Advanced Programming

From Software Engineering to OOP in Java

Instructor: Ali Najimi

Author: Hossein Masihi

Department of Computer Engineering

Sharif University of Technology

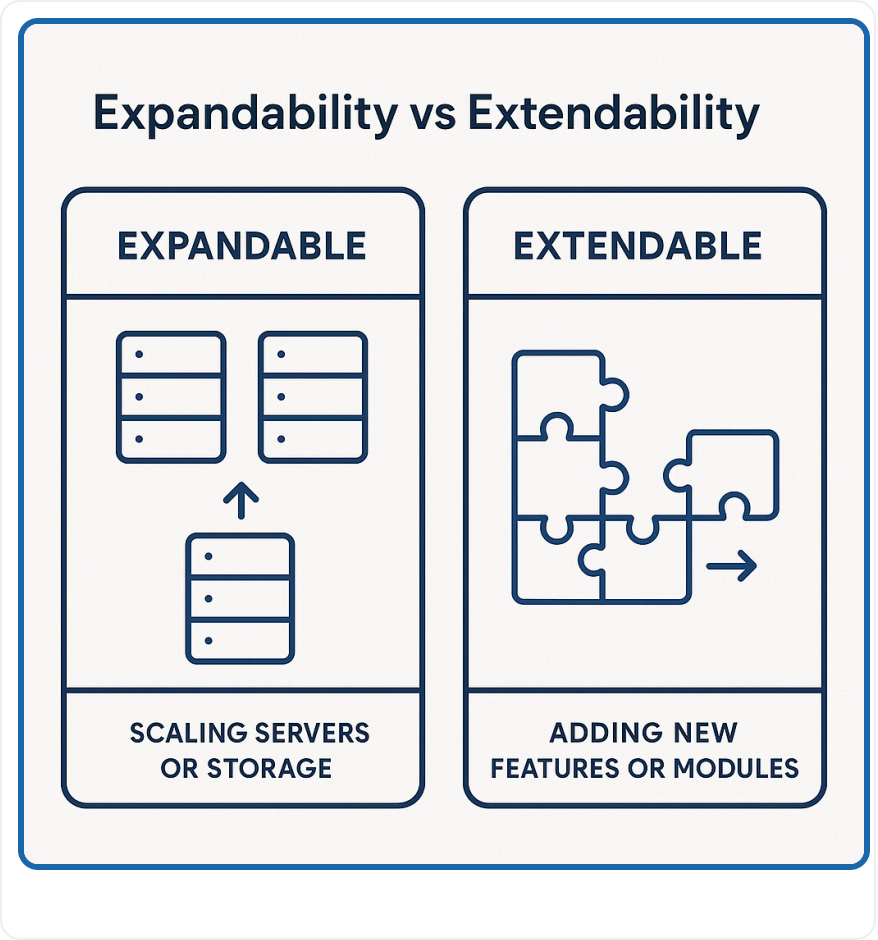
Fall 2025

Table of Contents

Expandable vs Extendable

- **Expandable:**
  Ability to grow in **capacity or scale**
  → Example: adding more memory, database nodes, or cages to the zoo.
- **Extendable:**
  Ability to grow in **functionality**
  → Example: adding new animal categories or new modules without rewriting old code.

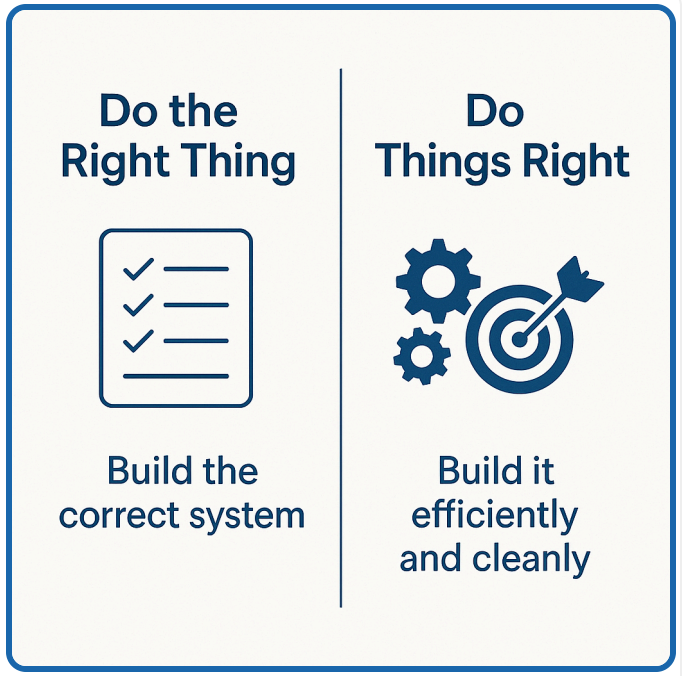
> Good software is both expandable and extendable.



Do the Right Thing vs Do Things Right

Principle	Focus
Do the Right Thing	Build a system effectively
Do Things Right	Build it efficiently, and cleanly

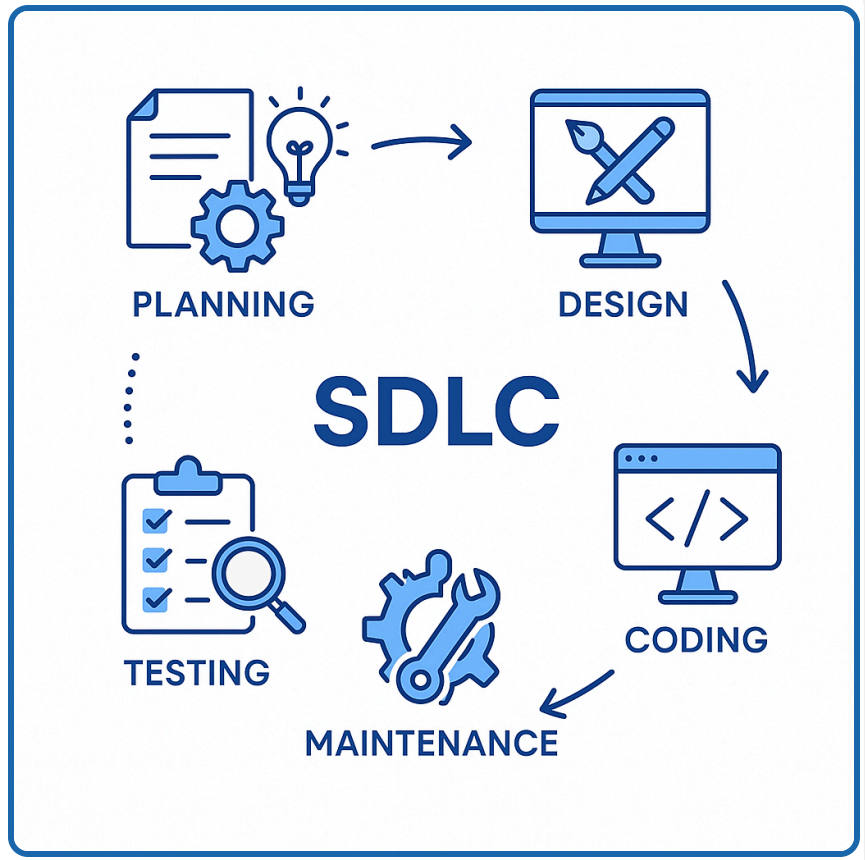
> Example:
> `Do the Right Thing:` Create a Zoo system: manages animals
> `Doing it Right:` Using OOP design, have testing, documentation.



Software Engineering

- Systematic approach to software design and construction.
- Key elements:
  - Process models (like **SDLC**)
  - Documentation and design
  - Quality control and testing
  - Maintenance and scalability

> The goal: to make software **reliable**, **maintainable**, and **evolvable**.



Applying SDLC: Overview

- SDLC defines **how** software evolves from idea to deployment.
- We'll now apply it step-by-step to a real system:
  The Zoo Management System.

Phase	Description
1	Requirement Analysis
2	Design (UML, Architecture)
3	Implementation (Java / OOP)
4	Testing
5	Deployment & Maintenance

Case Study — Zoo Management System

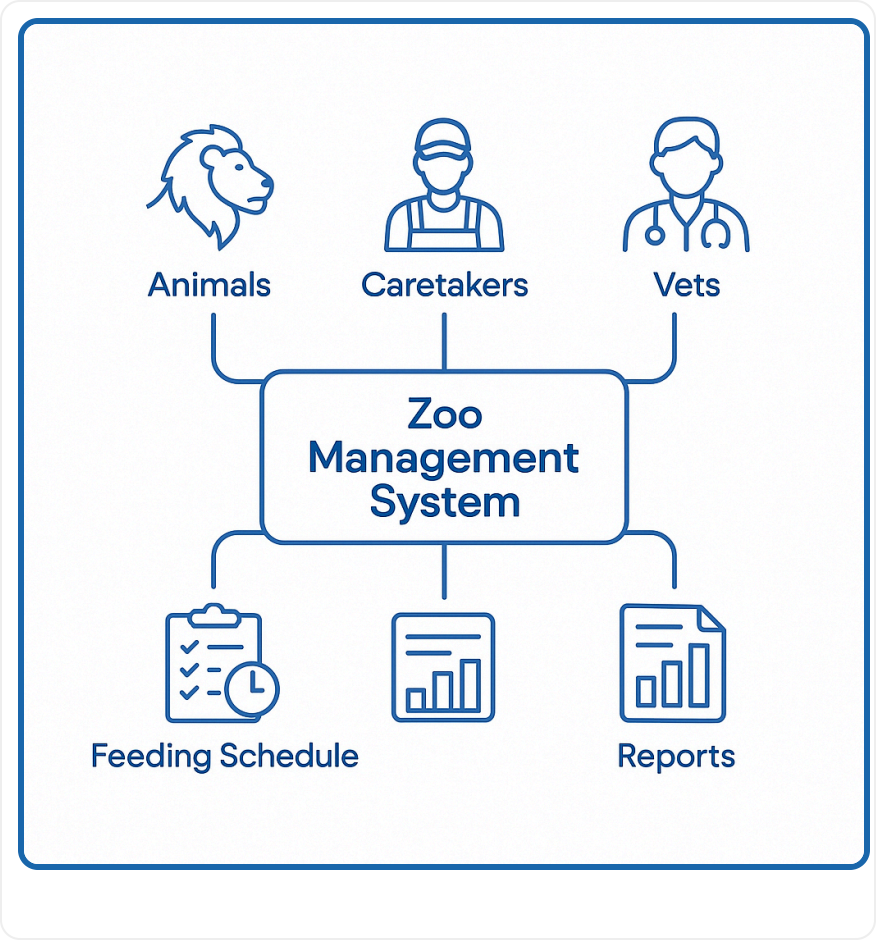
Problem Definition

A zoo needs a system to:

- Track animals, health, and feeding schedules
- Manage caretakers and vets
- Generate daily and monthly reports

Goal

To design and implement a **modular**, **extendable**, and **object-oriented** solution.



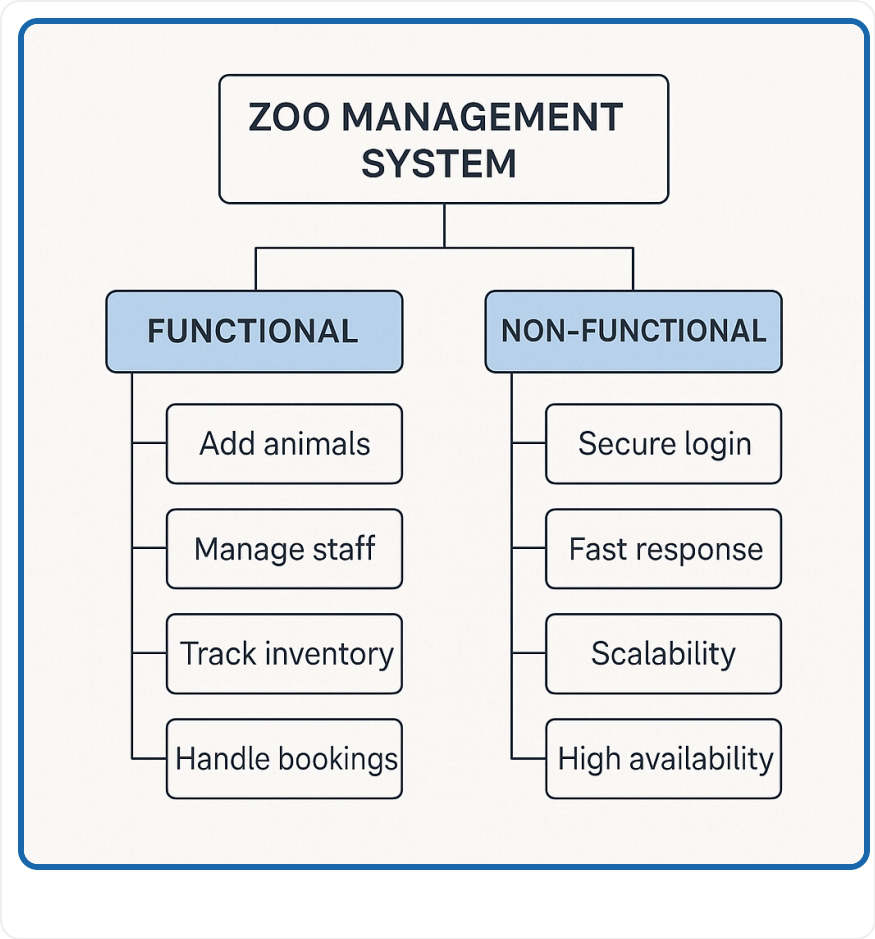
1. Requirement Analysis

Functional Requirements

- Create - Read - Update - Delete
- For Animals, Storage, etc.
- Record feedings and treatments
- Manage staff and shifts

Non-Functional Requirements

- Secure login for staff
- Fast response time (<1s)
- Future-proof, extendable design



2. System Design — UML & Architecture

Architecture Layers

- **UI Layer:** Staff Dashboard
- **Logic Layer:** ZooManager (Java)
- **Data Layer:** MySQL / PostgreSQL

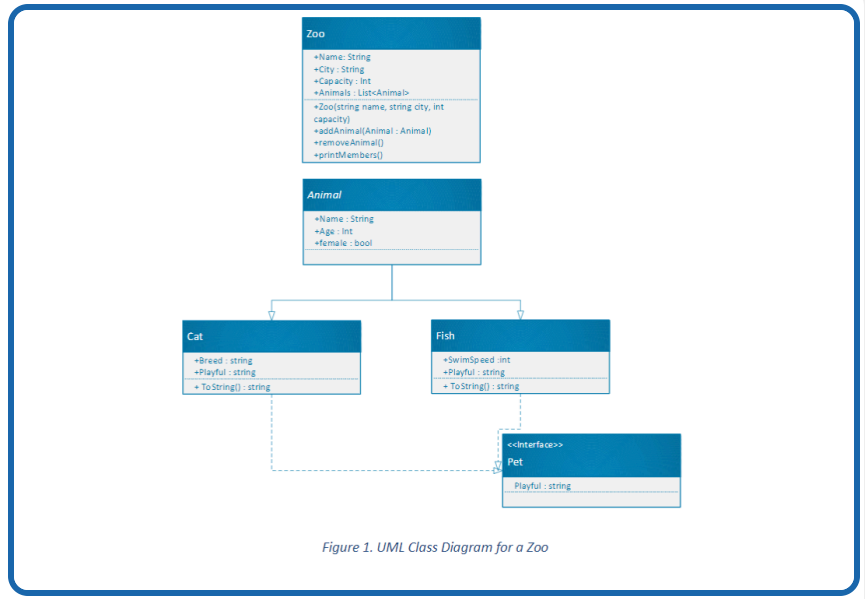
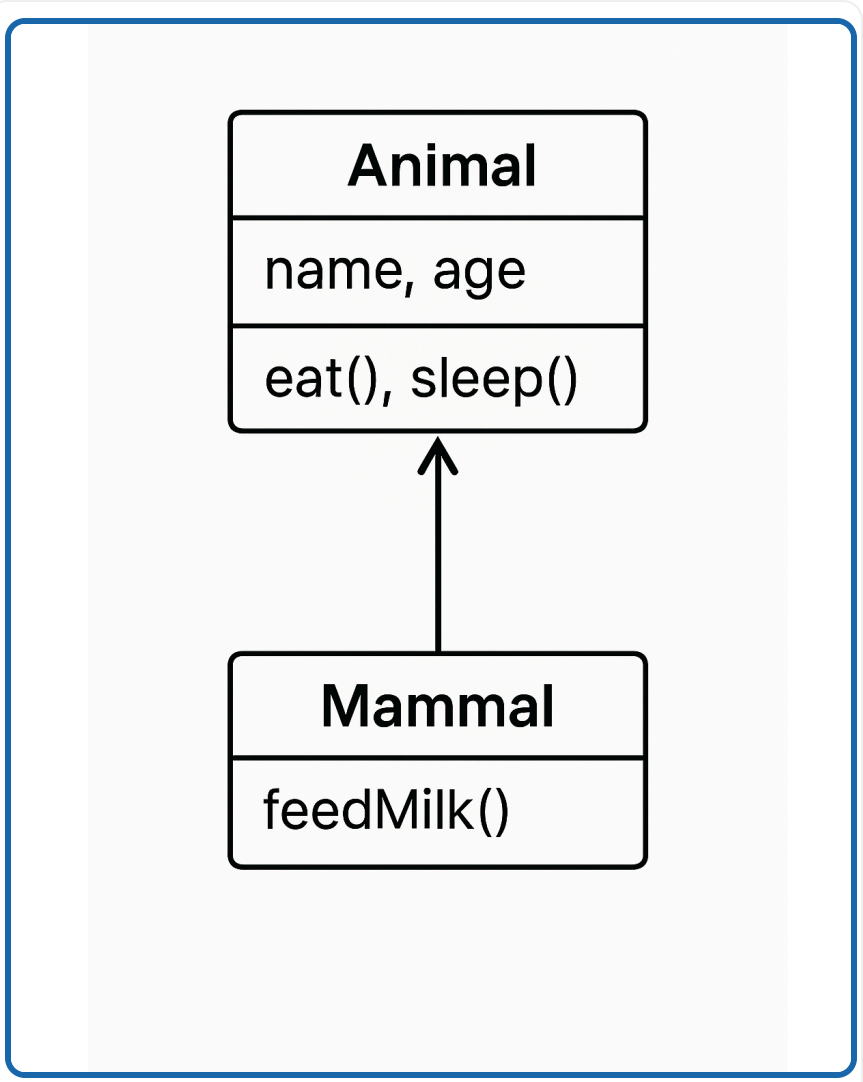


Figure 1. UML Class Diagram for a Zoo

2. System Design — UML & Architecture

UML Class Diagram

```
+----------------+
| Animal         |
|----------------|
| name, age      |
|----------------|
| eat(), sleep() |
+----------------+
        ▲
        |
+-------------+
| Mammal      |
|-------------|
| feedMilk()  |
+-------------+
```

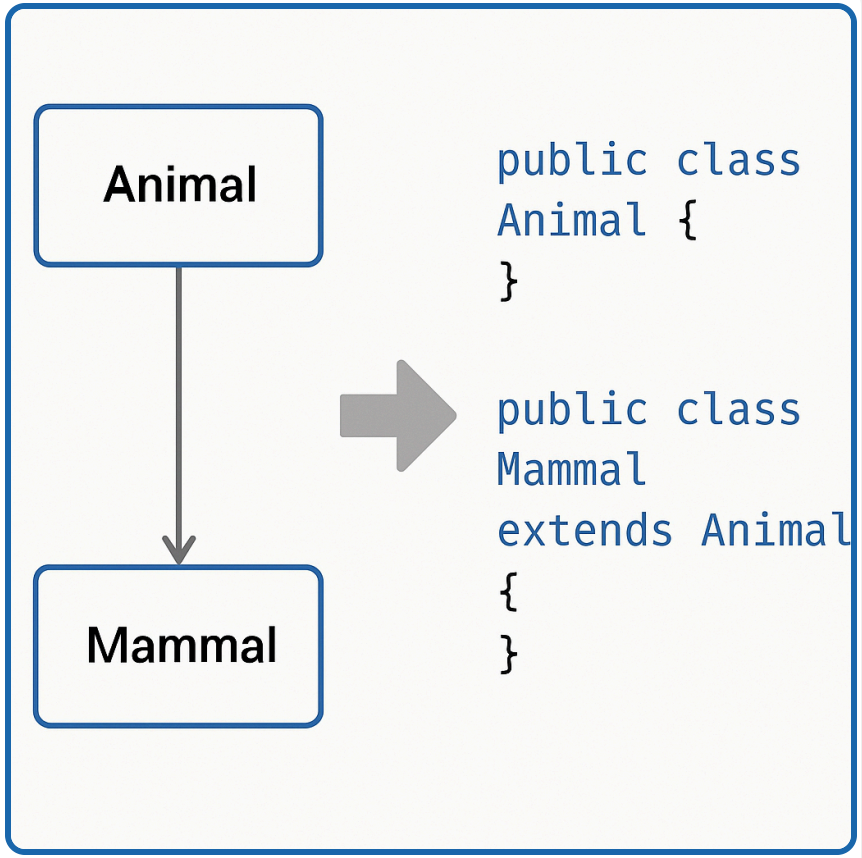


3. Implementation — From UML to Java

```java
public class Animal {
  private String name;
  private int age;

  public Animal(String name, int age) {
    this.name = name;
    this.age = age;
  }

  public void eat() {
    System.out.println(name + " is eating...");
  }
}
```

3. Implementation — From UML to Java

```java
public class Mammal extends Animal {
    public Mammal(String name, int age) {
        super(name, age);
    }

    public void feedMilk() {
        System.out.println(getName() + " is feeding milk.");
    }
}
```



4. Testing Phase

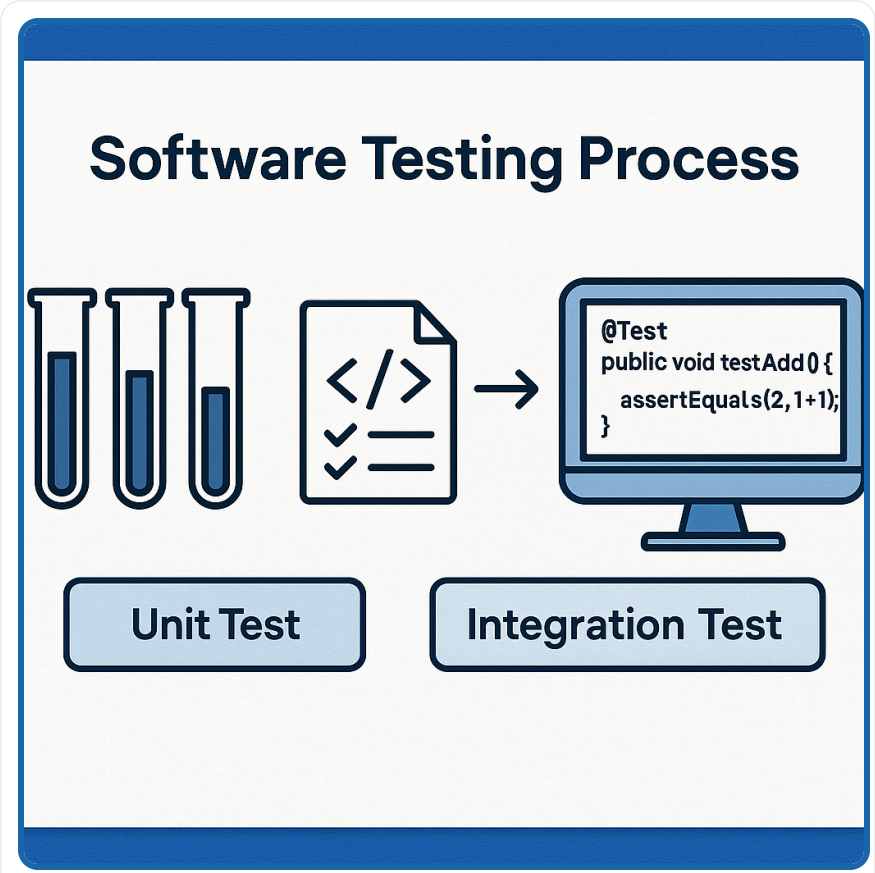
Unit Testing Example

```
@Test
public void testEat() {
    Animal lion = new Animal("Lion", 5);
    lion.eat(); // Expected: "Lion is eating..."
}
```

Integration Tests

- Add animal → Assign caretaker → Generate report
- Check data persistence and relations

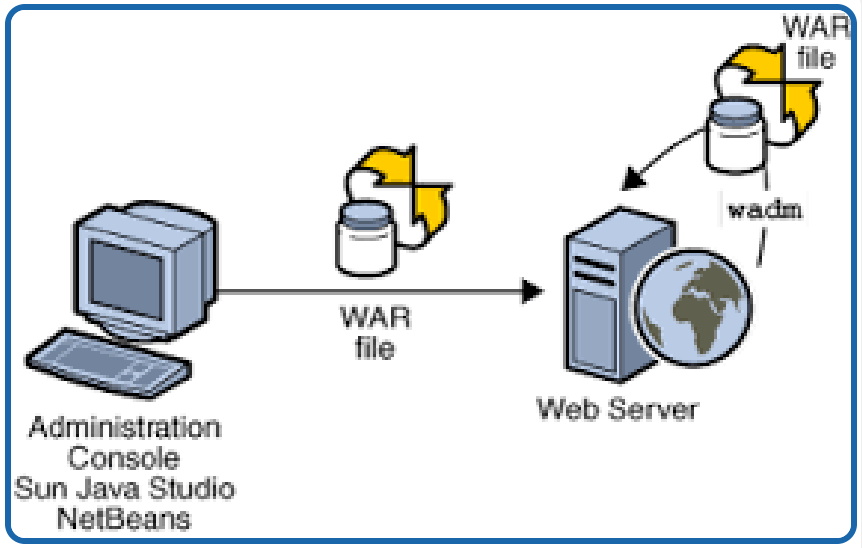
> Testing validates: correctness and integration.



5. Deployment and Maintenance

- Package the Java application into a `.jar`
- Deploy on a local server or cloud instance
- Connect to a MySQL/PostgreSQL database
- Future maintenance:
  - Add new animal subclasses
  - Extend report generation
  - Improve UI responsiveness

> The system remains extendable without rewriting core logic.



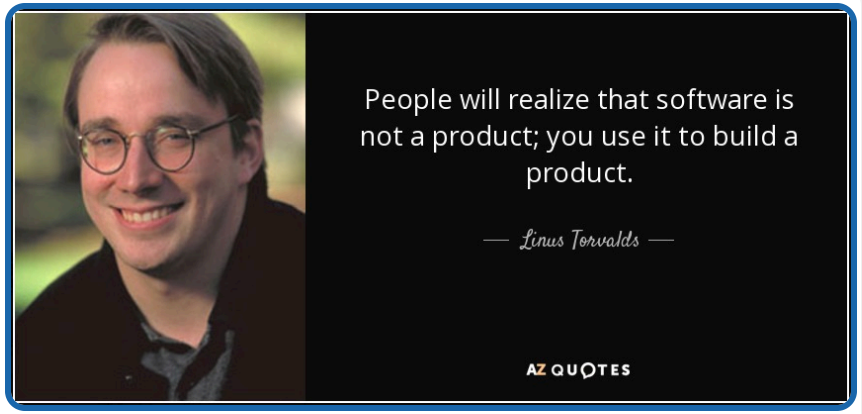
Final Discussion

Concept	In Practice
Expandable	More animals, cages, or DB capacity
Extendable	New modules like visitor management
Do the Right Thing	Address the zoo’s real operational needs
Do Things Right	Use OOP, SDLC, and clean Java design
SDLC Applied	Zoo System built step by step, tested, deployed

> Software engineering and OOP are two sides of the same goal: building reliable, scalable, and maintainable systems.

Thank You

From Software Engineering to OOP in Java



Advanced Programming – Fall 2025 – Sharif University of Technology