INTERFACES

- Interface is a concept which is used to achieve abstraction in Java.
- we can achieve full abstraction using interfaces.
- Interfaces are syntactically similar to classes, but you cannot create instance of an Interface and their methods are declared without any body.

Syntax:

```
interface interface_name
{
// public, static & final variables
//  methods with no body
}
```

- It can have only abstract methods i.e no concrete methods are allowed
- It Support multiple inheritance
- Methods inside interface must not be static, final
- All variables declared inside interface are implicitly public, static and final.
- All methods declared inside interfaces are implicitly public and abstract, even if you don't use public or abstract keyword.
- Interface can extend one or more other interface.
- Interface cannot implement a class.

- Interface can be nested inside another interface.
- An interface is implemented by a class.
- The class that implements an interface must provide code for all the methods defined in the interface, otherwise, it must be defined as an abstract class.
- The class uses a keyword *implements* to implement an interface.
- A class can implement any number of interfaces.
- When a class wants to implement more than one interface, we use the *implements* keyword followed by a comma-separated list of the interfaces implemented by the class.

class *className* implements *InterfaceName*{

...

boby-of-the-class

...

}

//DEFINING AND IMPLEMENTING AN INTERFACE

Ex:

```
interface A
{
  int var=25;
  void call();
}
class B implements A
{
```

```
  public void call()
{
 System.out.println("last day of the month");
}}
class InterfaceDemo1
{
 public static void main(String args[])
{
B ob=new B();
ob.call();
}}
```

Partial Implementation of Interfaces

- The class that implements an interface must provide code for all the methods defined in the interface, otherwise, it must be defined as an abstract class.

```
interface A
{
  void call1();
  void call2();
  void call3();
}
class B implements A
{
  public void call1()
{
  System.out.println("last day of the month");
}}
class PartialImpInterface
```

```
{
 public static void main(String args[])
{
B ob=new B();
ob.call();
}}
```

- ➔ The above program will give an error as the child class B is not implementing all the methods of parent interface A.
- ➔ So we should make B abstract as it is getting incomplete methods from its parent.
- ➔ And then we should write a child class of B which gives the complete definition of all the remaining methods.

```
interface A
{
  void call1();
  void call2();
  void call3();
}
abstract class B implements A
{
  public void call1()
```

```
{
  System.out.println("last day of the month");
}}
class C extends B
{
 public void call2()
{
System.out.println("April");
}
public void call3()
{
System.out.println("2021");
}
}
class PartialImpInterface
{
 public static void main(String args[])
{
C ob=new C();
ob.call1();
ob.call2();
```

ob.call3();}}

variables in interfaces

➔ Variables in interface can be used as constant values directly in if condition and also in switch cases.

```
interface A
{
  int YES=1;
  int NO=0;
}
class B implements A
{
  int check(int m)
  {
     if (m<18)
       return NO;
     else
       return YES;
  }
}
class C implements A
```

```
{
  void verify(int r)
{
   switch(r)
{
   case YES: System.out.println("Eligible");break;
   case NO: System.out.println("Not Eligible");break;
}}}
class InterfaceVarDemo
{
 public static void main(String args[])
{
   B ob1=new B();
   C ob2=new C();
   ob2.verify(ob1.check(15));
   ob2.verify(ob1.check(35));
}}
```

Extending Interfaces:

➔ an interface can extend another interface

```
interface A
{
  void call1();
```

```
}
interface B extends A
{
  void call2();
}
class C implements B
{
 public void call1()
{
 System.out.println("from A");
}
 public void call2()
{
 System.out.println("from B");
}
}
class InterfaceExtendDemo
{
public static void main(String args[])
{
 C ob=new C();
  ob.call1();
  ob.call2();
}}
```

MULTIPLE INHERITANCE USING INTERFACES

- ➔ Multiple inheritance in java is not possible if both the parents are classes
- ➔ It is possible if one of the parent or both parents are interfaces.

```
interface A
{
  void call1();
```

```
}
class B
{
    void call2()
{
 System.out.println("hi");
}}
class C extends B implements A
{
 public void call1()
{
 System.out.println("hello");


}}
class MultipleInhDemo1
{
 public static void main(String args[])
{
 C ob=new C();
 ob.call1();
 ob.call2();
}}
```

```
 //ANOTHER FORM OF MULTIPLE INHERITANCE
interface A
{
   void meth1();
}
interface B
{
 void  meth2();
}
class C implements A,B
{
public void meth1()
{
  System.out.println(“good”);
}
public void meth2()
{
System.out.println(“afternoon”);
}
}
class MultipleInheritanceDemo
{
public static void main(String args[])
{
```

```
C ob=new C();
ob.meth1();
ob.meth2();
}}
```