OOP Foundations & Interfaces – Exercises

Section A: Conceptual Exercises

1. Define the following terms as used in Object-Oriented Programming:
   - Class
   - Object
   - Encapsulation
   - Abstraction
2. State **four advantages** of using Object-Oriented Programming compared to procedural programming.
3. Distinguish between:
   - a) A class and an object
   - b) Encapsulation and abstraction
   - c) Inheritance and polymorphism
4. Explain why **interfaces** are important in Java. Give **two practical use cases**.
5. Why does Java not support multiple inheritance using classes? How do interfaces solve this problem?
6. a) Explain the concept of **interface inheritance** in Java
   b) With the help of a diagram and code example, demonstrate how interfaces promote loose coupling.
7. Discuss how **encapsulation and interfaces** together improve software maintainability.

Section B: Classes & Objects -Practical Exercises

Exercise 1: Creating a Class

Create a Java class called `Student` with the following:

- Attributes:
  - `studentId`
  - `name`
  - `course`
- Methods:
  - `displayDetails()`

Write a `main` method to create two `Student` objects and display their details.

Exercise 2: Constructors

Modify the `Student` class to:

- Include a **parameterized constructor**
- Initialize all attributes using the constructor

- Demonstrate object creation using the constructor

Exercise 3: Encapsulation

Create a class `BankAccount` with:

- Private attributes: `accountNumber`, `balance`
- Public methods: `deposit()`, `withdraw()`, `getBalance()`

Ensure that:

- Balance cannot be directly accessed
- Withdrawal amount does not exceed the balance

Section C: Interfaces (Core Focus)

Exercise 4: Implementing an Interface

Create an interface called `Payable` with a method:

```
double calculatePay();
```

Create two classes:

- `Employee`
- `Contractor`

Both classes should implement the `Payable` interface and provide their own implementation of `calculatePay()`.

Exercise 5: Multiple Interfaces

Create two interfaces:

- `Printable`
- `Scannable`

Each interface should have one method.

Create a class `MultiFunctionPrinter` that implements **both interfaces**.

Exercise 6: Interface as a Data Type (Very Important)

Write a program where:

- An interface reference is used to refer to an implementing class

- Demonstrate runtime polymorphism

Example:

```
Payable p = new Employee();
p.calculatePay();
```

Explain what happens during execution.

Section D: Abstract Classes vs Interfaces

Exercise 7: Abstract Class

Create an abstract class `Shape` with:

- An abstract method `area()`
- A concrete method `display()`

Create subclasses `Circle` and `Rectangle` and implement the `area()` method.

Exercise 8: Interface vs Abstract Class (Theory + Code)

a) List **four differences** between interfaces and abstract classes.
b) Convert the `Shape` abstract class into an interface and modify the subclasses accordingly.

Section E: Applied / Scenario-Based Exercises

Exercise 9: Real-World Modeling

Design a system for a **University Management System** using OOP concepts:

- Create an interface `UserActions`
- Classes: `Student`, `Lecturer`, `Administrator`
- Each class should implement the interface differently

Exercise 10: Marker Interface (Advanced)

Create a marker interface called `Auditable`.

- Implement it in a class `Transaction`
- Write a method that checks if an object implements `Auditable` and prints `"Auditing enabled"`

Section F: Mini Practical Assignment

Exercise 11: Mini Project (CAT-Level)

Design a **Library Management System** using:

- At least **3 classes**
- At least **1 interface**
- Encapsulation (private attributes)
- Constructor overloading

Submit:

- Java source code
- Brief explanation of how OOP principles were applied

NB: Use Java How to program as your reference book