Scenario Based Problems

Story 1: Smart Library System

Scenario:
A library wants a system to manage different types of books.

Requirements:

1. Create a class Book with:
    - Fields: `bookId`, `title`, `price`
    - Constructor to initialize values
    - Method `calculateFine(int daysLate)`
2. Create child classes:
    - `TextBook`
    - `Magazine`
3. Override `calculateFine()`:
    - TextBook → ₹2 per day
    - Magazine → ₹5 per day
4. Maintain a **static variable** `libraryName` shared across all books.
5. Use **polymorphism** to calculate fine using parent reference.

Story 2: Online Food Delivery App

Scenario:
An app calculates final bill amount for different food orders.

Requirements:

1. Create class `Order`
    - Fields: `orderId`, `baseAmount`
    - Constructor
    - Method `calculateBill()`
2. Create subclasses:
    - `RegularOrder` → no discount
    - `PremiumOrder` → 20% discount
3. Store **static variable** `deliveryCharge = 40`.
4. Calculate total bill using **polymorphism**.

Story 3: Bank Interest Calculator

Scenario:
A bank provides different interest rates for account types.

Requirements:

1. Class `Account`
    - Fields: `accountNumber`, `balance`
    - Method `calculateInterest()`
2. Subclasses:
    - `SavingsAccount` → 4%
    - `CurrentAccount` → 2%

3. Static field `bankName`.
4. Use parent reference to call child interest methods.

Story 4: Employee Management System

Scenario:
A company manages employees and generates email IDs.

Requirements:

1. Base class `Employee`
    - Fields: `empId`, `name`
    - Method `generateEmail()`
        - Format: `name@company.com`
        - Convert name to lowercase
2. Subclass `Manager`
    - Extra field: `department`
    - Override `generateEmail()` to include department

String Concepts Used:

- `toLowerCase()`
- String concatenation
- Method overriding

Story 5: E-Commerce Product Categorization

Scenario:
Products are grouped and searched by name.

Requirements:

1. Class `Product`
    - Fields: `productId`, `productName`
    - Method `isMatch(String keyword)`
        - Return true if productName contains keyword
2. Subclass `ElectronicProduct`
    - Add brand name
    - Match keyword against both name & brand

String Methods:

- `contains()`
- `equalsIgnoreCase()`

Story 6: Mobile Phone Contact App

Scenario:
Contacts are stored and searched.

Requirements:

1. Base class `Contact`
    - Fields: `name`, `phoneNumber`
    - Method `display()`

2. Subclass `BusinessContact`
    - Extra field: `companyName`
    - Display name in **uppercase**

String Handling:

- `toUpperCase()`
- String formatting

Story 7: Student Result System

Scenario:
University calculates student grade based on name and marks.

Requirements:

1. Class `Student`
    - Fields: `rollNo`, `name`, `marks`
    - Method `calculateGrade()`
2. Subclass `EngineeringStudent`
    - Extra field: `branch`
    - Append branch to student name using String methods

Concepts:

- String concatenation
- Inheritance

Story 9: Online Learning Platform (Mini Project)

Scenario:
Platform offers multiple course types.

Requirements:

1. Base class `Course`
    - Fields: `courseId`, `courseName`, `price`
    - Method `getFinalPrice()`
2. Subclasses:
    - `RecordedCourse` → 10% discount
    - `LiveCourse` → 5% discount
3. Static field `platformName`
4. Course name formatting:
    - Convert to Title Case using String methods
5. Display all course details using polymorphism.

Concepts Combined:
✔ Inheritance
✔ Polymorphism
✔ Static
✔ Constructors
✔ String handling