SE 350 - Object-Oriented Software Development

Programming Paradigms

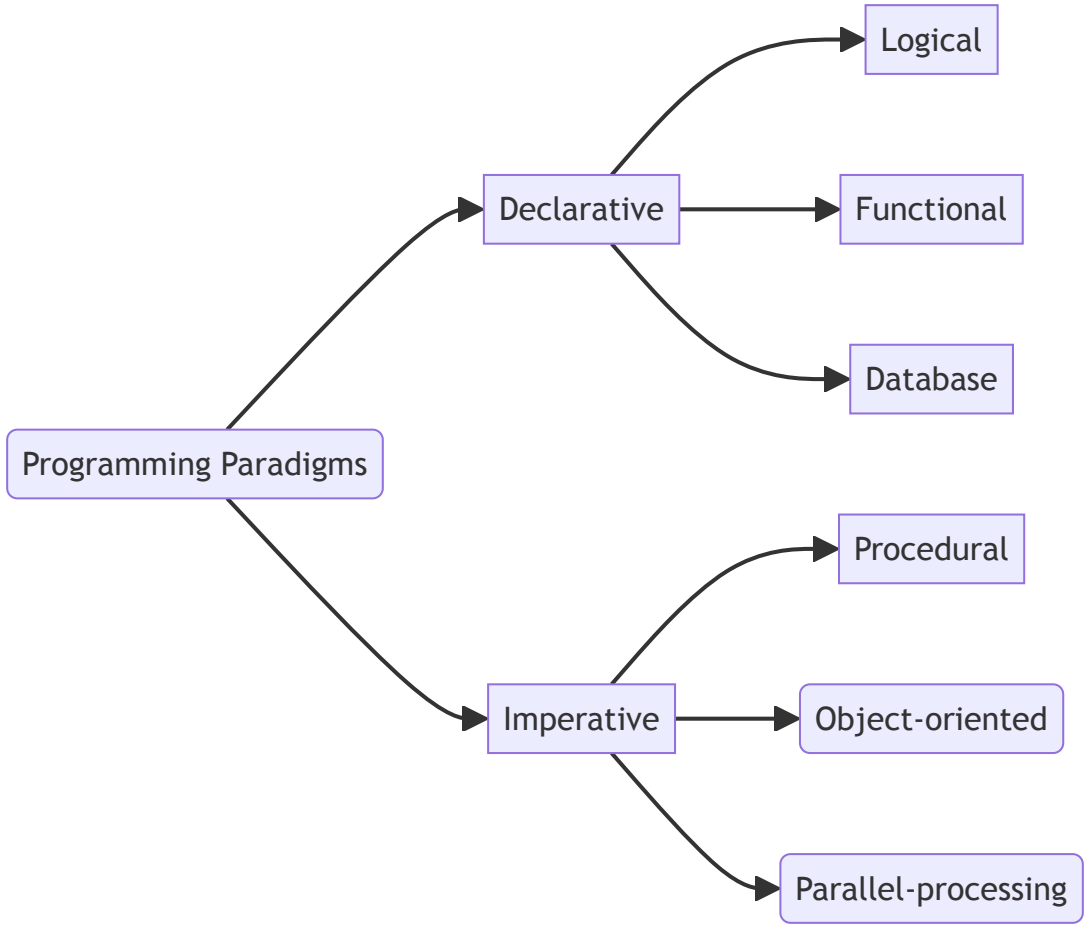
Instructor: Stefan Mitsch

Learning Objectives

- Distinguish imperative from declarative programming
- Understand the main features of procedural programming and functional programming

Programming Paradigms

- Imperative: how to execute logic
- Declarative: what to compute



? How Many Programming Languages Are There?

- TIOBE
- Wikipedia
- FOLDOC
- The Language List

Imperative Programming

- How to execute program logic
- Defines control flow as sequence of statements
- Language examples
  - Procedural: C, Pascal
  - Object-oriented: C++, Java, PHP, Python, Ruby
  - Parallel-processing: NESL

```
int total = 0;
for (int i = 0; i <= 5; i++) {
    total += i;
}
System.out.println(total);
```

Declarative Programming

- What to execute
- Defines program logic instead of control flow
- Language examples
  - Logic: Prolog
  - Functional: Lisp, Scala, Julia, Haskell, R, Matlab
  - Database: SQL

```
select * from users
```

```
def fib(n: Int) : Long = n match {
  case 0 | 1 => n
  case _ => fib(n-1) + fib(n-2)
}
```

Procedural Programming

- Control flow in program steps
- Decomposes software top to bottom into procedures
- Subroutines or functions
- Features
  - Simplicity in small-scale programs
  - Easy to match with execution in processor
  - Mutable data

```
program Fibonacci;
function fib(n: Integer): Integer;
var a: Integer = 1;
    b: Integer = 1;
    f: Integer;
    i: Integer
begin
    if (n=1) or (n=2) then fib := 1
    else begin
        for i := 3 to n do begin
            f := a+b;
            b := a;
            a := f;
        end;
        fib := f;
    end;
end;
```

Functional Programming

- Evaluate mathematical functions
- Functions: pure without side effects, side-effecting, higher-order
- Features
  - Clear responsibilities
  - Immutable data
  - Contracts and correctness

```
def fibonacci(n: Int): Int =
    if n <= 1 then n
    else fibonacci(n-1) + fibonacci(n-2)
```

Summary

- Imperative programming defines execution steps (program flow)
- Declarative programming defines what to compute instead of how
- Procedural programming decomposes a problem into hierarchically structured procedures
- Functional programming decomposes a problem into mathematical functions, emphasizes data immutability

Exercises

- Implement a function in Java that computes the Fibonacci numbers
  - In an imperative style
  - In a recursive (functional) style
  - Compare the performance of the two functions on increasingly large inputs
  - Implement a tail-recursive function that does not have a large performance penalty