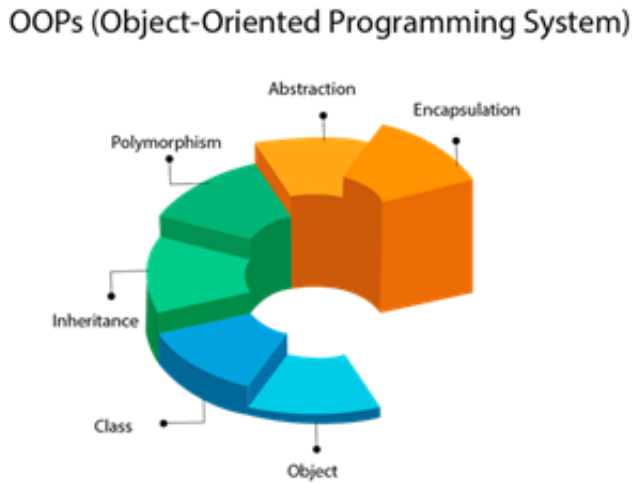
OOPS

Oops

Object-oriented programming (OOP) is a programming paradigm that uses “Objects “and their interactions to design applications.

- Classes
- Objects
- Abstraction
- Encapsulation
- Inheritance
- Polymorphism



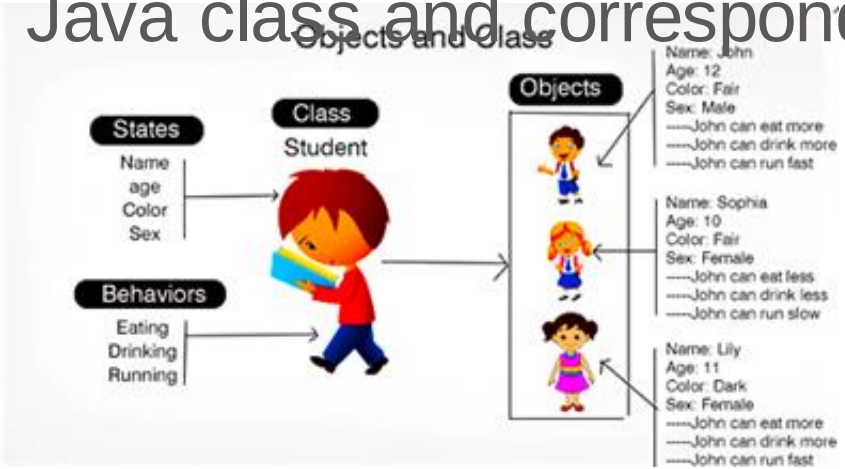
Classes

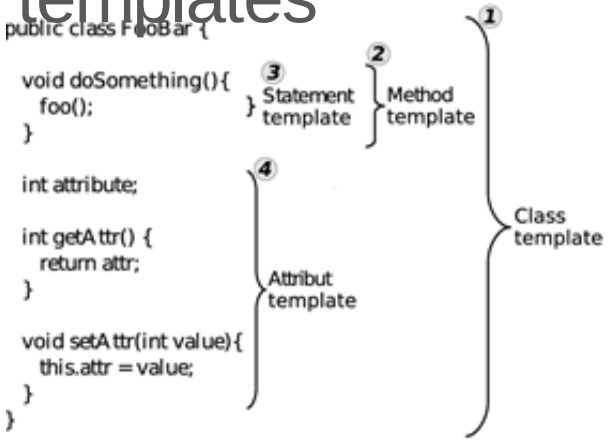
- **Class** is blueprint or an idea of an Object
- Class is the group of objects exhibiting same behavior will come under the same group.
- **Classes** are data types based on which objects are created. Objects with similar properties and methods are grouped together to form a Class. Thus a Class represents a set of individual objects.
- Characteristics of an object are represented in a class as **Properties**.
- The actions that can be performed by objects become functions of the class and is referred to as **Methods**.

Objects

- Any entity that has state and behavior is known as an object.
- **Object** is an instance of a class which is an entity with its own attribute values and methods.
- The object of a class can be created by using the new keyword in Java Programming language.
- Creating an Instance
  - ClassName refVariable = new Constructor();

Java class and corresponding templates





Abstraction

- **Abstraction** refers to the act of representing essential features without including the background details or explanations
- Abstraction is a process which displays only the information needed and hides the unnecessary information.
- We can say that the main purpose of abstraction is data hiding.
- Abstraction means selecting data from a large number of data to show the information needed, which helps in reducing programming complexity and efforts.

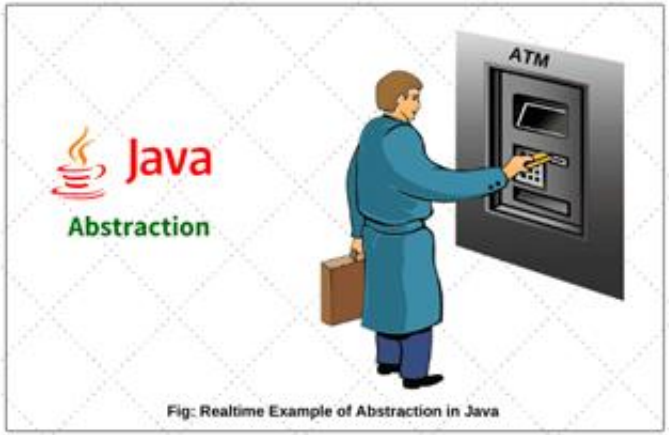
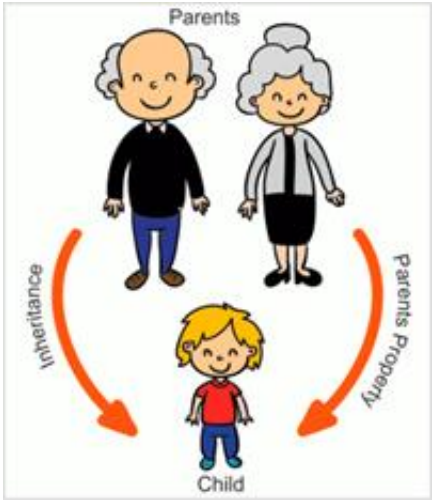
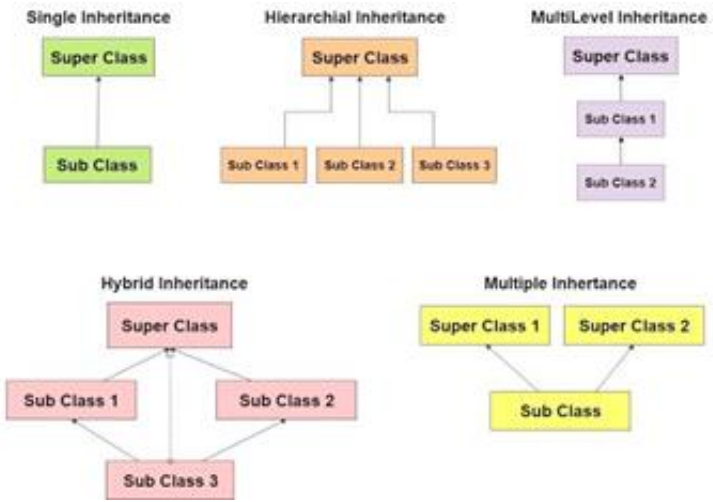


Fig: Realtime Example of Abstraction in Java

Inheritance

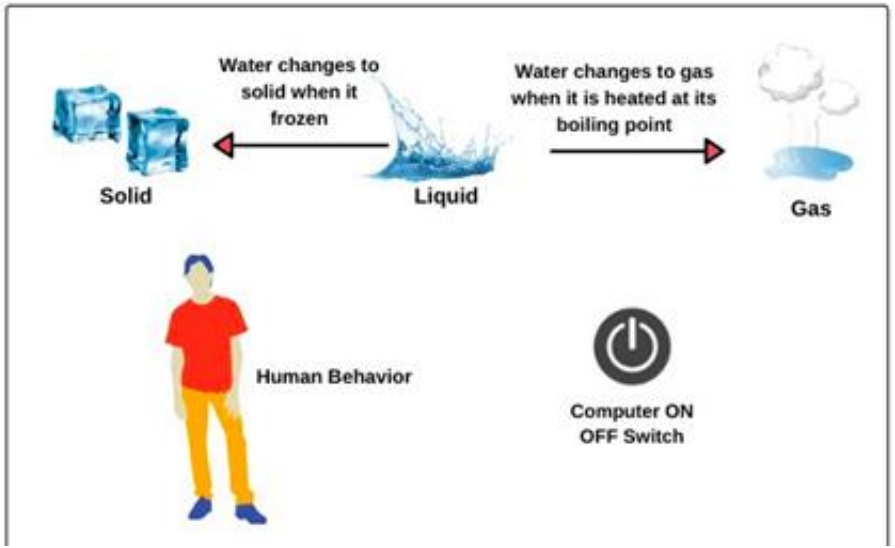
- **Inheritance** allows to reuse classes by deriving a new class from an existing one.
- The existing class is called the parent class, or superclass, or base class.
- The derived class is called the child class or subclass.
- The child class inherits characteristics of the parent class(i.e the child class inherits the methods and data defined for the parent class.





Polymorphism

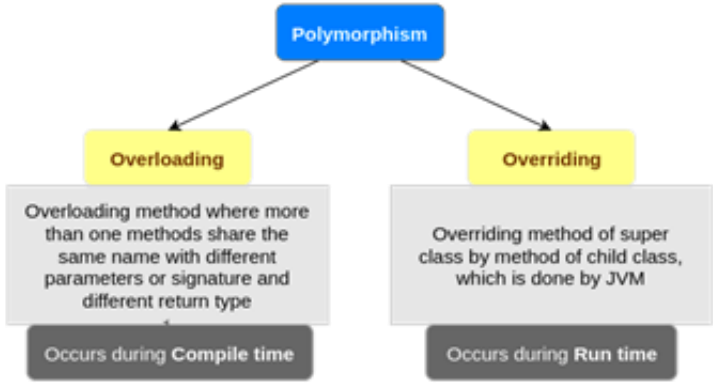
- The word **polymorphism** means having many forms. In simple words, we can define polymorphism as the ability of a message to be displayed in more than one form.
- Polymorphism is the concept where an object behaves differently in different situations.
- In Java polymorphism is mainly divided into two types:
  - Compile-time Polymorphism
  - Runtime Polymorphism



Types of Polymorphism

Method Overloading

- When there are multiple functions with the same name but different parameters then these functions are said to be overloaded. Functions can be overloaded by change in the number of arguments or/and a change in the type of arguments.



Method overriding

- It 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.

ENCAPSULATION

- Encapsulation is a process of wrapping code and data together into a single unit.
- The process that binds together the data and code into a single unit and keeps both from being safe from outside interference and misuse.
- In this process, the data is hidden from other classes and can be accessed only through the current class's methods. Hence, it is also known as data hiding
- Encapsulation acts as a protective wrapper that prevents the code and data from being accessed by outsiders. These are controlled through a well-defined interface.

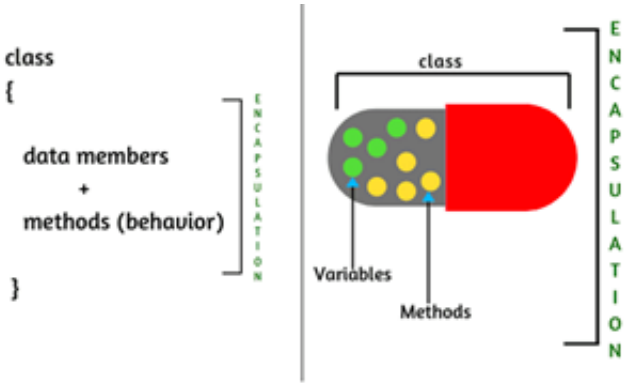


Fig: Encapsulation

Advantage/Benefit of OOPs

- **Security:** In OOP, Data is encapsulated with methods in the class so that data is protected and secured from accidental modification by other external non-member methods.
- **Reusability:** Through inheritance, we can use the features of an existing class in a new class without repeating existing code that saves a lot of time for developers, and also increases productivity.
- **Effective communication:** In OOP, objects can communicate via message passing technique that makes interface descriptions with outside systems much simpler.
- **Developing complex software:** OOPs is the most suitable approach for developing complex software because it minimizes the complexity through the feature of inheritance.
- **Easily upgraded:** Object-oriented system can be easily upgraded from small to large systems because OOP uses bottom-up approach.
- **Easy partition of work:** It is easy to partition complicated work in a project based on objects.
- **Maintenance:** The maintenance of object-oriented code is easier.
- **Efficiency:** The concepts of OOP provide better efficiency and an easy development process.