## ABSTRACTION

It is an OOP **technique** that hides certain details and only shows the important information.

It can be achieved in 2 ways:

- **Abstract Classes (Uses abstract modifier)**
- **Interfaces (Uses implements keyword)**

## ABSTRACT Class

It cannot be **instantiated**, to access it you need to inherit it from another class.

## ABSTRACT Methods

**Can only** be **declared inside** an **abstract class**, It is a **Method without** a **body** and it **needs** to be **overridden** in the **subclass** of the **abstract class**.

## USING Abstraction (Abstract Classes)

```
abstract class Animal
        {
            abstract void makeSound();
         }

 class Dog extends Animal
   {
            void makeSound()
```

```
                {
                 //Arf
                 }
        }

    class Cat extends Animal
        {
                void makeSound()
                    {
                     //Meow
                     }
        }
```

**IMPLEMENTS Keyword**

**Used after** the **class name** so that we can **implement** an **interface** in that **certain class**.

**Implemented interface requires** the **class** to **override** every **method** inside the **interface**.

**PS** We can **implement 1** or **more interfaces** in **each class**.

**INTERFACE**

It is full Abstract Class that is implemented in other classes.

Any **method** you declare would be an **abstract method**, by default you can use the **default** modifier to create a method with body. The method would be **static**.

Any **variable** that you declare would be **static** and **final**.

**USING Abstraction (Interfaces)**

```
Interface Animal
        {
          void makeSound();
         }

 class Dog implements Animal
   {
          void makeSound()
              {
               //Arf
               }
    }

class Cat implements Animal
   {
          void makeSound()
              {
               //Meow
```