Abstraction lets you focus on what the [object](https://www.javatpoint.com/object-and-class-in-java) does instead of how it does it.

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

***Abstract Class:***

* A class that is declared using “**abstract**” keyword is known as abstract class.
* An abstract class can have ***abstract methods***(methods without body) as well as ***concrete methods*** (regular methods with body).
* An abstract class can not be **instantiated**, which means you are not allowed to create an **object** of it.
* An abstract class must be declared with an abstract keyword.
* It can have abstract and non-abstract methods.
* It cannot be instantiated.
* It can have [constructors](https://www.javatpoint.com/java-constructor) and static methods also.
* It can have final methods which will force the subclass not to change the body of the method.

***Why we need an abstract class?***

* Lets say we have a class Animal that has a method sound() and the subclasses of it like Dog, Lion, Horse, Cat etc.
* So when we know that all the animal child classes **will and should override** this method, then there is no point to implement this method in parent class. Thus, making this method abstract would be the good choice as by making this method abstract **we force all the sub classes to implement this method**( otherwise you will get compilation error), also we need not to give any implementation to this method in parent class.

***Abstract Class Example:***

abstract class Animal{

//abstract method

public abstract void sound();

}

//Dog class extends Animal class

public class Dog extends Animal{

public void sound(){

System.out.println("Woof");

}

public static void main(String args[]){

Animal obj = new Dog();

obj.sound();

}

}

There are cases when it is difficult or often unnecessary to implement all the methods in parent class. In these cases, we can declare the parent class as abstract, which makes it a special class which is not complete on its own.

If a child does not implement all the abstract methods of abstract parent class, then the child class must need to be declared abstract as well.

 Interface provides full abstraction as none of its methods have body.

While providing implementation in class of any method of an interface, it needs to be mentioned as public.

Class that implements any interface must implement all the methods of that interface, else the class should be declared abstract.

All the interface methods are by default **abstract and public**.

Variables declared in interface are **public, static and final** by default.

Interface variables must be initialized at the time of declaration otherwise compiler will throw an error.

A **class** can implement any **number of interfaces**.

13) If there are **two or more same methods** in two interfaces and a class implements both interfaces, implementation of the method once is enough.

A class cannot implement two interfaces that have methods with same name but different return type.

Variable names conflicts can be resolved by interface name.

interface A

{

int x=10;

}

interface B

{

int x=100;

}

class Hello implements A,B

{

public static void Main(String args[])

{

/\* reference to x is ambiguous both variables are x

\* so we are using interface name to resolve the

\* variable

\*/

System.out.println(x);

System.out.println(A.x);

System.out.println(B.x);

}

}

public abstract class Employee {

private String name;

private String address;

private int number;

public Employee(String name, String address, int number) {

System.out.println("Constructing an Employee");

this.name = name;

this.address = address;

this.number = number;

}

public double computePay() {

System.out.println("Inside Employee computePay");

return 0.0;

}

public void mailCheck() {

System.out.println("Mailing a check to " + this.name + " " + this.address);

}

public String toString() {

return name + " " + address + " " + number;

}

public String getName() {

return name;

}

public String getAddress() {

return address;

}

public void setAddress(String newAddress) {

address = newAddress;

}

public int getNumber() {

return number;

}

}