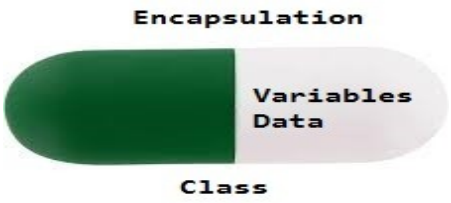
Encapsulation - It is defined as the wrapping up of data under a single unit. It is the mechanism that binds together code and the data it manipulates.

- Technically in encapsulation, the **variables** or **data** of a **class** is **hidden** from any other class and can be accessed only through any member function of own class in which they are declared.

- It is also known as combination of **data-hiding** and **abstraction** OR it is a way to achieve **data-hiding**.

- It can be achieved by: Declaring all the **variables** in the **class** as **private** and writing public methods in the class to set and get the values of variables.

- The **Java Bean** class is the example of a fully encapsulated class.



- Example:

```
// Java program to demonstrate encapsulation
public class Encapsulate  {
   // private variables declared these can only be accessed by public methods of class
   private String geekName;
   private int geekRoll;
   private int geekAge;

   public int getAge()   { // get method for age to access private variable geekAge
     return geekAge;
   }
   public String getName()  {   // get method for name to access private variable geekName
     return geekName;
   }
   public int getRoll()  { // get method for roll to access private variable geekRoll
      return geekRoll;
   }

   public void setAge( int newAge)   {  // set method for age to access private variable geekage
     geekAge = newAge;
   }
```

```
    // set method for name to access private variable geekName
    public void setName(String newName)   {
      geekName = newName;
    }

    // set method for roll to access private variable geekRoll
    public void setRoll( int newRoll)  {
      geekRoll = newRoll;
    }
}
```

Advantages of Encapsulation:

- Data Hiding: The user will have **no idea** about the **inner implementation** of the class. It will not be visible to the user that how the class is storing values in the variables. He only knows that we are passing the values to a setter method and variables are getting initialized with that value.

- Increased Flexibility: We can make the variables of the class as **read-only** (omit **setter** methods) or **write-only** (omit **getter** methods) depending on our requirement.

- Reusability: It improves the re-usability and is easy to change with new requirements.

- Testing code is easy: Encapsulated code is easy to test for unit testing.
  Unit Testing: It is a level of software testing where **individual units/ components** of a software are tested. The purpose is to validate that each unit of the software performs as designed. A **unit** is the **smallest testable part** of any software. It usually has one or a few inputs and usually a single output.