**Encapsulation in Java**

**Introduction**

Encapsulation is defined as the wrapping up of data under a single unit. It is the mechanism that binds together code and the data it manipulates. Another way to think about encapsulation is that it is a protective shield that prevents the data from being accessed by the code outside this shield.

Encapsulation uses private variables to hide internal data and exposes them through public methods like getters and setters. This ensures controlled access and hides implementation details, combining data hiding and abstraction.

**Real-life Example:**

Imagine a mountain bike's gears. They hide the internal mechanisms, allowing riders to switch gears with simple levers. You don't need to know how it works inside; you just use the controls. That's encapsulation: hiding complexity behind a user-friendly interface.

**Java Example:**

In a banking application, encapsulation ensures that sensitive information like the account balance is kept private within the BankAccount class.

For instance, we would declare the account balance as a private attribute, preventing direct access from outside the class. Instead, we provide methods like deposit() and withdraw() to interact with the account securely.

This way, other parts of the system can perform transactions without directly accessing or modifying the account balance, ensuring data integrity and security.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_