Difference Between Polymorphism and Inheritance in OOP

www.differencebetween.com

Key Difference - Polymorphism vs Inheritance in OOP

Object-Oriented Programming (OOP) is commonly used to develop software. Many programming languages support object-oriented programming. Object-oriented programming is a methodology to design a program using classes and objects. A class in OOP is a blueprint to create an object. A class has properties and methods. An object is an instance of a class. OOP contains four pillars such as Inheritance, Polymorphism, Abstraction and Encapsulation. This article discusses the difference between Polymorphism and Inheritance in OOP. The **key difference** between Polymorphism and Inheritance in OOP is that **Polymorphism is the ability of an object to behave in multiple ways and Inheritance is to create a new class using properties and methods of an existing class.**

What is Polymorphism in OOP?

Polymorphism is to indicate multiple forms. One object can have multiple behaviours. Polymorphism can be divided into two categories. They are overloading and overriding.

Overloading

Refer the below program written in Java.

```
public class HelloWorld
{

  public static void main(String[] args)
  {
    A obj= new A();
    obj.sum();
    obj.sum(2,3);
  }
}

public class A
{
    public void sum(){
        System.out.println("Sum");
    }

    public void sum(int a, int b){
        System.out.println("Sum is "+(a+b));
    }

}
```

Figure 01: Overloading

According to the above program, an object of type A is created. When calling obj.sum(); it will give the output related to method sum(). When calling the obj.sum(2,3); it will give the output related to sum(int a, int b). It can be observed that the same object has different behaviours depending on the situation. When there are multiple methods with the same name, but with different parameters, it is known as **overloading**. It is also known as **static binding** or **compile time polymorphism**.

Overriding

Another type of Polymorphism is **overriding**. Refer the below program written in Java.

```
public class HelloWorld
{
  public static void main(String[] args)
  {
    A obj = new B();
    obj.display();
  }
}

public class A
{
  public void display(){
    System.out.println("A");
  }

}

public class B extends A{

  public void display(){
    System.out.println("B");
  }

}
```

Figure 02: Overriding

According to the above program, there is a method display() in class A. Class B extends from class A. Therefore, all methods in class A is accessible by class B. It is inheritance. The inheritance concept is described further later.

Class B also have the same method display(). When creating an object of type A and calling display method, the output will give B. Class A display method is overridden by class B display method. So, the output is B.

When there are methods with same name and same parameters but in two different classes, and they are linked with inheritance it is known as overriding. It is also known as **Late binding, Dynamic Binding, Runtime Polymorphism**. Overloading and overriding are called as Polymorphism. It is a major concept in Object Oriented Programming.

What is Inheritance in OOP?

Refer the below program written in Java.

```
public class HelloWorld
{

  public static void main(String[] args)
  {
    B obj= new B();
    obj.sub();
    obj.sum();
  }
}

public class A{

  public void sum(){
    System.out.println("sum");
  }
}

public class B extends A{

  public void sub(){
    System.out.println("sub");
  }

}
```


Figure 03: Example of Inheritance

According to the above program, class A has method sum() and class B has method sub().

The sum() method of class A can be used in class B using extend keyword. Reusing the properties and methods in an existing class to create a new class is known as Inheritance. Even there is no sum() method in class B; it is inherited from class A. Inheritance is useful for code reusability. The older class is called the **base class, superclass** or **parent class**. The derived class is called the **subclass** or **child class**.

There are various types of inheritance. They are Single-Level Inheritance, Multi-Level Inheritance, Multiple Inheritance, Hierarchical Inheritance and Hybrid Inheritance. In Single Inheritance, there is one super class and one sub class. If class A is the super class and class B is the sub class, all the properties and methods of class A is accessible by class B. There is only one level. Therefore it is called as single-level inheritance.

In Multi-Level Inheritance, there are three levels of classes. The intermediate class inherits from super class. The sub class inherits from the intermediate class. If there are three classes as A, B and C and A is the super class and B is the intermediate class. Then B inherits from A and C inherits from B, it is a Multi-Level Inheritance.

In Multiple Inheritance, there are many super classes and one sub class. If there are three super classes called A, B, C and D is the sub class, then class D can inherit from A, B and C. Multiple Inheritance is supported in programming language C++. It is not

supported in programming languages such as Java or C#. Interfaces are used for implementing Multiple Inheritance in these languages.

If there are classes called A as super class and B, C are sub classes, those sub classes can inherit properties and methods of class A. That kind of inheritance type is known as Hierarchical Inheritance.

There is another special inheritance type which is known as Hybrid inheritance. It is a combination of multi-level and multiple inheritances. If A, B, C and D are classes and B is inheriting from A and D is inheriting from both B and C, then it is a Hybrid inheritance.

What are the Similarities Between Polymorphism and Inheritance in OOP?

- Both are concepts of Object Oriented Programming.

What is the Difference Between Polymorphism and Inheritance in OOP?

Polymorphism vs Inheritance in OOP	
Polymorphism is an ability of an object to behave in multiple ways.	Inheritance is to create a new class using properties and methods of an existing class.
Usage	
Polymorphism is used for objects to call which form of methods at compile time and runtime.	Inheritance is used for code reusability.
Implementation	
Polymorphism is implemented in methods.	Inheritance is implemented in classes.
Categories	
Polymorphism can be divided into overloading and overriding.	Inheritance can be divided into single-level, multi-level, hierarchical and multiple inheritance.

Summary - Polymorphism vs Inheritance in OOP

Polymorphism and Inheritance are major concepts in Object Oriented Programming. The difference between Polymorphism and Inheritance in OOP is that Polymorphism is

a common interface to multiple forms and Inheritance is to create a new class using properties and methods of an existing class. Both concepts are widely used in Software Development.

Reference

1.*8.3 What is Inheritance in Java | Lecture | Tutorial*, Telusko Learning, 15 May 2014. Available here

2. *Polymorphism in Java Tutorial*, Telusko Learning, 15 May 2014. Available here

How to Cite this Article?

APA: Difference Between Polymorphism and Inheritance in OOP.(2018 January 16). Retrieved (date), from http://differencebetween.com/difference-between-polymorphism-and-vs-inheritance-in-oop/

MLA: "Difference Between Polymorphism and Inheritance in OOP" Difference Between.Com. 16 January 2018. Web.

Chicago: “Difference Between Polymorphism and Inheritance in OOP.” Difference Between.Com. http://differencebetween.com/difference-between-polymorphism-and-vs-inheritance-in-oop/ accessed (accessed [date]).