Encapsulation in Java

**Encapsulation in Java** is a *process of wrapping code and data together into a single unit*, for example, a capsule which is mixed of several medicines.



We can create a fully encapsulated class in Java by making all the data members of the class private. Now we can use setter and getter methods to set and get the data in it.

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

Advantage of Encapsulation in Java

By providing only a setter or getter method, you can make the class **read-only or write-only**. In other words, you can skip the getter or setter methods.

It provides you the **control over the data**. Suppose you want to set the value of id which should be greater than 100 only, you can write the logic inside the setter method. You can write the logic not to store the negative numbers in the setter methods.

It is a way to achieve **data hiding** in Java because other class will not be able to access the data through the private data members.

The encapsulate class is **easy to test**. So, it is better for unit testing.

The standard IDE's are providing the facility to generate the getters and setters. So, it is **easy and fast to create an encapsulated class** in Java.

Simple Example of Encapsulation in Java

Let's see the simple example of encapsulation that has only one field with its setter and getter methods.

*File: Student.java*

1. //A Java class which is a fully encapsulated class.
2. //It has a private data member and getter and setter methods.
3. **package** com.javatpoint;
4. **public** **class** Student{
5. //private data member
6. **private** String name;
7. //getter method for name
8. **public** String getName(){
9. **return** name;
10. }
11. //setter method for name
12. **public** **void** setName(String name){
13. **this**.name=name
14. }
15. }

*File: Test.java*

1. //A Java class to test the encapsulated class.
2. **package** com.javatpoint;
3. **class** Test{
4. **public** **static** **void** main(String[] args){
5. //creating instance of the encapsulated class
6. Student s=**new** Student();
7. //setting value in the name member
8. s.setName("vijay");
9. //getting value of the name member
10. System.out.println(s.getName());
11. }
12. }

Compile By: javac -d . Test.java

Run By: java com.javatpoint.Test

Output:

vijay

Read-Only class

1. //A Java class which has only getter methods.
2. **public** **class** Student{
3. //private data member
4. **private** String college="AKG";
5. //getter method for college
6. **public** String getCollege(){
7. **return** college;
8. }
9. }

Now, you can't change the value of the college data member which is "AKG".

1. s.setCollege("KITE");//will render compile time error

Write-Only class

1. //A Java class which has only setter methods.
2. **public** **class** Student{
3. //private data member
4. **private** String college;
5. //getter method for college
6. **public** **void** setCollege(String college){
7. **this**.college=college;
8. }
9. }

Now, you can't get the value of the college, you can only change the value of college data member.

1. System.out.println(s.getCollege());//Compile Time Error, because there is no such method
2. System.out.println(s.college);//Compile Time Error, because the college data member is private.
3. //So, it can't be accessed from outside the class

Another Example of Encapsulation in Java

Let's see another example of encapsulation that has only four fields with its setter and getter methods.

*File: Account.java*

1. //A Account class which is a fully encapsulated class.
2. //It has a private data member and getter and setter methods.
3. **class** Account {
4. //private data members
5. **private** **long** acc\_no;
6. **private** String name,email;
7. **private** **float** amount;
8. //public getter and setter methods
9. **public** **long** getAcc\_no() {
10. **return** acc\_no;
11. }
12. **public** **void** setAcc\_no(**long** acc\_no) {
13. **this**.acc\_no = acc\_no;
14. }
15. **public** String getName() {
16. **return** name;
17. }
18. **public** **void** setName(String name) {
19. **this**.name = name;
20. }
21. **public** String getEmail() {
22. **return** email;
23. }
24. **public** **void** setEmail(String email) {
25. **this**.email = email;
26. }
27. **public** **float** getAmount() {
28. **return** amount;
29. }
30. **public** **void** setAmount(**float** amount) {
31. **this**.amount = amount;
32. }
34. }

*File: TestAccount.java*

1. //A Java class to test the encapsulated class Account.
2. **public** **class** TestEncapsulation {
3. **public** **static** **void** main(String[] args) {
4. //creating instance of Account class
5. Account acc=**new** Account();
6. //setting values through setter methods
7. acc.setAcc\_no(7560504000L);
8. acc.setName("Sonoo Jaiswal");
9. acc.setEmail("sonoojaiswal@javatpoint.com");
10. acc.setAmount(500000f);
11. //getting values through getter methods
12. System.out.println(acc.getAcc\_no()+" "+acc.getName()+" "+acc.getEmail()+" "+acc.getAmount());
13. }
14. }

[**Test it Now**](https://compiler.javatpoint.com/opr/test.jsp?filename=TestEncapsulation)

Output:

7560504000 Sonoo Jaiswal sonoojaiswal@javatpoint.com 500000.0