# **Polymorphism**

**Which object-oriented concept is achieved by using overloading and overriding?**

Polymorphism.

**What is polymorphism and what are the types of it?**

Single task can be done in different way.

Method overloading(compile time polymorphism),method overriding(run time polymorphism)

**What is method overriding?**

Specific implementation of a method for child class.

**What is method overloading?**

If a class have multiple methods by same name but different parameters, it is known as Method Overloading.

**Difference between method overloading and overriding?**

|  |  |
| --- | --- |
| **Method overloading** | **Method overriding** |
| In case of method overloading, signature of method changes | While in case of method overriding it remain same. |
| Can overload method in one class | Can only be done on subclass. |
| Can overload static, final or private method in Java | Can not override static, final and private method in Java |
| Overloaded method in Java is bonded by static binding | Overridden methods are subject to dynamic binding. |

**What is run time polymorphism and compile time polymorphism?**

**Compile time polymorphism:**

it is nothing but the method overloading in java. In simple terms we can say that a class can have more than one methods with same name but with different number of arguments or different types of arguments or both.

**Runtime polymorphism:**

Runtime polymorphism or dynamic method dispatch is a process in which a call to an overridden method is resolved at runtime rather than at compile-time.

In this process, an overridden method is called through the reference variable of a super class. The determination of the method to be called is based on the object being referred to by the reference variable.

**Can we achieve polymorphism through data member?**

No, Polymorphism is always achieved via behavior of an object only properties of an object do not play any role in case of polymorphism.

**can we overload main() method?**

Yes, we can have many main() methods in a class by overloading the main method.

**package** FPPackage;

**public** **class** MainOverloading {

**public** **static** **void** main(String[] args) {

System.***out***.println("Normal main method");

*main*(3);

*main*("Subbu","Selenium");

}

**public** **static** **void** main(**int** args) {

System.***out***.println("Normal main method with one argument");

}

**public** **static** **void** main(String args1, String args2) {

System.***out***.println("Normal main method with two arguments");

}

}

When you run the above program it always prints the first main method regardless how many arguments you pass as this is the entry point for a class.

So, you have to call the other main methods from the program.