**Encapsulation**

Encapsulation in Java 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 in Java −

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

**Advantages of Encapsulation**:

* **Data Hiding:** The user will have no idea about the inner implementation of the class. It will not be visible to the user that how the class is storing values in the variables. He only knows that we are passing the values to a setter method and variables are getting initialized with that value.
* **Increased Flexibility:** We can make the variables of the class as read-only or write-only depending on our requirement.
* **Reusability:** Encapsulation also improves the re-usability and easy to change with new requirements.
* **Testing code is easy:** Encapsulated code is easy to test for unit testing.

**Disadvantages of Encapsulation in Java**

The main disadvantage of the encapsulation in Java is it increases the length of the code and slow shutdown execution.

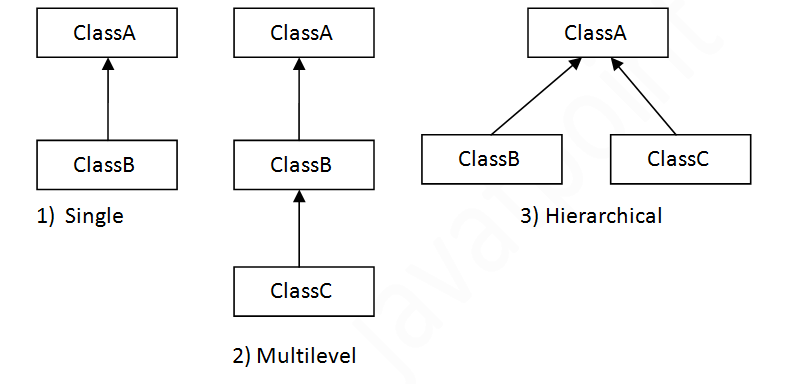
**Inheritance**

Inheritance is an important pillar of OOP(Object Oriented Programming). It is the mechanism in java by which one class is allow to inherit the features(fields and methods) of another class.

**Important terminology:**

* **Super Class:**The class whose features are inherited is known as super class(or a base class or a parent class).
* **Sub Class:** The class that inherits the other class is known as sub class(or a derived class, extended class, or child class). The subclass can add its own fields and methods in addition to the superclass fields and methods.
* **Reusability:**Inheritance supports the concept of “reusability”, i.e. when we want to create a new class and there is already a class that includes some of the code that we want, we can derive our new class from the existing class. By doing this, we are reusing the fields and methods of the existing class.

On the basis of class, there can be three types of inheritance in java: single, multilevel and hierarchical.



**Advantages**

* One of the key benefits of inheritance is to minimize the amount of duplicate code in an application by sharing common code amongst several subclasses. Where equivalent code exists in two related classes, the hierarchy can usually be refactored to move the common code up to a mutual superclass. This also tends to result in a better organization of code and smaller, simpler compilation units.
* Inheritance can also make application code more flexible to change because classes that inherit from a common superclass can be used interchangeably. If the return type of a method is superclass
* **Reusability** - facility to use public methods of base class without rewriting the same.
* **Extensibility** - extending the base class logic as per business logic of the derived class.
* **Data hiding** - base class can decide to keep some data private so that it cannot be altered by the derived class
* **Overriding**-With inheritance, we will be able to override the methods of the base class so that meaningful implementation of the base class method can be designed in the derived class.

**Disadvantages**

* One of the main disadvantages of inheritance in Java (the same in other object-oriented languages) is the increased time/effort it takes the program to jump through all the levels of overloaded classes. If a given class has ten levels of abstraction above it, then it will essentially take ten jumps to run through a function defined in each of those classes
* Main disadvantage of using inheritance is that the two classes (base and inherited class) get tightly coupled. This means one cannot be used independent of each other.
* Also with time, during maintenance adding new features both base as well as derived classes are required to be changed. If a method signature is changed then we will be affected in both cases (inheritance & composition)
* If a method is deleted in the "base class" or aggregate, then we will have to re-factor in case of using that method. Here things can get a bit complicated in case of inheritance because our programs will still compile, but the methods of the subclass will no longer be overriding base class methods. These methods will become independent methods in their own right.

**Polymorphism**

Polymorphism in Java is a concept by which we can perform a single action in different ways. Polymorphism is derived from 2 Greek words: poly and morphs. The word "poly" means many and "morphs" means forms. So polymorphism means many forms.

In Java polymorphism is mainly divided into two types:

* Compile time Polymorphism
* Runtime Polymorphism

1. **Compile time polymorphism**

 It is also known as static polymorphism. This type of polymorphism is achieved by function overloading or operator overloading.

* **Method Overloading**: 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 **change in type of arguments**.
* **Operator Overloading**: Java 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.

In java, Only “+” operator can be overloaded:

* To add integers
* To concatenate strings

1. **Run time polymorphism**

It is also known as Dynamic Method Dispatch. It is a process in which a function call to the overridden method is resolved at Runtime. This type of polymorphism is achieved by Method Overriding.

* [Method overriding](https://www.geeksforgeeks.org/overriding-in-java/), 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**.

**Advantages of polymorphism**

**1.** Method overloading allows methods that perform similar or closely related functions to be accessed through a common name. For example, a program performs operations on an array of numbers which can be int, float, or double type. Method overloading allows you to define three methods with the same name and different types of parameters to handle the array operations.

**2.** Method overloading can be implemented on constructors allowing different ways to initialize objects of a class. This enables you to define multiple constructors for handling different types of initializations.

**3**. Method overriding allows a sub class to use all the general definitions that a super class provides and add specialized definitions through overridden methods.

**4.** Method overriding works together with inheritance to enable code reuse of existing classes without the need for re-compilation.

**Disadvantages of Polymorphism**

* One of the disadvantages of polymorphism is that developers find it difficult to implement polymorphism in codes.
* Run time polymorphism can lead to the performance issue as machine needs to decide which method or variable to invoke so it basically degrades the performances as decisions are taken at run time.
* Polymorphism reduces the readability of the program. One needs to identify the runtime behavior of the program to identify actual execution time.

**Abstraction**

Data Abstraction is the property by virtue of which only the essential details are displayed to the user. The trivial or the non-essentials units are not displayed to the user. Ex: A car is viewed as a car rather than its individual components.

Data Abstraction may also be defined as the process of identifying only the required characteristics of an object ignoring the irrelevant details. The properties and behaviors of an object differentiate it from other objects of similar type and also help in classifying/grouping the objects.

In java, abstraction is achieved by interfaces and abstract classes. We can achieve 100% abstraction using interfaces.

**Abstract classes and Abstract methods :**

1. An abstract class is a class that is declared with [abstract keyword.](https://www.geeksforgeeks.org/abstract-keyword-in-java/)
2. An abstract method is a method that is declared without an implementation.
3. An abstract class may or may not have all abstract methods. Some of them can be concrete methods
4. A method defined abstract must always be redefined in the subclass,thus making [overriding](http://contribute.geeksforgeeks.org/overriding-in-java/) compulsory OR either make subclass itself abstract.
5. Any class that contains one or more abstract methods must also be declared with abstract keyword.
6. There can be no object of an abstract class. That is, an abstract class can not be directly instantiated with the [new operator](https://www.geeksforgeeks.org/new-operator-java/).
7. An abstract class can have parametrized constructors and default constructor is always present in an abstract class.

**Advantages of Abstraction**

1. It reduces the complexity of viewing the things.
2. Avoids code duplication and increases reusability.
3. Helps to increase security of an application or program as only important details are provided to the user.