**Day 18**

**Encapsulation in Java**

* Encapsulation in Java is a process of wrapping code and data together into a single unit.
* Encapsulation in Java is the process by which data (variables) and the code that acts upon them (methods) are integrated as a single unit.
* By encapsulating a class's variables, other classes cannot access them, and only the methods of the class can access them.
* By providing only a setter or getter method, you can make the class**read-only or write-only.**

**Advantages**

* It provides you the **control over the data.**
* 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.

Example

Class Student

{

private string name;

public String getNames()

{

return name;

}

public void setName(String name)

{

this.name = name;

}

}

class Test

{

public static void main(String args[])

{

Student ob = new Student();

ob.setName(“Jitty”);

System.out.println(ob.getName());

}

}

**Final Keyword**

* The final keyword is a non-access modifier used for classes, attributes and methods, which makes them non-changeable (impossible to inherit or override).
* The final keyword is useful when you want a variable to always store the same value.
* The final keyword in java is used to restrict the user. The java final keyword can be used in many contexts. Final can be:

1. Variable
2. Method
3. Class

**Final variable**

If you make any variable as final, you cannot change the value of final variable. It will be constant.

Example

class A

{

final int speedLimit = 40; //final variable

void show()

{

speedLimit = 50;

public static void main(String args[])

{

A ob = new A();

ob.show(); // compile time error. Since variable value will be constant.

}

}

**Final Method**

If you make any method as final, you cannot override it.

class A

{

final void show()

{

System.out.println(“Show”);

}

}

class B extends A

{

void Show()

{

System.out.println(“Display”);

}

public static void main(String args[])

{

B ob = new B();

Ob.Show(); // compile time error.

}

}

**Final Class**

If you make any class as final, you cannot extend it.

Example

final class A

{

}

class B extends A // compile time error

{

void show()

{

System.out.println(“Display”);

}

public static void main(String args[])

{

B ob = new B();

Ob.Show(); // compile time error.

}

}

**Note:**

* Final class extend is not possible
* Final method, we can extend, but we cannot override.