Abstraction in Java

Why Do We Use Abstraction and the `abstract` Keyword in Java?

1. To Hide Complexity and Show Only Essentials

Abstraction hides complex implementation and shows only important features.

Example:

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

2. To Provide a Common Template (Contract)

Abstract classes define a contract that subclasses must follow.

Example:

```
abstract class Shape {
    abstract void draw();
}
```

3. To Support Partial Implementation

Abstract classes can have both abstract and non-abstract methods.

Example:

```
abstract class Vehicle {
    abstract void start();
```

```
    void fuel() {
        System.out.println("Needs petrol or diesel");
    }
}
```

4. To Achieve Runtime Polymorphism

You can refer to subclass objects using abstract class reference.

Example:

```
Animal a = new Dog();
a.makeSound(); // Calls Dog's version
```

Why We Use the `abstract` Keyword:

Use	Description
abstract class	Declares a class that can't be instantiated
abstract void method();	Declares a method to be implemented by subclass

When Should You Use Abstraction?

- **To define a common blueprint**
- **To hide internal logic**
- **To enforce mandatory methods in subclasses**
- **To provide partial implementation**

Simple Example:

```
abstract class Animal {
    abstract void sound();
    void sleep() {
        System.out.println("Animal sleeps");
    }
}

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

Final Summary Table:

Concept	Purpose
abstract class	Shared structure with some behavior
abstract method	Must be implemented by subclass
Abstraction	Hides internal details and shows only necessary info