**11/07/2025:**

Modern JAVA is from version 9-17

**Introduction to modern JAVA:**

To implement modern Java, into any particular, existing outdated system, we need to use it as a toolkit, and we will break our application into modules. Let’s say we want to build a smart bank, then we will.

* Let say we want to build a smart bank so the very first thing we will do is to create our first module i,e bank.core.
* We will create a module for the application interface ,i.e, bank.app
* After this we will move towards **phase II.**

**Customers as records(Java 14):**

Instead of creating bulky classes for customer data we use, records to store customer info.

Example:

Phase 2 (Customer as Records – Java 14):

1. Instead of creating bulky classes for customer data we use recorsd ro store customer information.

• Records command // modern java

• Command: record Customer(String name, int accountNo, double balance) {}

• No need to write constructors, getters, or toString() anymore — everything was auto-generated

Phase 3: Control account types with sealed classes (Java 17):

1. To prevent misuse we will create a sealed class - bank account so no one could create random new account types because it will only offer Current and Saving account.

sealed class BankAccount permits SavingsAccount, CurrentAccount {}

//Sealed classes

**Note: Only trusted types can inherit the bank account class.**

**Phase 4:**

**Detecting types at runtime(Java 16) pattern matching**

While generating receipt or running reports the system uses smart instance of **pattern-matching** to identify account types quickly without **messy typecasting.**

**Phase 5: Elegant Receipts with Text Blocks (Java 15)**  
  
Gone were the days of clumsy string concatenation. Receipts were now beautifully formatted using text blocks:  
  
String receipt = """  
=== Bank Receipt ===   
Customer: %s  
Account Number: %d  
Account Type: %s  
Balance: ₹%.2f  
""".formatted(...);

**Problem Statement: Bank Management System using Modern Java (Java 14-17 Features)**

**Scenario:**

**You are developing a Bank Management System that uses the latest features of Java (from Java 9 to Java 17) to handle different types of accounts and users. The system should ensure data integrity, type safety, and a clean code structure using Records, Sealed Classes, Pattern Matching, and Modules.**

**Requirements:**

**1. Modular Structure (Java 9)**

**Create two modules:**

**bank.core contains business logic**

**bank.app main application module**

**2. Record (Java 14+)**

**Create a record called Customer with:**

**String name**

**int accountNo**

**double balance**

**3. Sealed Class Hierarchy (Java 17)**

**Create a sealed class BankAccount with two permitted classes**

**4. Pattern Matching with instanceof (Java 16)**

**In the main logic, check the type of BankAccount using instanceof and print account-specific actions.**

**5. Text Blocks (Java 15)**

**Use a text block to display a formatted bank receipt.**

**Example Output:**

**Bank Receipt\*\*\***

**Customer: Ak**