Abstract Classes in Java

Definition:

An abstract class in Java is a class that cannot be instantiated (i.e., you cannot create objects from it) and is used as a base class for other classes. It is declared using the keyword abstract. An abstract class can have both abstract methods (without a body) and concrete methods (with a body).

Syntax:

```
abstract class Animal {
   abstract void sound();  // Abstract method (no body)

   void sleep() {         // Concrete method
      System.out.println("Sleeping...");
   }
}
```

Key Points:

- Abstract class can have constructors and static methods.
- It can contain fields (variables), methods (both abstract and non-abstract).
- A class extending an abstract class must implement all its abstract methods.
- Abstract classes are used when you want to provide common functionality for all subclasses, but also force certain methods to be implemented.

Example:

```
abstract class Animal {
   abstract void sound(); // abstract method

   void eat() {
      System.out.println("This animal eats food.");
   }
}

class Dog extends Animal {
   void sound() {
      System.out.println("Dog barks");
   }
}
```

```
public class Main {
   public static void main(String[] args) {
      Dog d = new Dog();
      d.sound();  // Output: Dog barks
      d.eat();    // Output: This animal eats food.
   }
}
```

Why Use Abstract Classes?

- To provide a template or blueprint for other classes.
- To ensure certain methods are overridden in subclasses.
- To enable code reusability by writing common code in the abstract class.