# Difference Between Data Hiding and Abstraction in Java

Difficulty Level : Easy Last Updated : 06 Jun, 2021

***Abstraction*** Is hiding the internal implementation and just highlight the set of services. It is achieved by using the abstract class and interfaces and further implementing the same. Only necessarily characteristics of an object that differentiates it from all other objects. Only the important details are emphasized and the rest all are suppressed from the user or reader.

**A real-life example of abstraction**

> *By using ATM GUI screen bank people are highlighting the set of services what the bank is offering without highlighting internal implementation.*

**Types of Abstraction:** There are basically three types of abstraction

1. Procedural Abstraction
2. Data Abstraction
3. Control Abstraction

**1. Procedural Abstraction:** From the word itself, there are a series of procedures in form of functions followed by one after another in sequence to attain abstraction through classes.

**2. Data Abstraction:** From the word itself, abstraction is achieved from a set of data that is describing an object.

**3. Control Abstraction:** Abstraction is achieved in writing the program in such a way where object details are enclosed.

**Advantages of Abstraction:**

- Users or communities can achieve security as there are no highlights to internal implementation.
- The enhancement will become very easy because without affecting end users one is able to perform any type of changes in the internal system
- It provides more flexibility to end-user to use the system very easily

- It improves the richness of application

**Implementation of Abstraction**: It is implemented as a class which only represents the important traits without including background detailing. Providing only the necessary details and hiding all its internal implementation. Below is the java implementation of abstraction:

---

```
// Java program showing the working of abstraction

// Importing generic libraries
import java.io.*;

// Creating an abstract class
// demonstrate abstraction
abstract class Creature {

    // Just providing that creatures has legs
    // Hiding the number of legs
    abstract void No_Of_legs();
}

// A new child class is extending
// the parent abstract class above
class Elephant extends Creature {

    // Implementation of the abstract method
    void No_Of_legs()
    {

        // Printing message of function in non abstract
        // child class
        System.out.println("It has four legs");
    }
}

// Again a new child class is extended from parent
//  Human class to override function created above
class Human extends Creature {

    // Same function over-riden
    public void No_Of_legs()
    {

        // Message printed if this function is called or
        // Implementation of the abstract method
        System.out.println("It has two legs");
    }
}
```

```
public class GFG {

    // Main driver method
    public static void main(String[] args)
    {

        // Creating human object showing the implementation
        Human ob = new Human();

        ob.No_Of_legs();

        // Creating object of above class in  main
        Elephant ob1 = new Elephant();

        // Calling the function in main by
        // creating object of above non abstract class
        ob1.No_Of_legs();
        // Implementation of abstraction
    }
}
```

### Output

```
It has two legs
It has four legs
```

Now, jumping onto the second concept though both the concepts are used to achieve encapsulation somehow there is a sleek difference as shown below:

***Data Hiding*** is hiding internal data from outside users. The internal data should not go directly that is outside person/classes is not able to access internal data directly. It is achieved by using an access specifier- a private modifier.

**Note: T**he recommended modifier for data members is private. The main advantage of data hiding is security

**Sample for data hiding:**

*class Account {*

*private double account_balance;*

*……..*

*…….*

*}*

*Here account balance of each say employee is private to each other being working i n the same organization. No body knows account balance of anybody. In java it is a chieved by using a keyword 'private' keyword and the process is called data hiding.*

It is used as security such that no internal data will be accessed without authentication. An unauthorized end user will not get access to internal data. Programmatically we can implement data hiding by declaring data elements as private. Now to access this data or for modification, we have a special method known as getter setter respectively.

Getter is used to accessing the private data and setter is used to modify the private data only after authentication.In simple terms, it is hiding internal data from outside users.

It is used as security such that no internal data will be accessed without authentication. An unauthorized end user will not get access to internal data. Programmatically we can implement data hiding by declaring data elements as private.

***Now to access this data or for modification, we have a special method known as getter setter respectively***.

**Concept involved in data Hiding:** Getter and setter

Getter is used to accessing the private data and setter is used to modify the private data only after authentication. In simple terms, it is hiding internal data from outside users. It is used as security such that no internal data will be accessed without authentication. An unauthorized end user will not get access to internal data. Programmatically we can implement data hiding by declaring data elements as private. Now to access this data or for modification, we have a special method known as getter setter respectively.

Getter is used to accessing the private data and setter is used to modify the private data only after authentication. In simple terms, it is hiding internal data from outside users. It is used as security such that no internal data will be accessed without authentication. An unauthorized end user will not get access to internal data. Programmatically we can implement data hiding by declaring data elements as private.

Now to access this data or for modification, we have a special method known as getter setter respectively. Getter is used to accessing the private data and setter is used to modify the private data only after authentication.

**Implementation of Data Hiding:**

---

```
// Java Program showing working of data hiding

// Importing generic libraries
import java.io.*;

// Class created named Bank
class Bank {

    // Private data (data hiding)
```

```
    private long CurBalance = 0;

    // Bank_id is checked for authentication
    long bank_id;
    String name;

    // Getter function to modify private data
    public long get_balance(long Id)
    {

        // Checking whether the user is
        // authorised or unauthorised

        // Comparing bank_id of user and the give Id
        // then only it will get access
        if (this.bank_id == Id) {

            // Return current balance
            return CurBalance;
        }

        // Unauthorised user
        return -1;
    }
    // Setter function
    public void set_balance(long balance, long Id)
    {
        // Comparing bank_id of user and the give Id
        // then only it will get access
        if (this.bank_id == Id) {
            // Update balance in current ID
            CurBalance = CurBalance + balance;
        }
    }
}

// Another class created- Employee
public class Emp {
    public static void main(String[] args)
    {
        // Creating employee object of bank type
        Bank _emp = new Bank();

        // Assigning employee object values
        _emp.bank_id = 12345;
        _emp.name = "Roshan";

        // _emp.get_balance(123456)
        _emp.set_balance(10000, 12345);
        // This will no get access as bank_id is given wrong
        // so
        // unauthorised user is not getting access that is
        // data hiding

        long emp_balance = _emp.get_balance(12345);
        // As this time it is valid user it will get access
```

```
        // Display commands
        System.out.println("User Name"
                           + "   " + _emp.name);
        System.out.println("Bank_ID"
                           + "   " + _emp.bank_id);
        System.out.println("Current Balance"
                           + "   " + emp_balance);
    }
}
```

## Output:

```
User Name Roshan
Bank_ID 12345
Current Balance 10000
```