Difference between Inheritance and Polymorphism

Difficulty Level : Basic • Last Updated : 24 Aug, 2020

Inheritance:

Inheritance is one in which a new class is created that inherits the properties of the already exist class. It supports the concept of code reusability and reduces the length of the code in object-oriented programming.

Types of Inheritance are:

1. Single inheritance
2. Multi-level inheritance
3. Multiple inheritance
4. Hybrid inheritance
5. Hierarchical inheritance

Example of Inheritance:

C++

```
#include "iostream"
using namespace std;

class A {
    int a, b;

public:
    void add(int x, int y)
    {
        a = x;
        b = y;
```

```
        cout << (a + b) << endl;
    }
};

class B : public A {
public:
    void print(int x, int y)
    {
        add(x, y);
    }
};

int main()
{
    B b1;
    b1.print(5, 6);
}
```

Java

```
class A {
    int a, b;
    public void add(int x, int y)
    {
        a = x;
        b = y;
        System.out.println("addition of a + b is:" + (a + b));
    }
}
class B extends A {
    public void sum(int x, int y)
    {
        add(x, y);
    }

    // Driver Code
    public static void main(String[] args)
    {
        B b1 = new B();
        b1.sum(5, 6);
    }
}
```

Output:

```
addition of a+b is:11
```

Here, class B is the derived class which inherit the property(**add method**) of the base class A.

Polymorphism:

Polymorphism is that in which we can perform a task in multiple forms or ways. It is applied to the functions or methods. Polymorphism allows the object to decide which form of the function to implement at compile-time as well as run-time.

Types of Polymorphism are:

1. Compile-time polymorphism (Method overloading)
2. Run-time polymorphism (Method Overriding)

Example of Polymorphism:

C++

```
#include "iostream"
using namespace std;

class A {
    int a, b, c;

public:
    void add(int x, int y)
    {
        a = x;
        b = y;
        cout << "addition of a+b is:" << (a + b) << endl;
    }

    void add(int x, int y, int z)
    {
        a = x;
        b = y;
        c = z;
        cout << "addition of a+b+c is:" << (a + b + c) << endl;
    }
```

```
    void print()
    {
        cout << "Class A's method is running" << endl;
    }
};

class B : public A {
public:
    void print()
    {
        cout << "Class B's method is running" << endl;
    }
};

int main()
{
    A a1;

    // method overloading (Compile-time polymorphism)
    a1.add(6, 5);

    // method overloading (Compile-time polymorphism)
    a1.add(1, 2, 3);

    B b1;

    // Method overriding (Run-time polymorphism)
    b1.print();
}
```

Java

```
class A {
    int a, b, c;

    public void add(int x, int y)
    {
        a = x;
        b = y;
        System.out.println("addition of a+b is:" + (a + b));
    }

    public void add(int x, int y, int z)
    {
        a = x;
        b = y;
        c = z;
        System.out.println("addition of a+b+c is:" + (a + b + c));
    }
```

```
    public void print()
    {
        System.out.println("Class A's method is running");
    }
};

class B extends A {
    public void print()
    {
        System.out.println("Class B's method is running");
    }

    // Driver Code
    public static void main(String[] args)
    {
        A a1 = new A();

        // method overloading (Compile-time polymorphism)
        a1.add(6, 5);

        // method overloading (Compile-time polymorphism)
        a1.add(1, 2, 3);

        B b1 = new B();

        // Method overriding (Run-time polymorphism)
        b1.print();
    }
}
```

Output:

```
addition of a+b is:11
addition of a+b+c is:6
Class B's method is running
```

Difference between Inheritance and Polymorphism:

S.NO	Inheritance	Polymorphism

S.NO	Inheritance	Polymorphism
1.	Inheritance is one in which a new class is created (derived class) that inherits the features from the already existing class(Base class).	Whereas polymorphism is that which can be defined in multiple forms.
2.	It is basically applied to classes.	Whereas it is basically applied to functions or methods.
3.	Inheritance supports the concept of reusability and reduces code length in object-oriented programming.	Polymorphism allows the object to decide which form of the function to implement at compile-time (overloading) as well as run-time (overriding).
4.	Inheritance can be single, hybrid, multiple, hierarchical and multilevel inheritance.	Whereas it can be compiled-time polymorphism (overload) as well as run-time polymorphism (overriding).
5.	It is used in pattern designing.	While it is also used in pattern designing.

Want to learn from the best curated videos and practice problems, check out the C++ Foundation Course for Basic to Advanced C++ and C++ STL Course for foundation plus STL. To complete your preparation from learning a language to DS Algo and many more, please refer **Complete Interview Preparation Course**.