Encapsulation

⇒ It is the mechanism of binding the data (variables) and behaviour (methods) into a single unit (class).
⇒ Real World Example :
- Capsule : medicine is binded or wrapped into a single unit
- Car - all car parts like stearing, beaks, gears are binded or wrapped into car

⇒ Java Example :

```
class Car
{
    int no_of_tyres;
    String brand;

    void start()
    {
        //car starts
    }
}
```

→ Technically, all the Java Classes are example of encapsulation....
But these are not strictly encapsulated class, to become strictly encapsulated class, it has to follow some rules....

⇒ Rules for Encapsulated Class :-
1. All the variables should be private
2. It should contain public getter and setter methods.

```
class Car
{
    private int no_of_tyres;
    private String brand;

    // public getter and setter methods

    void start()
    {
        //car starts
    }
}
```

Use of Getter & Setter methods

```
class Car
{
    private int no_of_tyres;
    private String brand;

    // public getter and setter methods

    void start()
    {
        //car starts
    }
}
```

```
class MainApp
{

}
```

if we want to use no_of_tyres and brand variable in another class, then we have to use getter and setter methods

⇒ Use of encapsulation :-
1. Security
   - Protects the data
2. Improves code maintability
3. Improves the flexibility
   etc

Packages

⇒ Package is the group of similar type of classes or interfaces or packages
⇒ For eg.

```
class CarLoan
{
    ----
}
```

```
class EducationLoan
{
    ----
}
```

```
class CalculateLoanInterest
{

}
```

package loan

```
class Accounts
{

}
```

```
class User
{
    ----
}
```

```
class AUthentication
{
    // register and login
}
```

package customers

⇒ How to use packages ?

1. Class should be in particular package

```
package loan;

class CarLoan
{
    ----
}
```

```
package loan;

class EducationLoan
{
    ----
}
```

```
package customer;

class User
{
    -----
}
```

2. Import classes from another package

```
package customer;

import loan.CarLoan;

class User
{
    CarLoan cl = new CarLoan();
}
```

```
package customer;

//import loan.CarLoan;
//import loan.EducationLoan;

import loan.*;

class User
{
    CarLoan cl = new CarLoan();

    EducationLoan el = new EducationLoan();
}
```

→ Import statement is used after package statement
→ Import statements can be multiple
→ Different ways to import classes :-
   >> import packagename.ClassName;
   >> import packagename.*;

→ Package statement should always be first statement
→ One class can be only in one package
   (package statement should only be one)

⇒ Package Naming Convention :

→ Company Name : Smart Programming
→ Domain : www.smartprogramming.in
→ Client Name : ICICI
→ Project Name : Loan Application
→ Modules :
   - CarLoan
   - HomeLoan
   - EducationLoan

```
in.smartprogramming.icici.loanapp.CarLoan
in.smartprogramming.icici.loanapp.HomeLoan
in.smartprogramming.icici.loanapp.EducationLoan
```

```
in.smartprogramming.icici.loanapp
```

```
com.google.springapp.dao
com.google.springapp.service
```

Access-Modifiers

⇒ Access-modifiers are the keywords which provides the accessability where we can use the variables or methods or classes etc
⇒ There are 4 access-modifiers :-
1. public - can be used in different package
2. protected - can be used in different package but in case of inheritance
3. default - can be used in same package but not different package
4. private - cannot be used in different class

p1

```
package p1;

public class A
{
    private void m1()
    {

    }
}
```

```
package p1;

public class B
{

}
```

p2

```
package p2;

public class C extends A
{
    public static void main(String[] args)
    {
        A obj = new A();
        obj.m1();
    }
}
```


Interview Questions

⇒ Encapsulation
1. What is encapsulation in Java, and why is it important?
2. How do we achieve encapsulation in Java?
3. What is the difference between encapsulation and data hiding?
4. Why should instance variables be private in an encapsulated class?
5. Can we achieve encapsulation without using getters and setters?
6. What are the advantages of encapsulation in Java?
7. Is encapsulation related to access modifiers? If yes, how?
8. Can a fully encapsulated class be mutable? How?
9. How does encapsulation improve maintainability in Java applications?
10. Provide an example of a real-world scenario where encapsulation is useful.
11. What is a POJO class in Java?
12. What is a JavaBean class in Java?
13. What is the difference between POJO and JavaBean classes?
14. What are the common mistakes that break encapsulation in Java?
15. How is encapsulation different from abstraction in Java?

⇒ Access Modifiers
16. What are the different types of access modifiers in Java?
17. What is the difference between private, default, protected, and public access modifiers?
18. Can a top-level class be private or protected in Java? Why or why not?
19. What happens if we do not specify an access modifier in Java?
20. Can we override a method by changing its access modifier?
21. Can a private method be inherited? Why or why not?
22. How does the protected modifier work in the same package and in different packages?
23. Can we restrict a class from being inherited using access modifiers?
24. What is the default access level of an interface method in Java?
25. Can a private method be overridden in a subclass? Why or why not?
26. Can we declare a constructor as private? Why would we do this?
27. What is the difference between default access and protected access in Java?
28. Can we make an interface method protected or private? Why or why not?

⇒ Packages
29. What is a package in Java, and why is it used?
30. What is the difference between import java.util.*; and import java.util.Scanner;?
31. Can a class in one package access a class in another package without import?
32. What happens if two packages contain a class with the same name?
33. How can we create a user-defined package in Java?
34. What is the difference between import static and normal import in Java?
35. What is the purpose of the package keyword?
36. Can we have multiple package statements in a single Java file?
37. What happens if a Java file does not have a package statement?
38. How does package access affect class visibility in Java?
39. What is the difference between built-in packages and user-defined packages?
40. Can we have sub-packages in Java? How are they created and accessed?
41. How do access modifiers interact with packages in Java?