**Polymorphism:**

*The dictionary definition of*polymorphism*refers to a principle in biology in which an organism or species can have many different forms or stages. This principle can also be applied to object-oriented programming and languages like the Java language. Subclasses of a class can define their own unique behaviors and yet share some of the same functionality of the parent class.*

Def: Polymorphism is the ability of an object to take on many forms.

* It helps the programmer to reuse the codes, i.e., classes once written, tested and implemented can be reused as required. Saves a lot of time.
* Single variable can be used to store multiple data types.
* Easy to debug the codes.

**Method Overloading** is a feature that allows a class to have more than one method having the same name.

* Method name should be matched and atleast one parameter should be different.

**Method Overriding** is a feature to redefine/reimplement the method in sub-class which is already defined in super class. ­

* Method Signature should be matched, meaning that Return type, method name, and the parameters.

**Overloading vs Overriding**

1. Overloading happens at [compile-time](https://beginnersbook.com/2013/04/runtime-compile-time-polymorphism/) while Overriding happens at [runtime](https://beginnersbook.com/2013/04/runtime-compile-time-polymorphism/): The binding of overloaded method call to its definition has happens at compile-time however binding of overridden method call to its definition happens at runtime.
2. Static methods can be overloaded which means a class can have more than one static method of same name. Static methods cannot be overridden, even if you declare a same static method in child class it has nothing to do with the same method of parent class.
3. The most basic difference is that overloading is being done in the same class while for overriding base and child classes are required. Overriding is all about giving a specific implementation to the inherited method of parent class.
4. [Static binding](https://beginnersbook.com/2013/04/java-static-dynamic-binding/) is being used for overloaded methods and [dynamic binding](https://beginnersbook.com/2013/04/java-static-dynamic-binding/) is being used for overridden/overriding methods.
5. private and final methods can be overloaded but they cannot be overridden. It means a class can have more than one private/final methods of same name but a child class cannot override the private/final methods of their base class.
6. Return type of method does not matter in case of method overloading, it can be same or different. However in case of method overriding the overriding method can have more specific return type.
7. Argument list should be different while doing method overloading. Argument list should be same in method Overriding.

**Abstraction:**

Abstraction is the process of hiding certain details and showing only essential information to the user. Abstraction can be achieved with either **abstract classes** or [**interfaces**](https://www.w3schools.com/java/java_interface.asp).

The abstract keyword is a non-access modifier, used for classes and methods:

* **Abstract class:** is a restricted class that cannot be used to create objects (to access it, it must be inherited from another class).
* **Abstract method:** can only be used in an abstract class, and it does not have a body. The body is provided by the subclass (inherited from).

|  |  |
| --- | --- |
| **Abstract class** | **Interface** |
| Abstract class can **have abstract and concrete** methods. | Interface can have **only abstract** methods. Since Java 8, it can have **default and static methods** also. |
| Abstract class **doesn't support multiple inheritance**. | Interface **supports multiple inheritance**. |
| Abstract class **can provide the implementation of interface**. | Interface **can't provide the implementation of abstract class**. |
| The **abstract keyword** is used to declare abstract class. | The **interface keyword** is used to declare interface. |
| An **abstract class** can extend another Java class and implement multiple Java interfaces. | An **interface** can extend another Java interface only. |
| An **abstract class** can be extended using keyword "extends". | An **interface** can be implemented using keyword "implements". |
| A Java **abstract class** can have class members like private, protected, etc. | Members of a Java interface are public by default. |

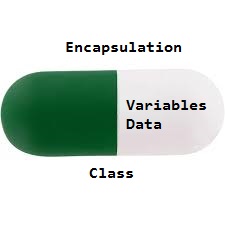
­­­­**­Constructor**is special method that is called when an object is instantiated. In other words, when you use the new keyword. The purpose of a Java constructor is to initialize the object before it is used.

* **No Return Type and name is same as class name.**
* **Constructor cannot be abstract, static, final, and synchronized.**
* Default is no-arg Constructor. If you don't define any constructor, the Java compiler will create/add a default, no-argument constructor for you. Thus, once the class is compiled it will always at least have a no-argument constructor.
* Call constructor from another constructor.
* Call super class constructor from another constructor.
* Constructor Overloading is possible.

**Keywords**:

1. [**byte**](https://www.javatpoint.com/byte-keyword-in-java)**:** Java byte keyword is used to declare a variable that can hold an 8-bit data values.
2. [**boolean:**](https://www.javatpoint.com/boolean-keyword-in-java) Java boolean keyword is used to declare a variable as a boolean type. It can hold True and False values only.
3. [**char**](https://www.javatpoint.com/char-keyword-in-java)**:** Java char keyword is used to declare a variable that can hold unsigned 16-bit Unicode characters
4. [**double**](https://www.javatpoint.com/double-keyword-in-java)**:** Java double keyword is used to declare a variable that can hold a 64-bit floating-point numbers.
5. [**float**](https://www.javatpoint.com/float-keyword-in-java)**:** Java float keyword is used to declare a variable that can hold a 32-bit floating-point number.
6. [**long**](https://www.javatpoint.com/long-keyword-in-java)**:** Java long keyword is used to declare a variable that can hold a 64-bit integer.
7. [**short**](https://www.javatpoint.com/short-keyword-in-java)**:** Java short keyword is used to declare a variable that can hold a 16-bit integer.
8. [**int**](https://www.javatpoint.com/int-keyword-in-java)**:** Java int keyword is used to declare a variable that can hold a 32-bit signed integer.
9. [**try**](https://www.javatpoint.com/try-catch-block)**:** Java try keyword is used to start a block of code that will be tested for exceptions. The try block must be followed by either catch or finally block.
10. [**catch**](https://www.javatpoint.com/try-catch-block)**:** Java catch keyword is used to catch the exceptions generated by try statements. It must be used after the try block only.
11. [**throw**](https://www.javatpoint.com/throw-keyword)**:** The Java throw keyword is used to explicitly throw an exception. The throw keyword is mainly used to throw custom exception. It is followed by an instance.
12. [**throws**](https://www.javatpoint.com/throws-keyword-and-difference-between-throw-and-throws)**:** The Java throws keyword is used to declare an exception. Checked exception can be propagated with throws.
13. [**finally**](https://www.javatpoint.com/finally-block-in-exception-handling)**:** Java finally keyword indicates a block of code in a try-catch structure. This block is always executed whether exception is handled or not.
14. [**continue**](https://www.javatpoint.com/java-continue)**:** Java continue keyword is used to continue the loop. It continues the current flow of the program and skips the remaining code at the specified condition.
15. [**default**](https://www.javatpoint.com/default-keyword-in-java)**:** Java default keyword is used to specify the default block of code in a switch statement.
16. [**do**](https://www.javatpoint.com/java-do-while-loop)**:** Java do keyword is used in control statement to declare a loop. It can iterate a part of the program several times.
17. [**while**](https://www.javatpoint.com/java-while-loop)**:** Java while keyword is used to start a while loop. This loop iterates a part of the program several times. If the number of iteration is not fixed, it is recommended to use while loop.
18. [**break**](https://www.javatpoint.com/java-break)**:** Java break keyword is used to break loop or switch statement. It breaks the current flow of the program at specified condition.
19. [**if**](https://www.javatpoint.com/java-if-else)**:** Java if keyword tests the condition. It executes the if block if condition is true.
20. [**else**](https://www.javatpoint.com/java-if-else)**:** Java else keyword is used to indicate the alternative branches in an if statement.
21. [**switch**](https://www.javatpoint.com/java-switch)**:** The Java switch keyword contains a switch statement that executes code based on test value. The switch statement tests the equality of a variable against multiple values.
22. [**for**](https://www.javatpoint.com/java-for-loop)**:** Java for keyword is used to start a for loop. It is used to execute a set of instructions/functions repeatedly when some conditions become true. If the number of iteration is fixed, it is recommended to use for loop.
23. [**private**](https://www.javatpoint.com/private-keyword-in-java)**:** Java private keyword is an access modifier. It is used to indicate that a method or variable may be accessed only in the class in which it is declared.
24. [**protected**](https://www.javatpoint.com/protected-keyword-in-java)**:** Java protected keyword is an access modifier. It can be accessible within package and outside the package but through inheritance only. It can't be applied on the class.
25. [**public**](https://www.javatpoint.com/public-keyword-in-java)**:** Java public keyword is an access modifier. It is used to indicate that an item is accessible anywhere. It has the widest scope among all other modifiers.
26. [**return**](https://www.javatpoint.com/return-keyword-in-java)**:** Java return keyword is used to return from a method when its execution is complete.
27. [**static**](https://www.javatpoint.com/static-keyword-in-java)**:** Java static keyword is used to indicate that a variable or method is a class method. The static keyword in Java is used for memory management mainly.
28. [**super**](https://www.javatpoint.com/super-keyword)**:** Java super keyword is a reference variable that is used to refer parent class object. It can be used to invoke immediate parent class method.
29. [**this**](https://www.javatpoint.com/this-keyword)**:** Java this keyword can be used to refer the current object in a method or constructor.
30. [**synchronized**](https://www.javatpoint.com/synchronization-in-java)**:** Java synchronized keyword is used to specify the critical sections or methods in multithreaded code.
31. [**transient**](https://www.javatpoint.com/transient-keyword)**:** Java transient keyword is used in serialization. If you define any data member as transient, it will not be serialized.
32. [**volatile**](https://www.javatpoint.com/volatile-keyword-in-java)**:** Java volatile keyword is used to indicate that a variable may change asynchronously.
33. [**enum**](https://www.javatpoint.com/enum-in-java)**:** Java enum keyword is used to define a fixed set of constants. Enum constructors are always private or default.
34. [**final**](https://www.javatpoint.com/final-keyword)**:** Java final keyword is used to indicate that a variable holds a constant value. It is applied with a variable. It is used to restrict the user.
35. [**abstract**](https://www.javatpoint.com/abstract-keyword-in-java)**:** Java abstract keyword is used to declare abstract class. Abstract class can provide the implementation of interface. It can have abstract and non-abstract methods.
36. [**package**](https://www.javatpoint.com/package)**:** Java package keyword is used to declare a Java package that includes the classes.
37. [**import**](https://www.javatpoint.com/package)**:** Java import keyword makes classes and interfaces available and accessible to the current source code.
38. [**class**](https://www.javatpoint.com/class-keyword-in-java)**:** Java class keyword is used to declare a class.
39. [**interface**](https://www.javatpoint.com/interface-in-java)**:** Java interface keyword is used to declare an interface. It can have only abstract methods.
40. [**extends**](https://www.javatpoint.com/inheritance-in-java)**:** Java extends keyword is used to indicate that a class is derived from another class or interface.
41. [**implements**](https://www.javatpoint.com/interface-in-java)**:** Java implements keyword is used to implement an interface.
42. [**instanceof**](https://www.javatpoint.com/downcasting-with-instanceof-operator)**:** Java instanceof keyword is used to test whether the object is an instance of the specified class or implements an interface.
43. [**new**](https://www.javatpoint.com/new-keyword-in-java)**:** Java new keyword is used to create new objects.
44. [**null**](https://www.javatpoint.com/null-keyword-in-java)**:** Java null keyword is used to indicate that a reference does not refer to anything. It removes the garbage value.
45. **void:** Java void keyword is used to specify that a method does not have a return value.

**Encapsulation** is a mechanism of wrapping the data (variables) and code acting on the data (methods) together as a single unit. In encapsulation, the variables of a class will be hidden from other classes, and can be accessed only through the methods of their current class. Therefore, it is also known as **data hiding**.



To achieve encapsulation −

* Declare the variables of a class as private.
* Provide public setter and getter methods to modify and view the variables values.

## Advantages:

* The fields of a class can be made read-only or write-only.
* A class can have total control over what is stored in its fields.