*Oop’s & Encapsulation*

What is OOP?

OOP stands for **Object-Oriented Programming.**

It include class,object,[abstraction](https://stackify.com/oop-concept-abstraction/), [encapsulation](https://stackify.com/oop-concept-for-beginners-what-is-encapsulation/), [inheritance](https://stackify.com/oop-concept-inheritance/) and[polymorphism](https://stackify.com/oop-concept-polymorphism/) we can say that these are the four pillar of oops.

*Encapsulation*

Binding of data and corresponding methods into a single unit is called Encapsulation.

If any java class follows data hiding and abstraction such type of class is said to be encapsulated class.

Example

**package** com.velocity;

**public** **class** Student {

**int** rollNo;

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

Student student=**new** Student();

student.rollNo=1; //passing positive roll number

student.rollNo=-10; //passing negative roll number

}

}

Passing the positive roll, No is correct but negative roll No not valid So roll No cannot be negative.

How we are going to achieve this by using encapsulation-

**package** com.velocity;

**public** **class** Student {

**private** **int** rollNo;

**public** **int** getRollNo() {

**return** rollNo;

}

**public** **void** setRollNo(**int** No) {

**if** (No > 0) {

rollNo = No;

} **else** {

rollNo = 0;

System.***out***.println("Roll number cannot be -ve ");

}

}

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

A a =**new** A();

a.setRollNo(-12);//roll number cannot be -ve

}

}

Output

Roll number cannot be -ve

*Point to be remember*

Every data member should be declared as private and for every member we have to maintain public getter & Setter methods.

**package** com.velocity;

**public** **class** Student {

**private** **int** studentId;

**private** String studentName;

**private** String studentCity;

**public** **int** getStudentId() {

**return** studentId;

}

**public** **void** setStudentId(**int** studentId) {

studentId = studentId;

}

**public** String getStudentName() {

**return** studentName;

}

**public** **void** setStudentName(String studentName) {

studentName = studentName;

}

**public** String getStudentCity() {

**return** studentCity;

}

**public** **void** setStudentCity(String studentCity) {

studentCity = studentCity;

}

}

**package** com.velocity;

**import** java.util.Scanner;

**public** **class** StudentMain {

**public** **static** **void** getUserInput() {

System.***out***.println("Enter the StudentId >> ");

Scanner scanner = **new** Scanner(System.***in***);

**int** id = scanner.nextInt();

System.***out***.println("Enter the Name >> ");

String name = scanner.next();

System.***out***.println("Enter the City >> ");

String city = scanner.next();

Student student = **new** Student();

student.setStudentId(id);

student.setStudentName(name);

student.setStudentCity(city);

System.***out***.println("Student Id>>" + student.getStudentId());

System.***out***.println("Student Name>>" + student.getStudentName());

System.***out***.println("Student City>>" + student.getStudentCity());

}

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

*getUserInput*();

}

}

*Output*

Enter the StudentId >>

1

Enter the Name >>

Rahul

Enter the City >>

Pune

Student Id>>1

Student Name>>Rahul

Student City>>Pune

**The main advantages of encapsulation are:**

1. We can achieve security.

2. Enhancement will become very easy.

3. It improves maintainability and modularity of the application.

4. It provides flexibility to the user to use system very easily.