Encapsulation:

1. Binding of the data members in the single entity is called as encapsulation.

For ex.

class is the entity which contains data variables and method

```
class Employee{
        int salary;
        public int getSalary(){
                return 10000;
        }
}
```

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

Why Encapsulation:

1. Suppose you have an employee class which contain salary as a variable.
2. If we do not follow encapsulation, we can create object of employee class and then this salary variable can to set to negative value also which is not valid in real world as salary cannot be negative.
3. So to overcome this problem and get more control and security over the data we need encapsulation

Example:

```
package com.velocity.demo;

public class Employee {

        public int salary;

}

public class MainTest{
        public static void main(String[] args) {
                Employee employee = new Employee();
                employee.salary=-12500; //salary cannot be negative
        }
}
```

To overcome this issue, we can encapsulate our class as:

```
package com.velocity.demo;

public class Employee {

	private int salary;

	public void setSalary(int sal) {
		//mechanism to check if salary is positive or negative
		if(sal>=0) {
			salary=sal;
		}
		else {
			salary=0;
			System.out.println("Salary cannot be negative");
		}
	}
}

class MainTest{
	public static void main(String[] args) {
		Employee employee = new Employee();
		employee.setSalary(-12500); //setter method won't set the
		salary as negative now
	}
}
```

Program for Encapsulation using Dynamic values.

```
package com.encapsulation;

public class Employee {

	private int employeeId;
	private String employeeName;
	private String employeeCity;


	public int getEmployeeId() {
		return employeeId;
	}

	public void setEmployeeId(int employeeId) {
		this.employeeId = employeeId;
	}

	public String getEmployeeName() {
		return employeeName;
	}

	public void setEmployeeName(String employeeName) {
		this.employeeName = employeeName;
	}

	public String getEmployeeCity() {
		return employeeCity;
	}

	public void setEmployeeCity(String employeeCity) {
		this.employeeCity = employeeCity;
	}
}
```

```
package com.encapsulation;

import java.util.Scanner;

public class TestMain {
	public static void getUserInput() {
		System.out.println("Enter the ID>>");
		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();

		Employee employee = new Employee();
		employee.setEmployeeId(id);
		employee.setEmployeeName(name);
		employee.setEmployeeCity(city);

		System.out.println("Employee Id>>" +
employee.getEmployeeId());
		System.out.println("Employee Name>>" +
employee.getEmployeeName());
		System.out.println("Employee City>>" +
employee.getEmployeeCity());

	}

	public static void main(String[] args) {
		getUserInput();
	}

}
```

Output-
Enter the ID>>
10
Enter the Name>>
ram
Enter the City
pune
Employee Id>>10
Employee Name>>ram
Employee 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.

The main disadvantage of encapsulation is it increases length of the code and slows down execution.