**Encapsulation(data hiding)**

**NOTE**:Capsule: multiple medicine into capsule

Encapsulation is the packing of data and function into a single component and hiding the internal implementation.

It is achieved by hiding the data by making the **variables private** and showing the functionality through **getter and setter methods.**

By providing only setter or getter method, you can make the class **read-only**(**getter) or write-only**(**setter**).

**Note**: To maintain privacy we use encapsulation

Example

**class** Encapsulation{

**private** **int** i;

**private** String str;

**void** set(**int** a, String stmt){ //setter method

i=a;

str=stmt;

}

**int** get\_i(){ //getter method

**return** i;

}

String get\_str(){ //getter method

**return** str;

}

}

**public** **class** Demo1 {

**public** **static** **void** main(String[] args) {

Encapsulation e = **new** Encapsulation();

e.set(11,"Java");

System.***out***.println("The value of i is "+e.get\_i());

System.***out***.println("The value of str is "+e.get\_str());

}

}

**Tightly Encapsulated Class**

When each variable is declared private in a particular class, it is commonly termed a “tightly encapsulated class” in Java. In such a case, programmers must check whether the concerned class contains the getter or setter methods and whether any of those methods are declared to be public or not.

**Advantages of Encapsulating data:**

* We can hide our data more efficiently. Hence, after implementation, user will not have any idea about the inner working of the class. To the user, only setting and initializing values is visible.
* It makes our data reusable.
* Encapsulated data is easy to test.

**Disadvantages of Encapsulating data:**

* Size of the code increases exponentially.
* As the size of the code increases, we need to provide additional implementation for each method.
* We provide additional methods, this increases code execution.