**Day 19**

**Encapsulation**

Encapsulation is the process of combining data and functions into a single unit called class. In Encapsulation, the data is not accessed directly; it is accessed through the functions present inside the class. In simpler words, attributes of the class are kept private and public getter and setter methods are provided to manipulate these attributes. Thus, encapsulation makes the concept of data hiding possible.(**Data hiding**: a language feature to restrict access to members of an object, reducing the negative effect due to dependencies. e.g. "protected", "private" feature in Java).

**Abstraction**

We try to obtain an **abstract view**, model or structure of a real life problem, and reduce its unnecessary details. With definition of properties of problems, including the data which are affected and the operations which are identified, the model abstracted from problems can be a standard solution to this type of problems. It is an efficient way since there are nebulous real-life problems that have similar properties.

In simple terms, it is hiding the unnecessary details & showing only the essential parts/functionalities to the user.

Data binding **:** Data binding is a process of binding the application UI and business logic. Any change made in the business logic will reflect directly to the application UI.

**Abstraction** is achieved in 2 ways :

* Abstract class
* Interfaces (Pure Abstraction)

1. **Abstract Class**

* An abstract class must be declared with an abstract keyword.
* It can have abstract and non-abstract methods.
* It cannot be instantiated.
* It can have constructors and static methods also.
* It can have final methods which will force the subclass not to change the body of the method.

abstract class Animal {

   abstract void walk();

   void breathe() {

       System.out.println("This animal breathes air");

   }

   Animal() {

       System.out.println("You are about to create an Animal.");

   }

}

class Horse extends Animal {

   Horse() {

       System.out.println("Wow, you have created a Horse!");

   }

   void walk() {

       System.out.println("Horse walks on 4 legs");

   }

}

class Chicken extends Animal {

   Chicken() {

       System.out.println("Wow, you have created a Chicken!");

   }

   void walk() {

       System.out.println("Chicken walks on 2 legs");

   }

}

public class OOPS {

   public static void main(String args[]) {

      Horse horse = new Horse();

      horse.walk();

      horse.breathe();

   }

}

**QUESTIONS**

**1. What is Encapsulation in Java? Why is it called Data hiding?**

Ans: The process of binding data and corresponding methods (behavior) together into a single unit is called encapsulation in Java.

In other words, encapsulation is a programming technique that binds the class members (variables and methods) together and prevents them from being accessed by other classes, thereby we can keep variables and methods safes from outside interference and misuse.

If a field is declared private in the class then it cannot be accessed by anyone outside the class and hides the fields within the class. Therefore, Encapsulation is also called data hiding.

**2. What are the important features of Encapsulation?**

Ans: Encapsulation means combining the data of our application and its manipulation in one place. It allows the state of an object to be accessed and modified through behavior. It reduces the coupling of modules and increases the cohesion inside them.

**3. What is the advantage of Encapsulation?**

Ans: There are the following advantages of encapsulation in Java. They are as follows:

* The encapsulated code is more flexible and easy to change with new requirements.
* It prevents the other classes to access the private fields.
* Encapsulation allows modifying implemented code without breaking other code who have implemented the code.
* It keeps the data and codes safe from external inheritance. Thus, Encapsulation helps to achieve security.
* It improves the maintainability of the application.

**4. What are the main benefits of using encapsulation in Java?**

Ans: The main benefits of using encapsulation are:

* The main benefit of encapsulation is the ability to modify the implemented code without breaking the others code who have implemented the code.
* It also provides us with maintainability, flexibility, and extensibility to our code.
* The fields of a class can be made read-only or write-only.
* A class can have total control over what is stored in its fields.

**5. How to achieve encapsulation in Java? Give an example.**

Ans: There are two key points that should be kept in mind to achieve the encapsulation in Java. They are as follows:

* Declare the variable of the class as private.
* Provide public setter and getter methods to modify the values of variables.

Let’s understand it with the help of an example program.

public class EncapsulationTest{

private String name;

private String idNum;

private int age;

public int getAge() {

return age;

}

public String getName() {

return name;

}

public String getIdNum() {

return idNum;

}

public void setAge( int newAge) {

age = newAge;

}

public void setName(String newName) {

name = newName;

}

public void setIdNum( String newId) {

idNum = newId;

}

}

**6. What is data hiding in Java?**

Ans: An outside person cannot access our internal data directly or our internal data should not go out directly. This oops feature is called data hiding n Java. After validation or authentication, the outside person can access our internal data.  
  
**7. How to achieve Data hiding programmatically?**

Ans: By declaring data members (variables) as private, we can achieve or implement data hiding. If the variables are declared as private in the class, nobody can access them from outside the class.

The biggest advantage of data hiding is we can achieve security.

**8. What is a Tightly encapsulated class in Java?**

Ans: If each variable is declared as private in the class, it is called tightly encapsulated class in Java. For tightly encapsulated class, we are not required to check whether class contains getter and setter method or not and whether these methods are declared as public or not.

**9. What is the difference between Abstraction and Encapsulation?**  
Or, how abstraction is different from encapsulation in Java?

Ans: There are the following differences between Abstraction and Encapsulation:

a) Abstraction solves the problem at the design level whereas encapsulation solves the problem at the implementation level.

b) Abstraction is implemented in Java using Interface and Abstract class whereas encapsulation is implemented using private and protected access modifiers.

c) Abstraction is used to hide the unwanted data and giving relevant data whereas encapsulation is used for hiding data and code in a single unit to prevent access from outside.

d) The real-time example of Abstraction is TV Remote Button whereas the real-time example of Encapsulation is medical medicine.

**10. Can we achieve abstraction without encapsulation in Java?**

Ans: Yes, we can achieve abstraction without encapsulation because both are different things and different concepts.

**11. What are getter and setter methods in Java?**

Ans: In Java, setter method is a method that is used for updating the values of a variable. This method is also known as mutator method.

Getter method is a method that is used to retrieve the value of a variable or return the value of the private member variable. This method is also known as an accessor method.

**12. In the following code, radius is declared as private in the class Circle, and myCircle is an instance of class Circle. Does the code cause any error problems? If so, explain why?**

class Circle {

private double radius = 1;

/\*\* Find area of the circle \*/

public double getArea() {

return radius \* radius \* Math.PI;

}

public static void main(String[] args) {

Circle myCircle = new Circle();

System.out.println("Radius is " + myCircle.radius);

System.out.println("Area of cirle: " +myCircle.getArea());

}

}

Ans: No, the above code will not create any problem. The code will be compiled successfully. The output is: Radius is 1.0, Area of cirle: 3.141592653589793.

**13. Does reflection violates encapsulation in Java?**

Ans: Reflection violates encapsulation because it reveals the internal data structures.

**14. Explain design pattern based on encapsulation in java?**

Ans: In many design patterns, Java uses the encapsulation technique and one of them is Factory pattern which is used to create the objects.

Factory pattern is a better choice in creating the object of those classes whose creation logic can vary. It is also used for creating different implementations of the same interface.

‘BorderFactory class’ of JDK is a good example of encapsulation in Java which creates different types of ‘border’ and encapsulates creation logic of border.

**15. How can the variable of the EncapsulationTest be accessed by using getter and setter methods?**

Ans: The public setXXX() and getXXX() methods are access points of the instance variable of EncapsulationTest class. Basically, these methods are known as getter and setter methods.

Therefore, any class that wants to access variable should access them through these getters and setters. The variables of the EncapsulationTest class can be accessed  as shown in the following code:

public class RunEncapTest {

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

{

EncaptulationTest encap = new EncapsulationTest();

encap.setName("John");

encap.setAge(22);

encap.setId(123456);

System.out.println("Name: " +encap.getName()); System.out.println("Age: " +encap.getAge());

System.out.println("Id: " +encap.getId());

}

}

This would produce following output: Name: John Age: 22 Id: 123456.