abstract

Abstraction:

Abstraction is the concept of hiding the Field or Method.

Concrete Class	Concrete Method
Abstract Class	Abstract Method
Interface	

Data Abstraction / Data Hiding:

Data Abstraction is achieved by using the *'private'* access modifier.

Parent.java

```
class Parent

{

        private int a = 10;

}
```

Child.java

```
class Child extends Parent

{

        public static void main(String... args)

        {

                Child c = new Child();


                System.out.println(c.a);

        }

}
```

Output:

```
error: a has private access in Parent
```

Method Abstraction / Method Hiding: (Implementation Hiding)

Method Abstraction is achieved by using the keyword *'abstact'*.

Concrete Method:

A Method which has it's body part is known as Concrete Method.

ConcreteMethod.java

```
class Demo
{
		void fun()
		{
			System.out.println("This is a Concrete Method");
		}

		public static void main(String... args)
		{
			Demo d = new Demo();

			d.fun();
		}
}
```

Output:

```
This is a Concrete Method
```

Concrete Class:

A class in which all methods are Concrete Methods is known as Concrete Class.

ConcreteClass.java

```
class Demo
{
		void fun()
		{
			System.out.println("This is a Concrete Method 'fun' ");
		}

		void gun()
		{
			System.out.println("This is a Concrete Method 'gun' ");
```

```
	}

	void sun()
	{
		System.out.println("This is a Concrete Method 'sun' ");
	}

	public static void main(String... args)
	{
		Demo d = new Demo();

		d.fun();
		d.gun();
		d.sun();
	}
}
```

Output:

```
This is a Concrete Method 'fun'

This is a Concrete Method 'gun'

This is a Concrete Method 'sun'
```

Abstract Method:

A method which is only declared and not defined is known as an Abstract Method.

We can declare a method using the keyword *‘abstact’*.

But we must declare the class as ‘abstract’ in which we are declaring the Abstract Method.

We cannot create the object of an Abstract Class because if we could, it’d be possible to access the Abstract Method using this object and as there is no body part for Abstract Methods, it generates an error!

Parent.java

```
abstract class Parent

{

	abstract void fun();

}
```

Now, This Abstract Methods declared in a Parent class can be defined inside the Child class inherited from the Parent Class, which tends to overriding.

Then we can call this Abstract Methods by using the object of the Child Class.

Child.java

```
class Child extends Parent
{
	void fun()
	{
		System.out.println("This is an Abstract Method overridden in Child Class");
	}

	public static void main(String... args)
	{
		Child c = new Child();

		c.fun();
	}
}
```

Output:

```
This is an Abstract Method overridden in Child Class
```

If we declare an Abstract Method inside a Concrete class, it throws a Compile Time Error!

i.e. Abstract Methods inside Concrete Classes are not allowed.

Because if we do so, then we can create the object of this class and using this access it’d possible to access the Abstract Method which is not allowed!

Demo.java

```
class Demo
{
	abstract void fun();
}
```

Output:

```
error: Demo is not abstract and does not override abstract method fun() in Demo
```

Abstract Methods cannot be static.

Demo.java

```
abstract class Demo

{

            static abstract void fun();

}
```

Output:

```
error: illegal combination of modifiers: abstract and static
```

An Abstract Class can contain:

All Abstract Methods.

OR

Some Concrete Methods - Some Abstract Methods

OR

All Concrete Methods

Parent.java

```
abstract class Parent

{

            abstract void fun();

            abstract void gun();

            abstract void sun();

}
```

Child.java

```
abstract class Child extends Parent
{
	void fun()
	{
		System.out.println("This is an Abstract Method 'fun' ");
	}

	void gun()
	{
		System.out.println("This is an Abstract Method 'gun' ");
	}
}
```

GrandChild.java

```
abstract class GrandChild extends Child
{
	void sun()
	{
		System.out.println(“This is an Abstract Method ‘sun’ “);
	}
}
```

Demo.java

```
class Demo extends GrandChild
{
	public static void main(String... args)
	{
		Demo d = new Demo();

		d.fun();
		d.gun();
		d.sun();
	}
}
```

Output:

```
This is an Abstract Method 'fun'

This is an Abstract Method 'gun'

This is an Abstract Method 'sun'
```

Here, the Parent Class contains all Abstract Methods, Child Class contains some Concrete Methods and some Abstract Methods and the GrandChild Class contains all Concrete Methods.

Constructor of an Abstract Class is allowed.

Parent.java

```
abstract class Parent
{
	abstract void fun();


	Parent()
	{
		System.out.println("This is the Constructor inside an Abstract Class 'Parent' ");
	}
}
```

Child.java

```
class Child extends Parent
{
	void fun()
	{
		System.out.println("This is an Abstract Method overridden in Child Class");
	}

```

```
        Child()
        {
                System.out.println("This is the Constructor of Class 'Child' ");
        }

        public static void main(String... args)
        {
                Child c = new Child();

                c.fun();
        }
}
```

Output:

```
This is the Constructor inside an Abstract Class 'Parent'

This is the Constructor of Class 'Child'

This is an Abstract Method overridden in Child Class
```