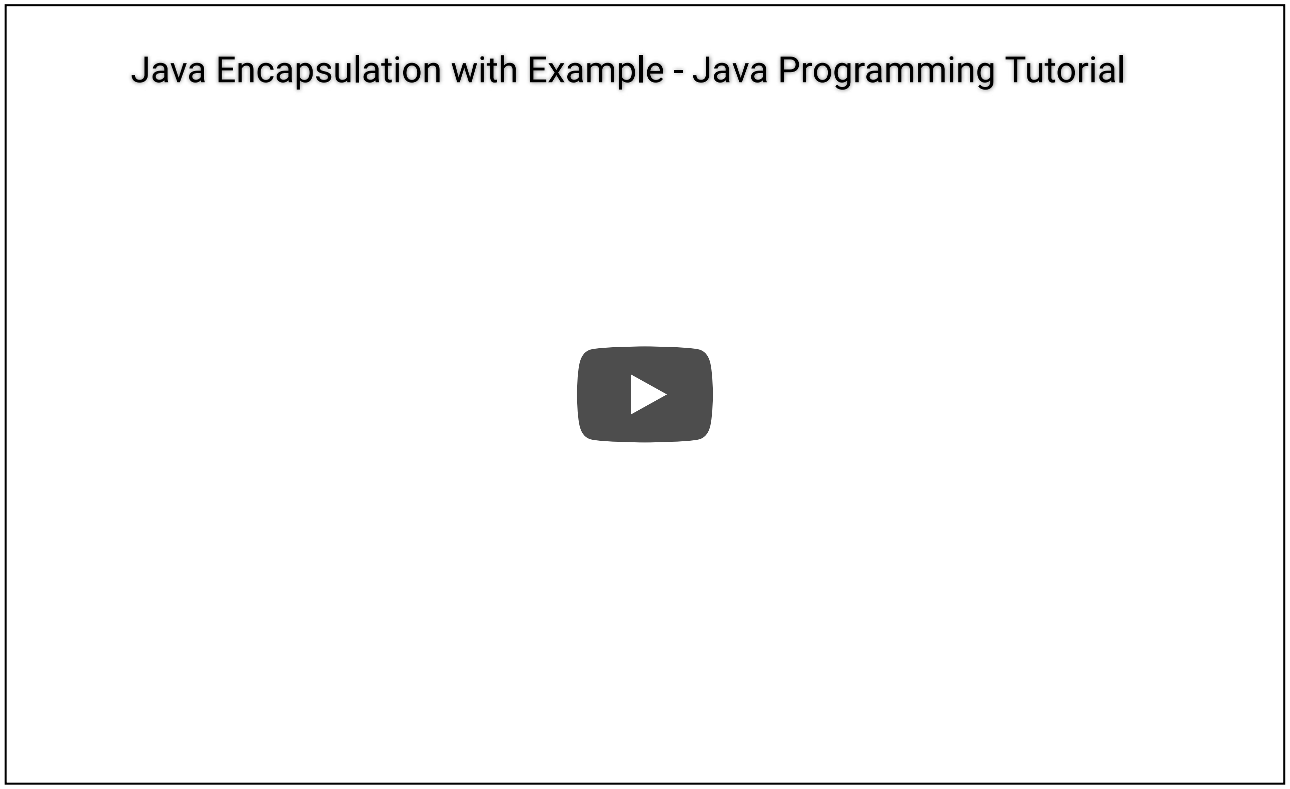
Encapsulation in Java OOPs with Example

What is Encapsulation in Java?

Encapsulation is a principle of wrapping data (variables) and code together as a single unit. It is one of the four OOP concepts. The other three are Inheritance, Polymorphism, and Abstraction.

In this tutorial, you will learn-



Click here (/faq.html#1) if the video is not accessible

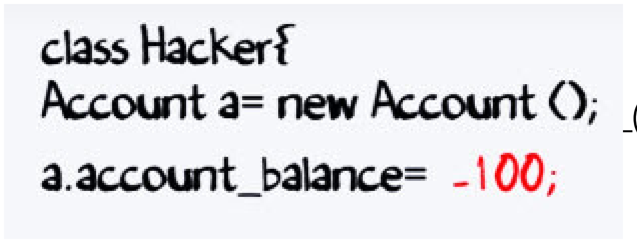
Learn Encapsulation with an Example

To understand what is encapsulation in detail consider the following bank account class with deposit and show balance methods

```
class Account {
    private int account_number;
    private int account_balance;

    public void show Data() {
        //code to show data
    }

    public void deposit(int a) {
        if (a < 0) {
            //show error
        } else
            account_balance = account_balance + a;
    }
}
```

Suppose a hacker managed to gain access to the code of your bank account. Now, he tries to deposit amount -100 into your account by two ways. Let see his first method or approach.

Approach 1: He tries to deposit an invalid amount (say -100) into your bank account by manipulating the code.

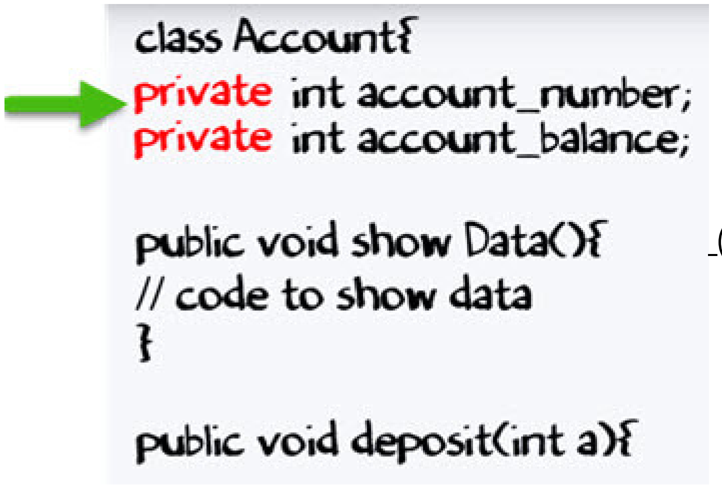


```
class Hacker{
Account a= new Account ();
a.account_balance= -100;
```

(/images/java/052016_0638_LearnJavaEn1.jpg)

Now, the question is – *Is that possible?* Let investigate.

Usually, a variable in a class are set as "private" as shown below. It can only be accessed with the methods defined in the class. No other class or object can access them.

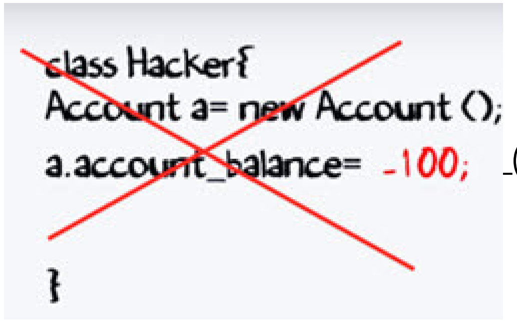


```
class Account{
private int account_number;
private int account_balance;

public void show Data(){
// code to show data
}

public void deposit(int a){
```

[(/images/java/052016_0638_LearnJavaEn2.jpg)](/images/java/052016_0638_LearnJavaEn2.jpg)

If a data member is private, it means it can only be accessed within the same class. No outside class can access private data member or variable of other class.

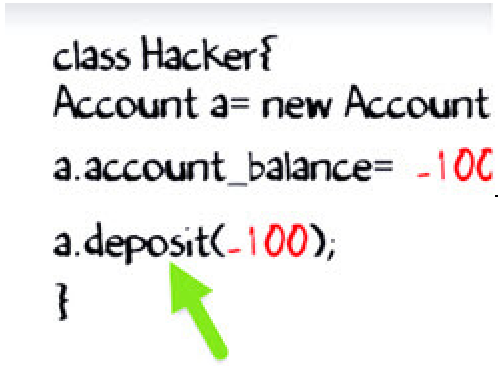
So in our case hacker cannot deposit amount -100 to your account.



```
~~class Hacker{
Account a= new Account ();
a.account_balance= -100;~~

}
```

[(/images/java/052016_0638_LearnJavaEn3.jpg)](/images/java/052016_0638_LearnJavaEn3.jpg)

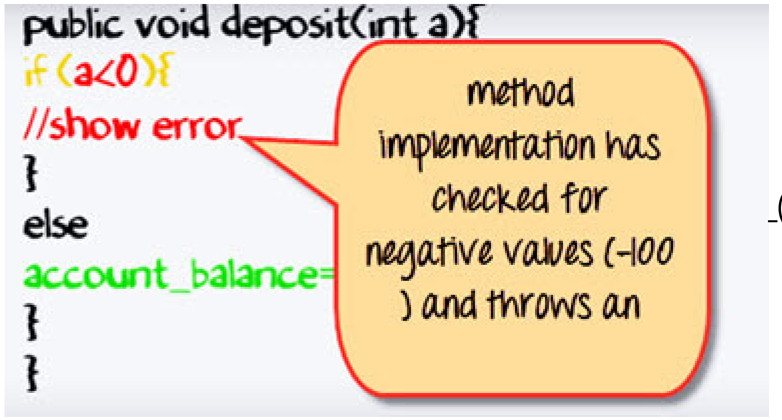
Approach 2: Hacker's first approach failed to deposit the amount. Next, he tries to do deposit a amount -100 by using "deposit" method.



```
class Hacker{
Account a= new Account
a.account_balance= -100
a.deposit(-100);
}
```

(/images/java/052016_0638_LearnJavaEn4.jpg)

But method implementation has a check for negative values. So the second approach also fails.

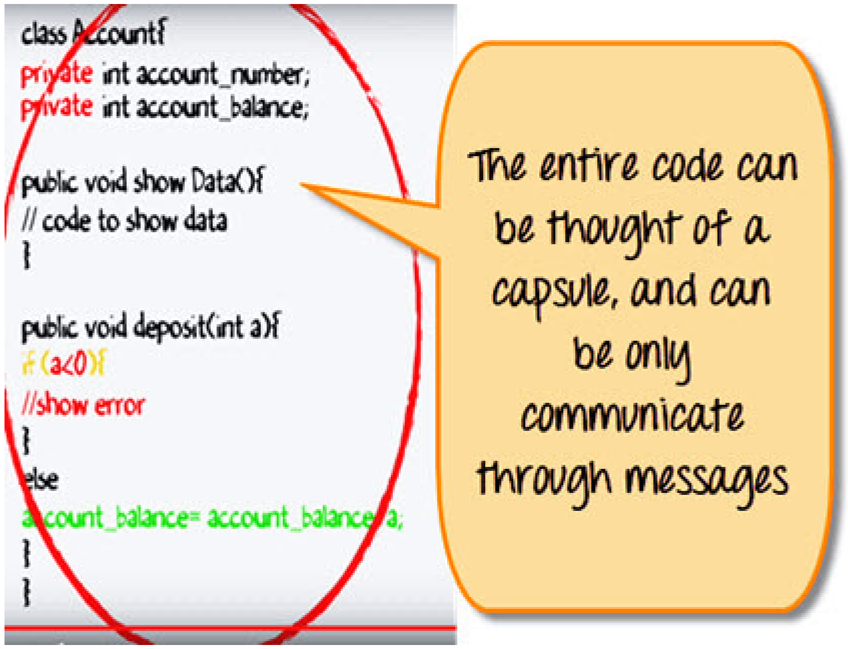


```
public void deposit(int a){
if (a<0){
//show error
}
else
account_balance=
}
}
```

method implementation has checked for negative values (-100) and throws an

(/images/java/052016_0638_LearnJavaEn5.jpg)

Thus, you never expose your data to an external party. Which makes your application secure.



```
class Account{
private int account_number;
private int account_balance;

public void show Data(){
// code to show data
}

public void deposit(int a){
if (a<0){
//show error
}
else
account_balance= account_balance+a;
}
}
```

The entire code can be thought of a capsule, and can be only communicate through messages

(/images/java/052016_0638_LearnJavaEn6.jpg)

The entire code can be thought of a capsule, and you can only communicate through the messages. Hence the name encapsulation.

Data Hiding in Java

Frequently, Java encapsulation is referred as **data hiding**. But more than data hiding, encapsulation concept is meant for better management or grouping of related data.

To achieve a lesser degree of encapsulation in Java, you can use modifiers like "protected" or "public". With encapsulation, developers can change one part of the code easily without affecting other.

Getter and Setter Methods in Java

If a data member is declared "private", then it can only be accessed within the same class. No outside class can access data member of that class. If you need to access these variables, you have to use public "getter" and "setter" methods.

Getter and Setter's methods are used to create, modify, delete and view the variables values.

The following code is an example of getter and setter methods:

```
class Account{
private int account_number;
private int account_balance;
    // getter method
        public int getBalance() {
        return this.account_balance;
    }
    // setter method
        public void setNumber(int num) {
        this.account_number = num;
    }
}
```

In above example, getBalance() method is getter method that reads value of variable account_balance and setNumber() method is setter method that sets or update value for variable account_number.

Abstraction vs. Encapsulation

Often encapsulation is misunderstood with Abstraction. Lets study-

- Encapsulation is more about "How" to achieve a functionality
- Abstraction is more about "What" a class can do.

A simple example to understand this difference is a mobile phone. Where the complex logic in the circuit board is encapsulated in a touch screen, and the interface is provided to abstract it out.

Advantages of Encapsulation in Java

- Encapsulation is binding the data with its related functionalities. Here functionalities mean "methods" and data means "variables"
- So we keep variable and methods in one place. That place is "class." Class is the base for encapsulation.
- With Java Encapsulation, you can hide (restrict access) to critical data members in your code, which improves security
- As we discussed earlier, if a data member is declared "private", then it can only be accessed within the same class. No outside class can access data member (variable) of other class.
- However, if you need to access these variables, you have to use **public "getter" and "setter"** methods.

Prev (/java-data-abstraction.html)

Report a Bug

Next (/java-variables.html)

Selection Sorting in Java Program with Example

(/selection-sorting-java.html)

questions-answers.html)

Top 85 JavaScript Interview Questions & Answers

(/javascript-interview-questions-answers.html)

How to Create Array of Objects in Java

(/array-of-objects.html)

Java Tutorials

(https://www.facebook.com/guru99com/) (https://twitter.com/guru99com) (https://www.youtube.com/channel/UC19i1XD6k88KqHlET8atqFQ) (https://forms.aweber.com/form/46/724807646.htm)

About

About Us (/about-us.html)
Advertise with Us (/advertise-us.html)
Write For Us (/become-an-instructor.html)
Contact Us (/contact-us.html)

Career Suggestion

SAP Career Suggestion Tool (/best-sap-module.html)
Software Testing as a Career (/software-testing-career-complete-guide.html)
Certificates (/certificate-it-professional.html)

Interesting

Books to Read! (/books.html)

Blog (/blog/)

Quiz (/tests.html)

eBook (/ebook-pdf.html)

Execute online

Execute Java Online (/try-java-editor.html)

Execute Javascript (/execute-javascript-online.html)

Execute HTML (/execute-html-online.html)

Execute Python (/execute-python-online.html)