Blog on Different types of polymorphism and comparison of polymorphism in C++ and Java

**Introduction**

In the real world, you might have seen a chameleon changing its color as per its requirement. If someone asks, “How it does that?”, you can simply say, “Because, it is polymorphic”. Similarly, in the programming world, objects possess the same functionality where each object can take multiple forms. This property is known as Polymorphism, where Poly means many and morph means change (or ‘form’). In this blog, let’s discuss this key concept of [Object Oriented Programming](https://www.edureka.co/blog/object-oriented-programming/) that is Polymorphism in Java and C++.

**Polymorphism** in OOP is the ability of an entity to take several forms. In other words, it refers to the ability of an [object](https://www.edureka.co/blog/java-tutorial/#obj) (or a reference to an object) to take different forms of objects. It allows a common data-gathering message to be sent to each class. Polymorphism encourages called as ‘extendibility’ which means an object or a class can have its uses extended.



In the above figure, you can see, Man is only one, but he takes multiple roles like – he is a dad to his child, he is an employee, a salesperson and many more. This is known as Polymorphism.

**Polymorphism in Java**

Consider a cell phone where you save your Contacts. Suppose a person has two contact numbers. For the ease of accessibility, your cellphone provides you the functionality where you can save two numbers under the same name.

Similarly, in [Java](https://www.edureka.co/blog/java-tutorial/), an object is only one but it can take multiple forms depending on the context of the program. Suppose you want to write a function to save two contact numbers of the same person, you can create it like –  **void createContact(String name, int number1, int number2)**.

Now, it’s not necessary that everyone in your contact list will have two contact numbers. Few of them might be having only a single contact number. In such situations, instead of creating another method with a different name to save one number for a contact, what you can do is, create another method with the same name i.e. **createContact().** But, instead of taking two contact numbers as parameters, take only one contact number as a parameter i.e. **void createContact(String name, int number1)**.

**createContact()**method has two different definitions. Here, which definition is to be executed depends upon the number of parameters being passed. If one parameter is passed, then only a single contact number is saved under the contact. But, if two contact numbers are passed to this method at the same time, then both will be saved under the same contact. *This is also known as****Method Overloading****.*

**Types of Polymorphism in Java**

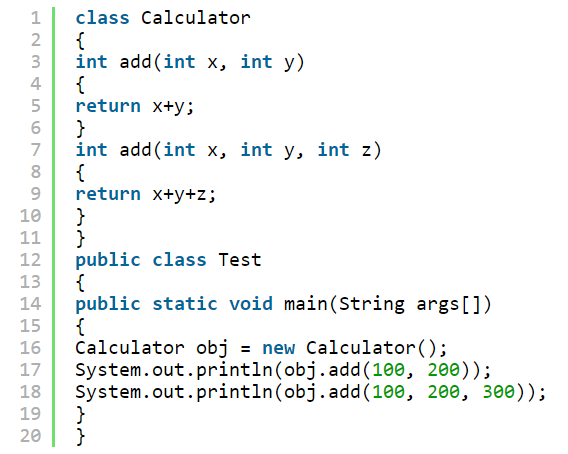
Java supports two types of polymorphism and they are as follows:

* Static Polymorphism
* Dynamic Polymorphism

**Static Polymorphism**

A polymorphism that is resolved during compile time is known as static polymorphism.

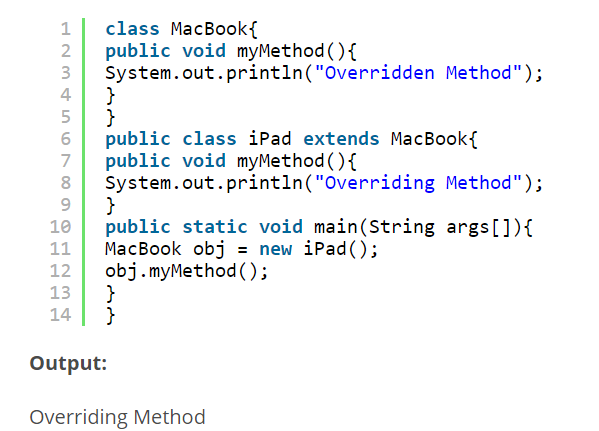
**Method Overloading** is a feature that allows a class to have two or more **method** to have the same name, but with different parameter lists. In the below example, you have two definitions of the same method add(). So, which add() method would be called is determined by the parameter list at the compile time. That is the reason this is also known as compile time polymorphism.



**Dynamic Polymorphism**

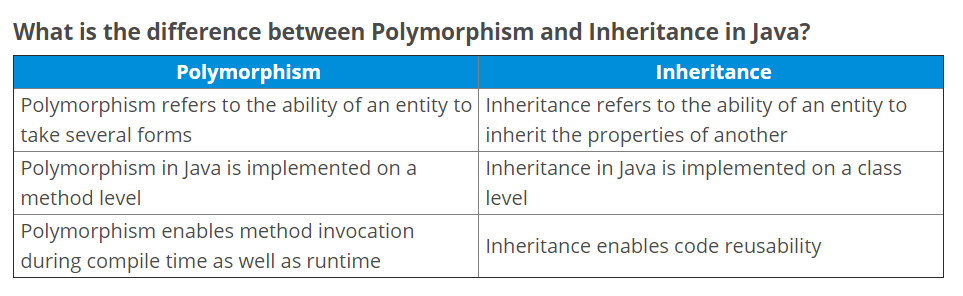
Dynamic polymorphism is a process in which a call to an overridden method is resolved at runtime, that’s why it is called runtime polymorphism. Method Overriding is one of the ways to achieve Dynamic Polymorphism. In any object-oriented programming language, **Overriding** is a feature that allows a subclass or child class to provide a specific implementation of a **method** that is already provided by one of its super-classes or parent classes.

In the below example, you have two classes **MacBook** and **iPad**. MacBook is a parent class and iPad is a child class. The child class is overriding the method **myMethod()** of the parent class. Here, I have assigned child class object to the parent class reference to determine which method would be called at run-time. It is the type of object that determines which version of the method would be called (not the type of reference).

****

**Advantages of Dynamic Polymorphism**

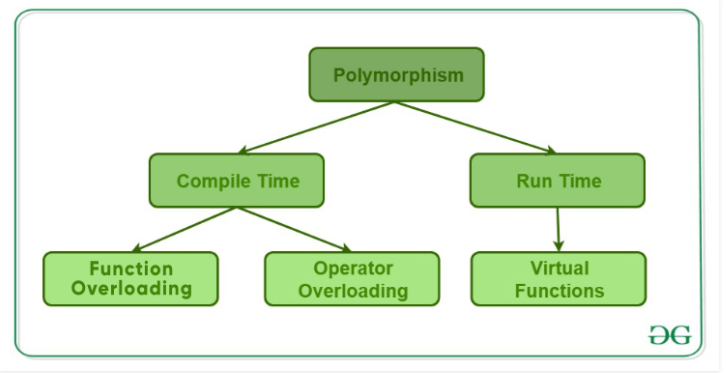
1. Dynamic Polymorphism allows Java to support the overriding of methods which is central for run-time polymorphism.
2. It allows a class to specify methods that will be common to all of its derivatives while allowing subclasses to define the specific implementation of some or all of those methods.
3. It also allows subclasses to add its specific methods subclasses to define the specific implementation of the same.

****

**Polymorphism in C++**

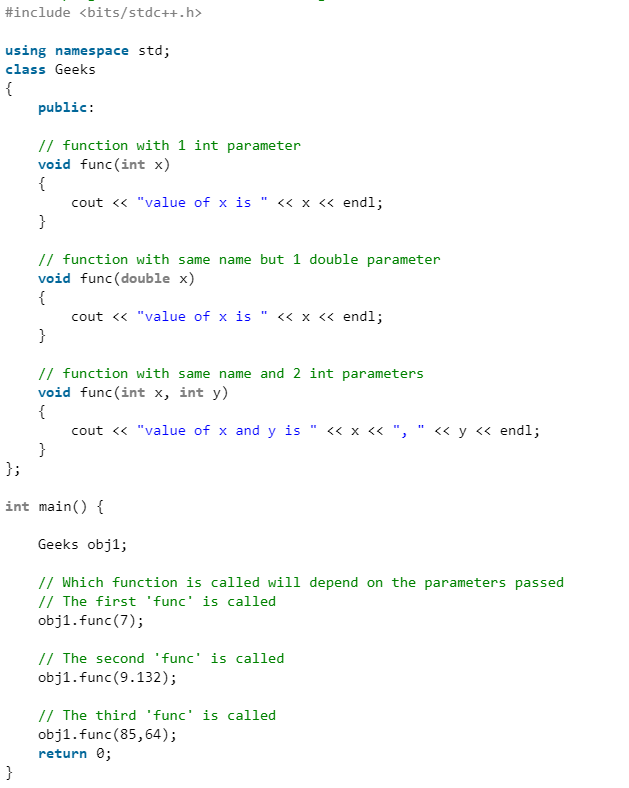
In C++ polymorphism is mainly divided into two types:

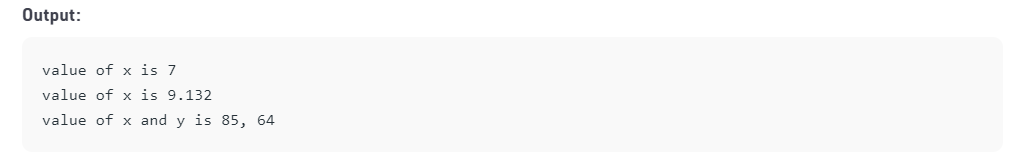
* Compile time Polymorphism
* Runtime Polymorphism



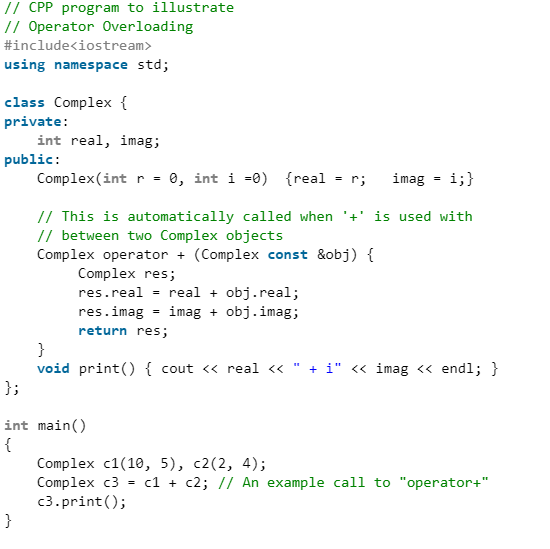
**Compile time polymorphism**: This type of polymorphism is achieved by function overloading or operator overloading.

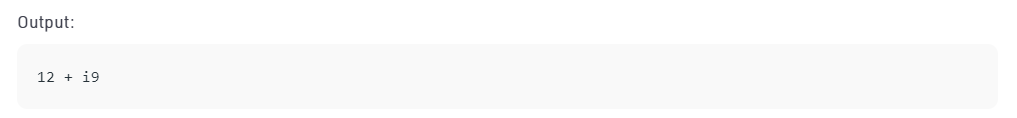
* [**Function Overloading**](https://www.geeksforgeeks.org/function-overloading-c/): When there are multiple functions with same name but different parameters then these functions are said to be overloaded. Functions can be overloaded by change in number of arguments or/and in type of arguments.





* [**Operator Overloading**](https://www.geeksforgeeks.org/operator-overloading-c/): C++ also provide option to overload operators. For example, we can make the operator (‘+’) for string class to concatenate two strings. We know that this is the addition operator whose task is to add two operands. So a single operator ‘+’ when placed between integer operands, adds them and when placed between string operands, concatenates them.





[**Runtime polymorphism**](https://www.geeksforgeeks.org/virtual-functions-and-runtime-polymorphism-in-c-set-1-introduction/): This type of polymorphism is achieved by Function Overriding.

* [**Function overriding**](https://www.geeksforgeeks.org/override-keyword-c/) on the other hand occurs when a derived class has a definition for one of the member functions of the base class. That base function is said to be overridden.

