OOP หรือ Object-Oriented Programming การเขียนโปรแกรมเชิงวัตถุ เป็นแนวคิดที่ใช้ในหลายภาษาโปรแกรม ซึ่งมีหลักการเหมือนกัน เช่น การใช้คลาส (Class), ออบเจ็กต์ (Object), การสืบทอด (Inheritance), การห่อหุ้ม (Encapsulation), และ การโพลีมอร์ฟิซึม (Polymorphism) แต่การใช้งานในแต่ละภาษาจะมีลักษณะเฉพาะตัวที่แตกต่างกันไป

ประกอบด้วย 4 หลักการหลัก ซึ่งเป็นพื้นฐานของการพัฒนาโปรแกรมเชิงวัตถุ ได้แก่

1. **การห่อหุ้ม (Encapsulation)** คือ การซ่อนรายละเอียดการทำงานภายในของออบเจ็กต์ และอนุญาตให้เข้าถึงหรือแก้ไขข้อมูลเฉพาะผ่านเมธอด (methods) ที่ถูกออกแบบไว้เท่านั้น
   - ช่วยป้องกันการเปลี่ยนแปลงข้อมูลโดยตรง และควบคุมการเข้าถึงข้อมูลผ่านตัวระบุการเข้าถึง (Access Modifiers) เช่น `public`, `private`, และ `protected` (Java, C++) หรือการใช้ `_` และ `__` ใน Python
2. **การสืบทอด (Inheritance)** คือ การที่คลาสลูก (Subclass) สามารถสืบทอดคุณสมบัติและพฤติกรรม (methods) จากคลาสพ่อ (Superclass) ได้ โดยไม่จำเป็นต้องเขียนโค้ดซ้ำ
   - คลาสลูกสามารถขยายหรือปรับปรุงพฤติกรรมของคลาสพ่อได้ เช่น การเขียนทับเมธอด (Method Overriding)
3. **โพลีมอร์ฟิซึม (Polymorphism)** คือ ความสามารถในการใช้เมธอดเดียวกันหรืออินเทอร์เฟซเดียวกันทำงานกับออบเจ็กต์หลายประเภท โดยที่แต่ละออบเจ็กต์สามารถมีการทำงานที่แตกต่างกันออกไป
   - ตัวอย่างเช่น การมีเมธอด `sound()` ที่อยู่ในคลาส `Animal` ซึ่งมีคลาสลูกอย่าง `Dog` และ `Cat` เขียนทับ (override) เมธอด `sound()` แต่มีพฤติกรรมต่างกัน
4. **แอบสแตรคชัน (Abstraction)** คือ การซ่อนรายละเอียดที่ไม่จำเป็นของการทำงานภายใน และแสดงเฉพาะสิ่งที่ผู้ใช้ต้องรู้หรือใช้งานเท่านั้น

- แอบสแตรคชันสามารถทำได้โดยการใช้คลาสแอบสแตรค (Abstract Class) หรือ อินเทอร์เฟซ (Interface) ที่ประกาศเมธอดไว้แต่ยังไม่ได้ระบุการทำงานจริง ซึ่งจะให้คลาสลูกมาเขียนเมธอดเหล่านั้นเอง

เปรียบเทียบการเขียนโปรแกรม OOP ด้วย Java,C++,Python

	Java	C++	Python
ประเภทภาษา	ภาษาที่คอมไพล์เป็น bytecode และรันบน JVM (Java Virtual Machine)	ภาษาที่คอมไพล์เป็น machine code โดยตรง	ภาษาที่ตีความ (interpreted language)
การประกาศคลาส	class ClassName { }	class ClassName { };	class ClassName:
การสร้างออบเจ็กต์	ClassName obj = new ClassName();	ClassName obj; หรือ ClassName* obj = new ClassName();	obj = ClassName()
การสืบทอด (Inheritance)	ใช้ extends สำหรับการสืบทอดเพียงคลาสเดียว	รองรับการสืบทอดหลายคลาส(multiple inheritance) โดยใช้ :	รองรับการสืบทอดหลายคลาสโดยใช้ ()
การเข้าถึงตัวแปรและเมธอด	ใช้ตัวระบุ public, protected, private ในการกำหนดระดับการเข้าถึง	เหมือนกับ Java (public, protected, private)	ใช้ _ และ __ (underscore) ในการกำหนดระดับการเข้าถึง (ไม่เข้มงวดเท่า Java)

โพลีมอร์ฟิซึม (Polymorphism)	สนับสนุนผ่าน method overriding และ overloading	สนับสนุนผ่าน method overriding และ overloading	สนับสนุน method overriding (แต่ไม่มีการ overloading ที่แท้จริง)
การจัดการหน่วยความจำ	ใช้ garbage collection ในการจัดการหน่วยความจำโดยอัตโนมัติ	ต้องจัดการหน่วยความจำด้วยตัวเอง (เช่น การลบออบเจ็กต์ด้วย delete)	ใช้ garbage collection เช่นเดียวกับ Java
คอนสตรัคเตอร์	มีคอนสตรัคเตอร์ชื่อเดียวกับชื่อคลาส สามารถ overload ได้	มีคอนสตรัคเตอร์ชื่อเดียวกับชื่อคลาส สามารถ overload ได้	ใช้เมธอด __init__ เป็นคอนสตรัคเตอร์
การห่อหุ้ม (Encapsulation)	ใช้ตัวระบุการเข้าถึง (access modifiers) เพื่อห่อหุ้มข้อมูล	เหมือนกับ Java โดยใช้ access specifiers (private, public, protected)	ห่อหุ้มผ่าน _ และ __ ในการซ่อนข้อมูล
การจัดการข้อยกเว้น (Exception Handling)	ใช้ try-catch-finally ในการจัดการข้อยกเว้น	ใช้ try-catch ในการจัดการข้อยกเว้น แต่มีรูปแบบไม่ซับซ้อนเท่า Java	ใช้ try-except-finally ในการจัดการข้อยกเว้น
อินเตอร์เฟซ/มัลติเพิลอินเฮอริตันซ์	สนับสนุนอินเตอร์เฟซ (interface) แต่ไม่รองรับ multiple inheritance โดยตรง	รองรับ multiple inheritance โดยตรง	สนับสนุน multiple inheritance และอินเตอร์เฟซ

ความเหมือน

- ทั้งสามภาษาเป็นภาษาที่สนับสนุนการเขียนโปรแกรมเชิงวัตถุ (OOP) และใช้แนวคิดหลักเดียวกัน เช่น การใช้คลาสและออบเจ็กต์
- การห่อหุ้มข้อมูล (Encapsulation), การสืบทอด (Inheritance), และการใช้โพลีมอร์ฟิซึม (Polymorphism) เป็นแนวคิดที่ใช้ในทุกภาษา

ความแตกต่าง

1.การคอมไพล์และการรัน

Java: คอมไพล์เป็น bytecode และรันบน JVM ทำให้สามารถรันได้บนหลายแพลตฟอร์ม

C++: คอมไพล์เป็น machine code โดยตรง ทำให้สามารถทำงานได้รวดเร็วกว่าในบางกรณี แต่ต้องจัดการหน่วยความจำเอง

Python: เป็นภาษาที่ตีความ ทำให้เขียนโค้ดได้ง่ายและยืดหยุ่น แต่ประสิทธิภาพอาจต่ำกว่า

2.**การจัดการหน่วยความจำ**

C++ ต้องจัดการหน่วยความจำเองโดยใช้ `new` และ `delete`

Java และ Python ใช้ระบบ garbage collection ในการจัดการหน่วยความจำโดยอัตโนมัติ

3.**รูปแบบการประกาศและสร้างออบเจ็กต์** รูปแบบในการสร้างคลาสและออบเจ็กต์แตกต่างกันในแต่ละภาษา

4.การสืบทอดหลายคลาส

C++ และ **Python** รองรับการสืบทอดหลายคลาสโดยตรง

Java ไม่รองรับการสืบทอดหลายคลาสโดยตรง แต่ใช้ `interface` แทน

แต่ละภาษามีความแข็งแกร่งในด้านต่าง ๆ เช่น **Java** เหมาะกับการพัฒนาแอปพลิเคชันขนาดใหญ่, **C++** เหมาะกับงานที่ต้องการประสิทธิภาพสูง, และ **Python** เหมาะกับการพัฒนาโปรเจกต์ที่ต้องการความรวดเร็วในการพัฒนาและความยืดหยุ่น

ตัวอย่าง Code

การประกาศคลาส

Java

```java
class Animal {
    void makeSound() {
        System.out.println("Animal sound");
    }
}
```

C++

```cpp
class Animal {
public:
    void makeSound() {
        cout << "Animal sound" << endl;
    }
};
```

Python

```python
class Animal:
    def make_sound(self):
        print("Animal sound")
```

การสืบทอด

Java

```java
class Dog extends Animal {
    void makeSound() {
        System.out.println("Bark");
    }
}
```

C++

```cpp
class Dog : public Animal {
public:
    void makeSound() {
        cout << "Bark" << endl;
    }
};
```

Python

```python
class Dog(Animal):
    def make_sound(self):
        print("Bark")
```