**What is Data Abstraction?**

**Data Abstraction** is the process of **hiding implementation details** and exposing only the necessary parts of an object to the user. It helps to reduce complexity by **only showing relevant information** while keeping the actual implementation hidden.

In Java, **data abstraction is achieved using**:

1. **Abstract classes** (abstract keyword)
2. **Interfaces** (interface keyword)

**Example of Data Abstraction in Java**

Let's take a **Bank Account** example where the user can deposit and withdraw money **without knowing the internal implementation of these operations**.

// Abstract class providing Data Abstraction

abstract class BankAccount {

protected double balance; // Hidden data (Encapsulation)

// Constructor to initialize balance

public BankAccount(double balance) {

this.balance = balance;

}

// Abstract method (Hides implementation details of withdrawal)

abstract void withdraw(double amount);

// Abstract method (Hides implementation details of deposit)

abstract void deposit(double amount);

// Concrete method (Common functionality)

public double getBalance() {

return balance; // Users access balance only via this method

}

}

// Concrete class implementing the abstract methods

class SavingsAccount extends BankAccount {

public SavingsAccount(double balance) {

super(balance);

}

@Override

void withdraw(double amount) {

if (amount > 0 && amount <= balance) {

balance -= amount;

System.out.println("Withdrawn: " + amount);

} else {

System.out.println("Invalid withdrawal.");

}

}

@Override

void deposit(double amount) {

if (amount > 0) {

balance += amount;

System.out.println("Deposited: " + amount);

} else {

System.out.println("Invalid deposit.");

}

}

}

// Main class

public class DataAbstractionExample {

public static void main(String[] args) {

BankAccount myAccount = new SavingsAccount(5000);

myAccount.deposit(2000); // Deposited: 2000

myAccount.withdraw(1000); // Withdrawn: 1000

System.out.println("Current Balance: " + myAccount.getBalance()); // Current Balance: 6000

}

}

**Why is This Data Abstraction?**

1. **Hides Data:**
   * The balance variable is **protected** (not directly accessible).
2. **Hides Implementation Details:**
   * The **user calls deposit() and withdraw()** but **doesn’t know how they are implemented.**
3. **Only Shows Relevant Behavior:**
   * The user only needs to **interact with the account**, not worry about the internal calculations.

**Summary**

* **Data Abstraction** **hides implementation details** while allowing interaction through defined methods.
* Achieved using **abstract classes** and **interfaces** in Java.