**Chapter 6**

**Abstract class & Interface**

**Abstract class**

A class which contains the abstract keyword in its declaration is known as abstract class.

* Abstract classes may or may not contain abstract methods, i.e., methods without body ( public void get(); )
* But, if a class has at least one abstract method, then the class must be declared abstract.
* If a class is declared abstract, it cannot be instantiated.
* To use an abstract class, you have to inherit it from another class, provide implementations to the abstract methods in it.
* If you inherit an abstract class, you have to provide implementations to all the abstract methods in it.



**Interface**

An **interface in Java** is a blueprint of a class. It has static constants and abstract methods.

The interface in Java is *a mechanism to achieve* [abstraction](https://www.javatpoint.com/abstract-class-in-java). There can be only abstract methods in the Java interface, not method body. It is used to achieve abstraction and multiple [inheritance in Java](https://www.javatpoint.com/inheritance-in-java).

In other words, you can say that interfaces can have abstract methods and variables. It cannot have a method body.



**Difference between abstract class and interface**



**Implementaion of multiple inheritance through interface**

An interface contains variables and methods like a class but the methods in an interface are abstract by default unlike a class. Multiple inheritance by interface occurs if a class implements multiple interfaces or also if an interface itself extends multiple interfaces.

A program that demonstrates multiple inheritance by interface in Java is given as follows:

## Example

interface AnimalEat {

   void eat();

}

interface AnimalTravel {

   void travel();

}

class Animal implements AnimalEat, AnimalTravel {

   public void eat() {

      System.out.println("Animal is eating");

   }

   public void travel() {

      System.out.println("Animal is travelling");

   }

}

public class Demo {

   public static void main(String args[]) {

      Animal a = new Animal();

      a.eat();

      a.travel();

   }

}

## Output

Animal is eating

Animal is travelling

Now let us understand the above program.

The interface AnimalEat and AnimalTravel have one abstract method each i.e. eat() and travel(). The class Animal implements the interfaces AnimalEat and AnimalTravel. A code snippet which demonstrates this is as follows:

interface AnimalEat {

   void eat();

}

interface AnimalTravel {

   void travel();

}

class Animal implements AnimalEat, AnimalTravel {

   public void eat() {

      System.out.println("Animal is eating");

   }

   public void travel() {

      System.out.println("Animal is travelling");

   }

}

In the method main() in class Demo, an object a of class Animal is created. Then the methods eat() and travel() are called. A code snippet which demonstrates this is as follows:

public class Demo {

   public static void main(String args[]) {

      Animal a = new Animal();

      a.eat();

      a.travel();

   }

}