Lec 15 Encapsulation:

This lesson explains the concept of encapsulation in Java, showing how data and methods are bundled together in a class and how access to data is controlled using private variables with public getter and setter methods. Encapsulation helps keep data safe, organized, and easy to manage, especially when dealing with large or sensitive information.

(Suppose we have to send data from class1 to class2) Rather than sending bulk data from one class1 to class2's method's parameter, it is better to use another class named Data (POJO, DTO, Bean,Model). In this data class we will make the instance variables, setters and getters of data to be sent.

Set class1 's data in Data class's object and then send this Data object to class2 where the class2 use Data classes getter method's to unravel the data of class1.

This whole process is called data encapsulation. Data is controlled/managed.

Understanding Objects, Classes, and Instances

In Java, a class is a user-defined data type, and an object is an instance of a class. Creating an object involves class loading, setting default values, and eventually destroying the object. The object is used to access instance variables and methods defined in the class.

Introduction to the `this` Keyword

The this keyword in Java refers to the current object of the class. It is used inside methods to access the instance variables and distinguish them from local variables with the same name.

Local vs Instance Variables: Variable Shadowing

If a method has a local variable with the same name as an instance variable, the local variable takes precedence inside the method. To access the instance variable in this case, you must use this.variableName.

The Role of the `this` Keyword in Methods

Java automatically provides the current object to non-static methods using this, so you don’t need to manually pass the object as a parameter. This

mechanism simplifies object access within class methods.

Main Use of `this`: Differentiating Variables

The primary use of this is to differentiate between instance variables and local variables when they share the same name, ensuring the correct variable is accessed or modified.

Introduction to Encapsulation

Encapsulation is a core concept of object-oriented programming that binds data (instance variables) and methods into a single unit (class). It also hides the internal state of an object from outside access, exposing only what is necessary.

Making Variables Private for Data Hiding

To implement encapsulation, instance variables are made private so that they cannot be accessed directly from outside the class. This prevents unauthorized or accidental modification of data.

Controlled Access with Getter and Setter Methods

Access to private data is provided through public methods, commonly known as getters (to read data) and setters (to modify data). These methods can include logic for authorization or validation before allowing access or changes.

Example: Access Control with Authorization

An example shows how a private balance variable is accessed through a public method that requires a PIN for authorization, illustrating how encapsulation can enforce rules and security.

Naming Conventions: Setters and Getters

Setters and getters should be named following conventions like setVariableName and getVariableName for clarity and consistency. Tools like Eclipse can auto-generate these methods.

Data Binding and Encapsulation for Data Transfer

Encapsulation is also used for data binding, where multiple related data fields are grouped into a single object (like a "packet") and passed between classes or methods. This is more efficient than passing many individual variables, especially with large datasets.

Using Encapsulation for Efficient Data Transfer (DTO)

Creating a class with private fields and public getters/setters allows all relevant data to be packed into one object (Data Transfer Object or DTO), which can be easily passed around. This approach is used in various frameworks and is known by different names like POJO, bean, entity, or model.

Encapsulation with Arrays and Collections

When managing multiple objects (like employees), encapsulation lets you store each object's data securely in arrays or collections, making data management organized and reducing the risk of mixing up information.

Recap and Next Steps

The session covers data hiding, controlled access, and data binding through encapsulation. It also introduces the idea that classes can have other features like constructors and methods, which will be covered in future lessons.