How to write setter and getter for a variable :

```
public class Student
{
  private int rollNumber;   //[Data Hiding]
  private String name;

  public void setRollNumber(int rollNumber)  //setter for rollNumber
  {
     this.rollNumber = rollNumber;
  }

  public int getRollNumber() //getter for rollNumber
  {
     return this.rollNumber;
  }

   public void setName(String name) //setter for name variable
   {
     this.name = name;
   }

   public String getName()  //getter for name
   {
      return this.name;
   }
}
```

*** What is Encapsulation in Java ?

[Accessing the private data via public methods like setter and getter]

In Medical Term we can say it is capsul

* Binding the data with its associated method in a single unit is called Encapsulation.
* Encapsulation ensures that **our private data must be accessible via public methods like setter and getter.**

```
public class Customer
  {
    private double customerBill;


    public void setCustomerBill(double  customerBill)
    {
       this.customerBill = customerBill;
    }

    public double getCustomerBill()
    {
       return this.cutsometBill;
    }
  }
```

ELC class Operation :

1) initalize the customer bill with user value
   [Parameterized Constructor]

2) To modify the existing bill amount
   [setter OR any public method]

3) To read the customerBill value outside of the BLC class
   [getter or Any public method]

4) To print customer bill
   [toString() method]

* It provides data security
* Without data hiding we cannot achieve encapsulation.
* Class is the example of encapsulation because we have data + method

How to achieve encapsulation in a class :

In order to achieve encapsulation we need to follow the following two steps

1) Declare all the fields with private access modifier. [Data Hiding]
2) Declare public methods like setter and getter for the non static variable in the class.

What is tightly encapsulated class and loosely encapsulated class ?

Tightly Encapsulated class :

If we declare all our variables with private access modifier then it is called Tightly encapsulated class

Example :

```
class Student
{
   private int roll;
   private String name;
   private String address;

   //Generate setter and getter for all the variables
}
```

Loosely encapsulated class :

If we declare our non static filed other than private access modifier then it is called loosely encapsulated class.

Example :

```
class Player
{
  private int id;
  protected String name;
  protected String address;
  //Generate setter and getter for all the variables
}
```

Method return type as a class :

While declaring a method in java, return type is compulsory.
As a method return type we have following options

1) void as a retutn type of the Method

2) Any primitive data type as a return type of the method.

3) Any class name/interface / enum / record we can take as a return type of the method.

```
public int m1()
{

  return 8;
}
```

Conclusion is : The return value **(8)** of the method must be compatible with return type **(int)** of the method

```
package com.ravi.encapsulation;

public class Test
{

   public Test m2()
   {
      //return null;

      //return new Test();

      Test t1 = new Test();
      return t1;

   }
}
```

Diagram for Shallow copy program :

```
package com.nit.shallow_copy;

public class ProductDemo {

   public static void main(String[] args)
   {
      Product p1 = new Product(111, "M Series", 22000);
      System.out.println(p1);

      Product p2 = p1;

      p2.setProductId(222);
      p2.setProductName("X Series");
      p2.setProductPrice(25000);
```