**Abstraction in Java**

**Abstraction in Java** is the process of hiding the implementation details and only showing the essential functionality or features to the user. This helps simplify the system by focusing on what an object does rather than how it does it. The unnecessary details or complexities are not displayed to the user.

**Television remote control**is an excellent **example**of **abstraction**. It simplifies the interaction with a TV by hiding the complexity behind simple buttons and symbols, making it easy without needing to understand the technical details of how the TV functions.

@Override

void turnOff() {

System.out.println("TV i// Demonstrating Abstraction in Java

abstract class Geeks {

abstract void turnOn();

abstract void turnOff();

}

// Concrete class implementing the abstract methods

class TVRemote extends Geeks {

@Override

void turnOn() {

System.out.println("TV is turned ON.");

}

s turned OFF.");

}

}

// Main class to demonstrate abstraction

public class Main {

public static void main(String[] args) {

Geeks remote = new TVRemote();

remote.turnOn();

remote.turnOff();

}

}

**Output**

TV is turned ON.

TV is turned OFF.

**Explanation**: In the above example, the “**Geeks**” abstract class hides implementation details and defines the essential methods **turnOn** and **turnOff**. The **TVRemote** class provides specific implementations for these methods. The main class demonstrates how the user interacts with the abstraction without needing to know the internal details.

**In**[**Java**](https://www.geeksforgeeks.org/java/)**, abstraction is achieved by**[**interfaces**](https://www.geeksforgeeks.org/interfaces-in-java/)**and**[**abstract classes**](https://www.geeksforgeeks.org/abstract-classes-in-java/)**.** We can achieve **100% abstraction using interfaces**. Data Abstraction may also be defined as the process of identifying only the required characteristics of an object ignoring the irrelevant details. The properties and behaviors of an object differentiate it from other objects of similar type and also help in classifying/grouping the objects.

**Abstraction Real-Life Example:**

*Consider a real-life example of a man driving a car. The man only knows that pressing the accelerator will increase the speed of a car or applying brakes will stop the car, but he does not know how on pressing the accelerator the speed is actually increasing, he does not know about the inner mechanism of the car or the implementation of the accelerator, brakes, etc. in the car. This is what abstraction is.*

**Abstract Classes and Abstract Methods**

* An abstract class is a class that is declared with an [abstract keyword.](https://www.geeksforgeeks.org/abstract-keyword-in-java/)
* An abstract method is a method that is declared without implementation.
* An abstract class may or may not have all abstract methods. Some of them can be concrete methods
* A abstract method must always be redefined in the subclass, thus making [overriding](https://www.geeksforgeeks.org/overriding-in-java/) compulsory or making the subclass itself abstract.
* Any class that contains one or more abstract methods must also be declared with an abstract keyword.
* There can be no object of an abstract class. That is, an abstract class can not be directly instantiated with the [new operator](https://www.geeksforgeeks.org/new-operator-java/).
* An abstract class can have parameterized constructors and the default constructor is always present in an abstract class.