè một phương thức của lớp cha. Điều đó cho phép lập trình viên sử dụng một phương thức theo những cách khác nhau tùy thuộc vào việc nó được gọi bởi một đối tượng của lớp cha hay một đối tượng của lớp con.

Trong **phương thức nạp chồng**, một phương thức đơn lẻ có thể thực hiện các chức năng khác nhau tùy thuộc vào ngữ cảnh mà nó được gọi. Có nghĩa là, một tên phương thức duy nhất có thể hoạt động theo những cách khác nhau, tùy thuộc vào đối số nào được truyền cho nó.