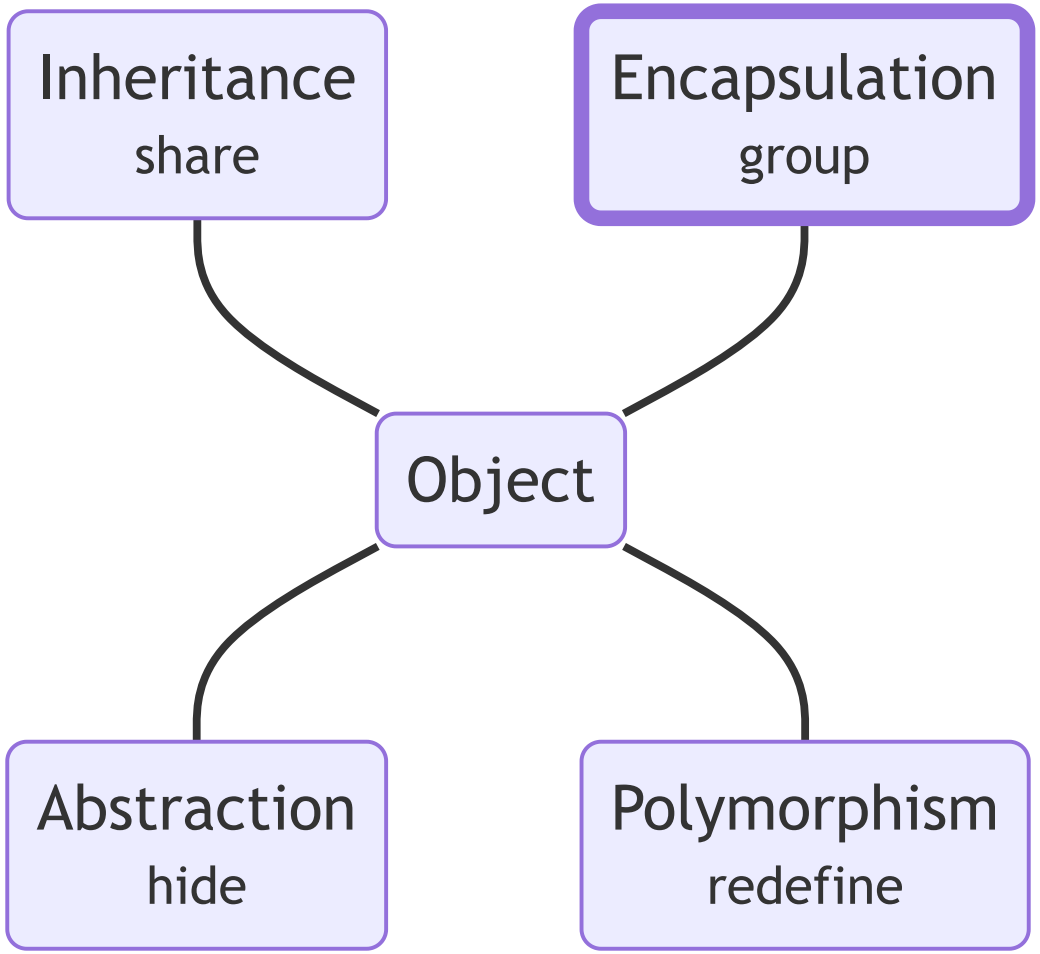
SE 350 - Object-Oriented Software Development

Object-Oriented Programming Principles: Encapsulation

Instructor: Stefan Mitsch


Inheritance
share
Encapsulation
group
Object
Abstraction
hide
Polymorphism
redefine

Learning Objectives

- Understand information and implementation hiding
- Understand defensive programming techniques

Encapsulation

- Encapsulate data and code in a single unit
- What should be encapsulated: anything that may change
- Why use encapsulation:
  - Separate interface from implementation
  - Control data access
  - Clean and documented interface for client programmers
- Information hiding: private members and public access
- Implementation hiding: interface and implementing class

Summary

- Abstraction: **what** functionality
- Encapsulation: **how** to achieve functionality
- Use encapsulation to hide implementation details
- Defensive programming: clearly document and check method contracts

```java
public abstract class Animal {
  // Encapsulation
  private LocalDate birthdate;
  public Animal(LocalDate birthdate) {
    if (birthdate.after(LocalDate.now())) {
      throw new IllegalArgumentException("...");
    }
    this.birthdate = birthdate;
  }
  // Access-control: read-only age
  public int getAge() {
    return Period.between(
        birthdate,
        LocalDate.now()
      ).getYears();
    }

  // Abstraction
  public String sound();
}

public abstract class Pet extends Animal {
  // Encapsulation
  private String name;

  public final String getName() { return name; }
  public final void setName(String name) {
    if (name.isEmpty()) throw new IllegalArgumentException("...");
    this.name = name;
  }
}

public class Dog extends Pet {
  @Override
  public String sound() { return "bark"; }
}
```

Exercises

- Create a class to represent a person with their name, birthday, and address
- Implement methods to access the data (read-only birthday, modifiable name and address)
- Implement a method to compute the age of the person
- Use defensive programming to protect data integrity