Course code : **CSE2005**

Course title : **Object Oriented Programming**

Data abstraction and Encapsulation

Objectives

This session will give the knowledge about

- Data Abstraction
- Encapsulation

Data Abstraction

Abstraction denotes essential characteristics of an object that distinguish it from all other kinds of objects and thus provide crisply defined conceptual boundaries, relative to the perspective of the viewer.

-Grady Booch

Abstraction is the process of taking only a set of essential characteristics from something

Data Abstraction

Example

- For a Doctor -> you are a Patient

  Name, Age, Old medical records

- For a Teacher -> you are a Student

  Name, Roll Number/RegNo, Education background

- For HR Staff -> you are _______________

Here, For a doctor you will not share your roll no.

Data Abstraction

```
public class Student {
    public int regno;
    public String name;
    public void learn() {
        //learn course
    }
}
```

Showing only essential data using public declaration

Providing only essential functions using public definition

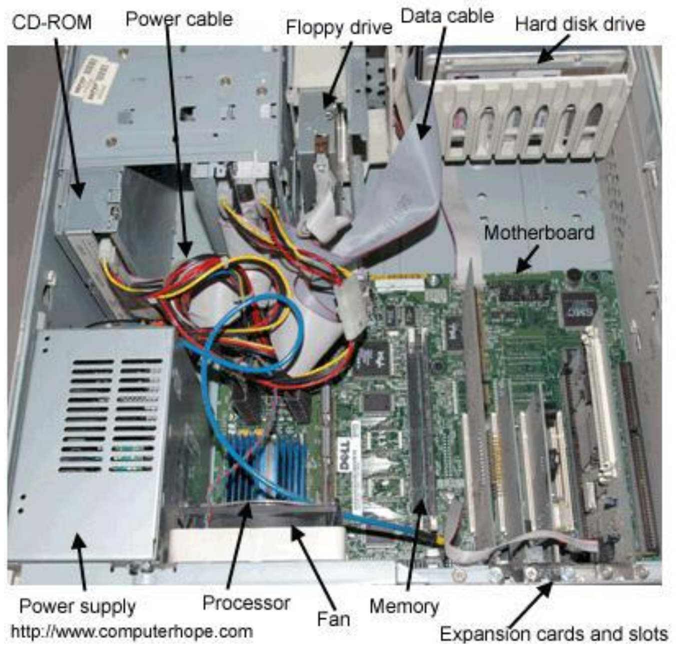
Creating Student Object

```
    Student s= new Student();
    s.display(); // access members using " . "
```

Encapsulation

Would you like it if your CPU is given to you like this?

What are the problems if it were given to you like this?



Encapsulation is the process of compartmentalizing the elements of abstraction that constitute its structure and behavior; encapsulation serves to separate the contractual interface of an abstraction and its implementation.

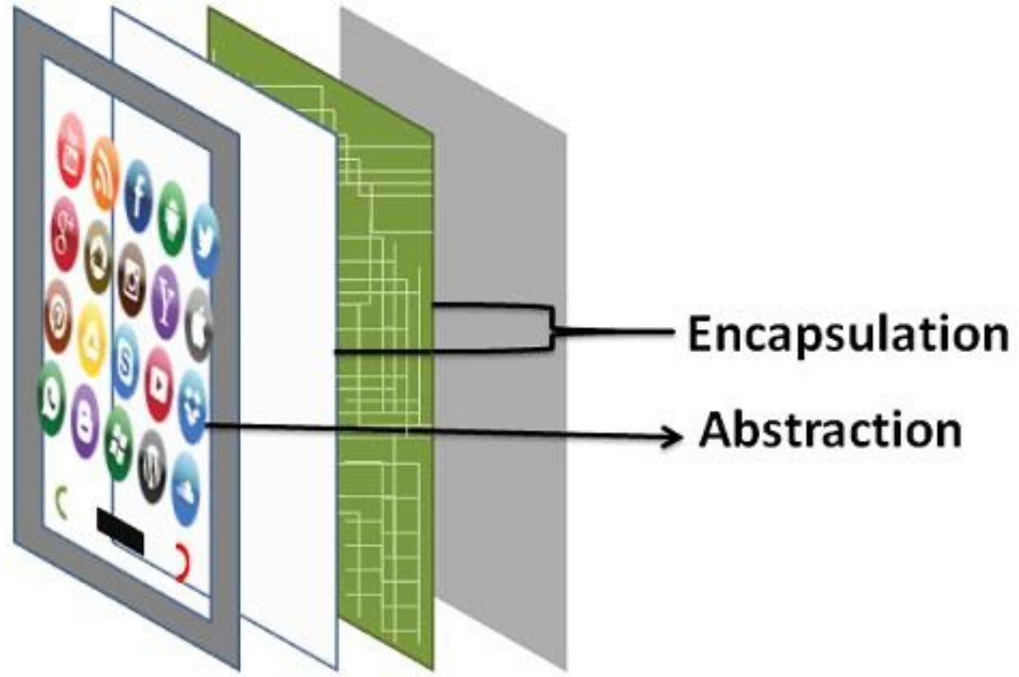
- Grady Booch

Encapsulation

Encapsulation is binding data and operations that work on data together in a construct.

Encapsulation involves Data and Implementation Hiding.

Encapsulation is hiding the implementation level details
Abstraction is exposing only the interface



Encapsulation

```
public class Student {
    ~~public~~ private int mobileNo;
    ~~public~~ private String nickName;
    ~~public~~ private void hobbies() {
        //learn course
    }
}
```

Hiding details using private declaration

Hiding implementation using private definition

Creating Student Object

```
    Student s= new Student();
    s.display(); // access members using " . "
```

Access Specifiers

- Java provides access specifiers to control access to class members
- Access specifiers help implement:
    - Encapsulation by hiding implementation-level details in a class
    - Abstraction by exposing only the interface of the class to the external world
- The **private** access specifier is generally used to encapsulate or hide the member data in the class
- The **public** access specifier is used to expose the member functions as interfaces to the outside world

Access Specifiers

Access Modifier	Private	Default	Protected	Public
within class	Y	Y	Y	Y
within package	N	Y	Y	Y
outside package by subclass only	N	N	Y	Y
outside package	N	N	N	Y

Access Specifiers

```
package vit.family;
public class Myself {
      private int packetMoney=2000;
      int salary=40000;
      protected int mob=909277882;
      public String name="Java";

      public void display(){
                System.out.println(packetMoney+
                      " "+salary+" "+mob+" "+name);
      }    }

package vit.family;
class Family extends Myself{
      public void display(){
            System.out.println(salary+" "+mob+" "+name);
      }      }
```

```
package vit.friends;
import vit.family.*;
public class Friends extends Myself {
      public void display(){
                System.out.println(mob+" "+name);
      }
}

package vit.friends;
import vit.family.Myself;
class Others {
                public void display(){
                                Myself obj=new Myself();
                                System.out.println(obj.name);
                }
}
```

Access Specifiers

```
package vit.demo;

class Point{
		private int x;
		private int y;
		public void setX(int x) {
				x = (x>79?79:(x<0?0:x));
		}
		public void setY(int y) {
				y = (y>24?24:(y<0?0:y));
		}
		public int getX() {
				return x;
		}
		public int getY() {
				return y;
		}
}

class Main {
		public static void main(String arg[]) {
				Point p1=new Point();
				p1.setX(22);
				p1.setY(24);
				System.out.println(p1.getX());
				System.out.println(p1.getY());
		}
}
```

Summary

We have discussed about

- Data Abstraction
- Encapsulation