Abstraction: Terms like abstraction and generalization are used when we try to model real world things in software. When we start to model objects for our application, we identify common features and behaviors that objects have, so we generalize them. We create a class hierarchy; the base class is the most general class which has common things for objects.

- Part of generalization is using abstraction. For example, if we consider Octopus, Dog and Penguin, you'd probably say all these are animals. Here Animal is an abstract concept which doesn't exist, except you can use it to describe a set of specific things. Here we describe a set of traits and behavior and group them into a particular class based on their functionality.
- Similarly, think of an example car. It has sets of thousands of individual parts, however we think of it as a well-defined object with its own unique behavior.
- As a thumb rule, we manage complexity through abstraction. We hide all the implementation details under an abstract class which has its own unique behavior.

Java supports abstraction in several ways.

- It allows us to create class hierarchy where the top of hierarchy which is base class is usually an abstract concept whether it is an abstract class or not. Java lets us create abstract classes.
- Java lets us also create abstract methods.
    - An abstract method has a method signature and a written type but doesn't have a method body. In a way, we can say that abstract method is unimplemented.
    - The purpose of creating an abstract method is to describe behaviour which any object of that type will always have.

Abstract classes and interfaces can have a mix of both concrete and abstract methods.

- Concrete method: A concrete method has a method body with usually at least one statement. It has an operational code that is executed under the right conditions. It is said that a concrete method implements an abstract method if it overrides 1.

In addition to access modifiers (Public, private, protected), we also have method modifiers.

Modifier	Purpose
abstract	When you declare a method abstract, a method body is always omitted. An abstract method can only be declared on an abstract class or an interface.
static	Sometimes called a class method, rather than an instance method, because it's called directly on the Class instance.
final	A method that is final cannot be overridden by subclasses.
default	This modifier is only applicable to an interface, and we'll talk about it in our interface videos.
native	This is another method with no body, but it's very different from the abstract modifier. The method body will be implemented in platform-dependent code, typically written in another programming language such as C. This is an advanced topic and not generally commonly used
synchronized	This modifier manages how multiple threads will access the code in this method.

Abstract Class:

- An abstract class is declared with abstract modifier.

```
abstract class Animal {}    // An abstract class is declared with the
                            // modifier.
```

- We can say an abstract class is incomplete, hence we can't create instance of an abstract class.

```
Animal a = new Animal();  // INVALID, an abstract class never gets instantiated
```

- However, an abstract class can still have a constructor which will be called by its subclasses during their construction.

- **An abstract class** purpose is to define the behavior its subclasses are required to have, so it always participates in inheritance.
- Classes extending abstract classes can be an abstract class or a concrete class. Similarly, classes extending concrete class can be a concrete class or an abstract class.

Abstract method: Similar to abstract class, abstract method is also declared with modifier abstract, As mentioned, it can have a written type, but it always ends with semi colon; It doesn't have a body, not even curly braces.

Note: An abstract class can only be declared on an abstract class or interface. It cannot be declared on a concrete class.

Why do we need an abstract class? What good is a class without any code in it?

- Abstract classes define method signatures without providing implementation.

An abstract class in the class hierarchy forces the designers of subclasses to think about and create unique and targeted implementation for abstract methods. Since an abstract class can't be instantiated, so if you are using abstract classes to design a framework for implementation, this is definitely an advantage..

- Subclasses of abstract methods must either implement all abstract methods or be declared abstract themselves. If a subclass failed to implement all abstract methods of its superclass, a compilation error occurs.
- This ensures that all subclasses adhere to the contract defined by the abstract class, guaranteeing the availability of required behavior in all subclasses.
- Abstraction helps in defining common behaviour across related classes while hiding the implementation details. Additionally, abstract classes can also achieve polymorphism, where objects of different concrete subclasses can be treated uniformly through their common abstract superclass.