



SWE 205: Introduction to Software Engineering

Lecture 16

Programming Paradigms

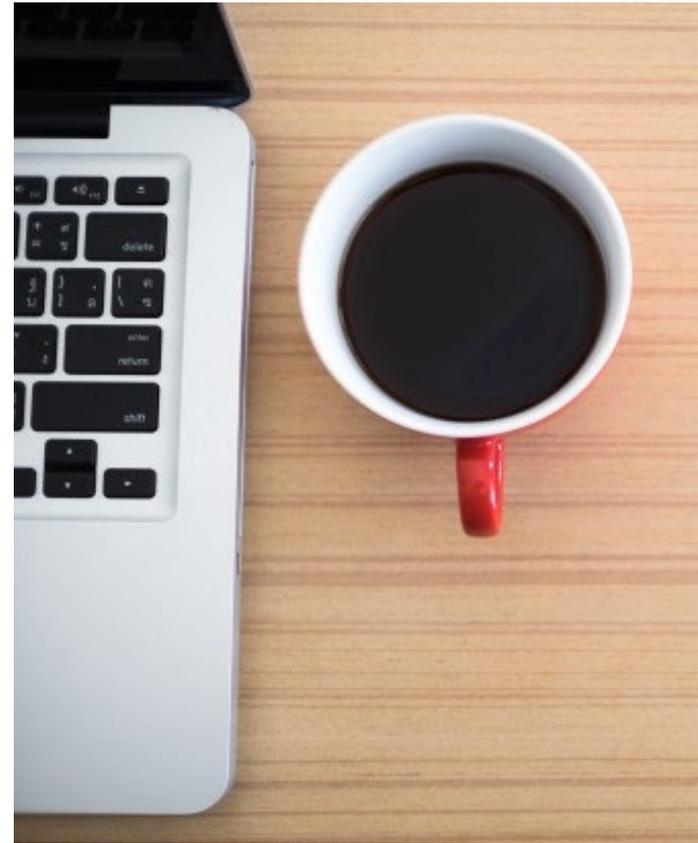
Procedural and Object Oriented

Course Topics

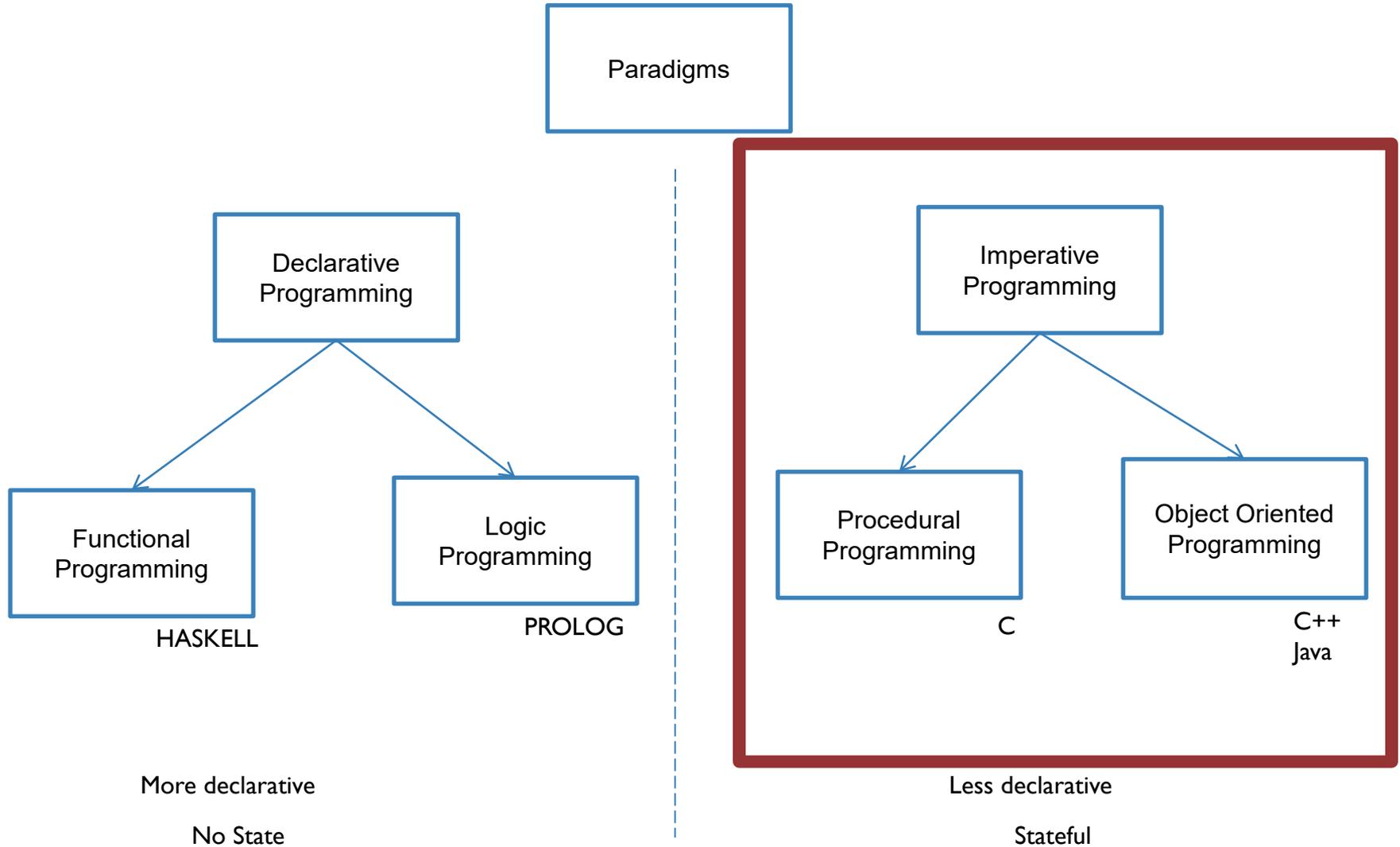
- ~~Introduction~~
- ~~Software Process Models~~
- ~~Requirements Engineering~~
- ~~Modeling~~
- Software Construction Techniques
- Testing
- Project Management
- Refactoring
- Ethical Issues

Lecture Objectives

- ✓ Procedural programming
- ✓ Object-oriented programming



Programming Paradigms



Declarative Vs Imperative

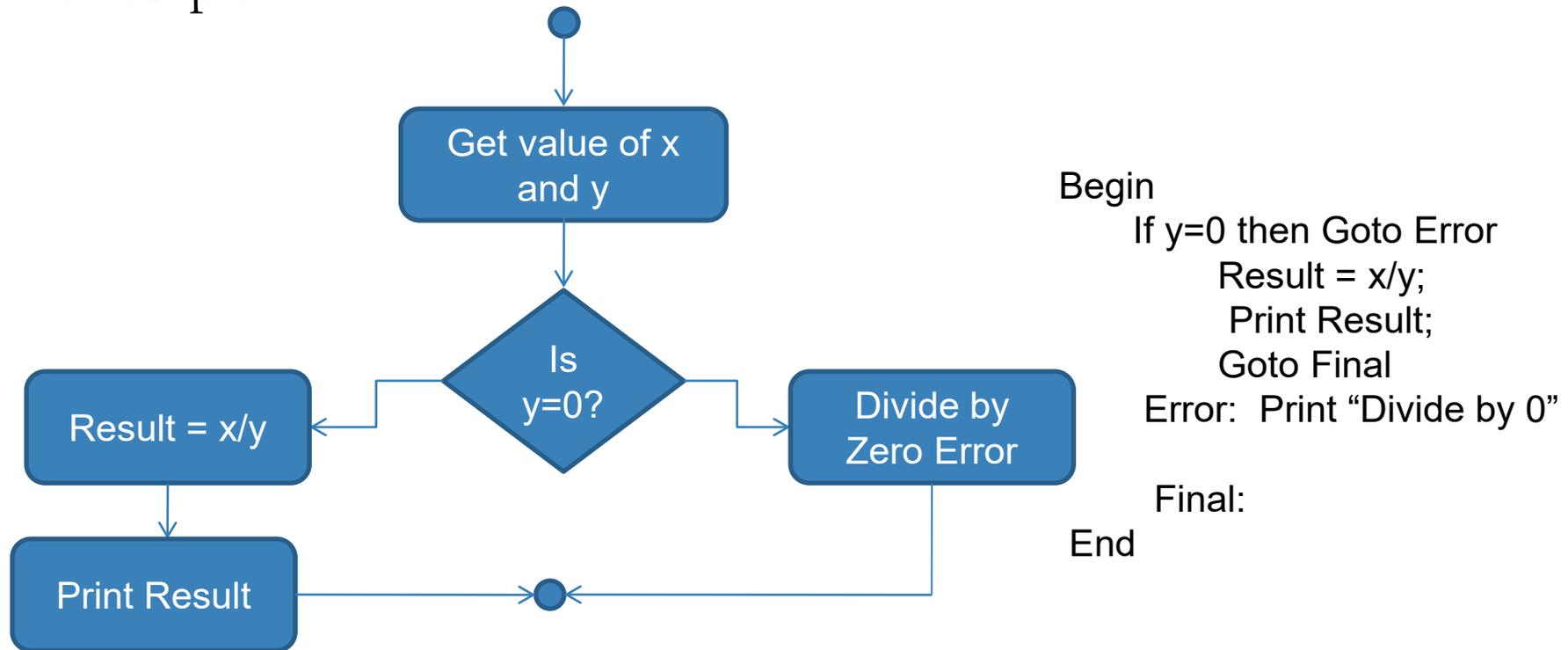


- **Imperative programming** is a programming paradigm that describes computation in terms of statements that change a program state. Imperative programs define sequences of commands for the computer to perform.

In an imperative program, we tell the computer 'how' we want to do a certain task.

Imperative Programming

- Activity Diagrams and Pseudo Code produced during the design phase can be directly used to generate an imperative program.
- Example



A bit of history: Basic

```
10 dim i
20 i = 0
30 i = i + 1
40 if i <> 10 then goto 90
50 if i = 10 then goto 70
60 goto 30
70 print "Program Completed."
80 end
90 print i; " squared = "; i * i
100 goto 30
```

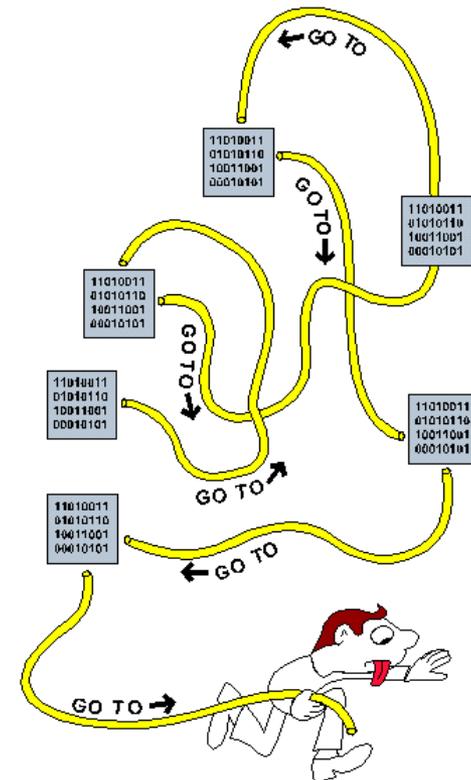
Procedural Programming

- Procedural programming is imperative programming in which the statements are structured into procedures (also known as subroutines or functions).

From Computer Desktop Encyclopedia
© 1998 The Computer Language Co., Inc.

- Using GOTO and label causes
“Spaghetti Code”

- To avoid this, we structure the program into sub-functions.



```
main
{
```

```
{
```

```
    Result = Add
(x,y);    }
```

```
Else If Op=='-'
{
    Result = Sub (x,y);
}
```

```
Else If Op=='/'
{
    Result = Div (x,y);
}
```

```
Else If Op=='*'
{
    Result = Mul(x,y);
}
```

```
Else Print "Illegal Operator"
```

```
Begin
```

```
Accept the Value of X and Y
```

```
int Add(x,y)
Accept the Value for Operation
{ return (x+y) }
```

```
If Op=='+' Goto Add
```

```
int Sub(x,y)
If Op=='-' Goto Subtract
{ return (x-y) }
```

```
If Op=='/' Goto Divide
```

```
int Div(x,y)
If Op=='*' Goto Multiply
{ return (x/y) }
```

```
Print "Wrong Operator"
```

```
int Mul(x,y)
Goto Final
{ return (x*y) }
```

```
Add: Result = x+y      Goto Result
```

```
Subtract: Result = x-y  Goto Result
```

```
Divide: Result = x/y    Goto Result
```

```
Multiply: Result = x*y  Goto Result
```

```
Result: Print Result
```

```
Final:
```

```
End
```

Begin

```
Accept the Value of X and Y
Accept the Value for Operation
If Op=='+' Goto Add
If Op=='-' Goto Subtract
If Op=='/' Goto Divide
If Op=='*' Goto Multiply
    Print "Wrong Operator"
    Goto Final
Add: Result = x+y    Goto Result
Subtract: Result = x-y    Goto Result
Divide: Result = x/y    Goto Result
Multiply: Result = x*y    Goto Result
Result: Print Result
Final:
```

End

main {

```
Accept the Value of X and Y
Accept the Value for Operation
If Op=='+'    Result = Add (x,y);
Else If Op=='-'    Result = Sub (x,y);
Else If Op=='/'    Result = Div (x,y);
Else If Op=='*'    Result = Mul(x,y);

Print Result

int Add(x,y){
    return (x+y)
}
int Sub(x,y){
    return (x-y)
}
int Div(x,y){
    return (x/y)
}
int Mul(x,y){
    return (x*y)
}
}
```

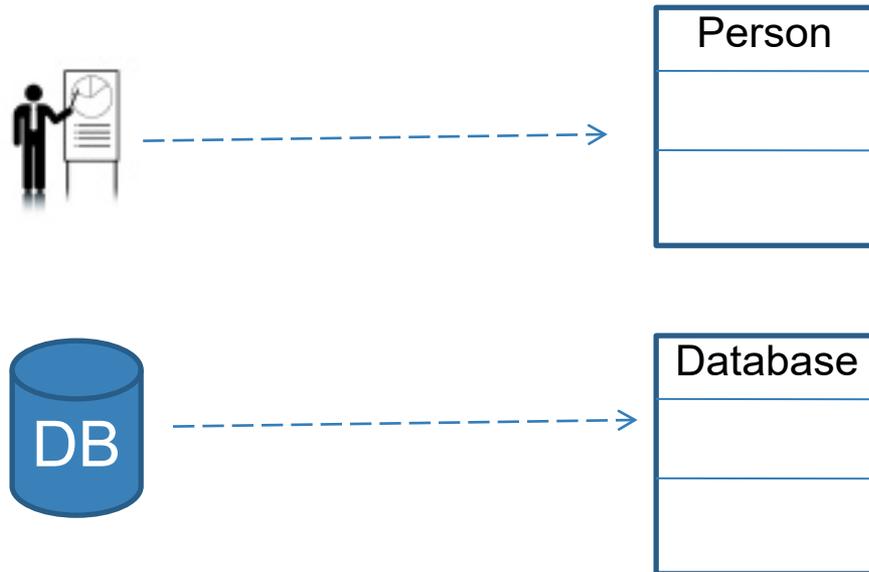
Procedural Programming



- Advantages
 - Modularity
 - Re-usability
 - The ability to re-use the same code at different places in the program without copying it.
 - Avoids Spaghetti Code

Object Oriented Programming Paradigm

Motivation



Real World Entities match to **Objects**

Advantages of Object-Oriented Paradigm

■ Encapsulation

- Used to hide the values or state of an object inside a class, preventing unauthorized parties' direct access to them. Publicly accessible methods are generally provided in the class (so-called getters and setters) to access the values, and other client classes call these methods to retrieve and modify the values within the object.

```
public class Employee {
    private BigDecimal salary = new BigDecimal(50000.00);

    public BigDecimal getSalary() {
        return salary;
    }

    public static void main() {
        Employee e = new Employee();
        BigDecimal sal = e.getSalary();
    }
}
```

Advantages of Object-Oriented Paradigm



■ Polymorphism

- ability of objects belonging to different data types to respond to method calls of methods of the same name, each one according to an appropriate type-specific behavior.
 - Two forms of Polymorphism:
 - Method overriding: the child class can use the OOP polymorphism concept to override a method of its parent class. That allows a programmer to use one method in different ways depending on whether it's invoked by an object of the parent class or an object of the child class.
 - Method overloading: a single method may perform different functions depending on the context in which it's called. That is, a single method name might work in different ways depending on what arguments are passed to it.
- 

Advantages of Object-Oriented Paradigm



- Reusability
 - Objects are potentially reusable components.
- Inheritance
 - It is an abstraction mechanism which may be used to classify entities. A sub-class inherits the attributes and operations from its super class and may add new methods or attributes of its own.

Procedural Vs Object Oriented

Procedural	Object-Oriented
functions	methods
modules	objects
argument	message
variable	attribute



Programming Wisdom @CodeWisdom · 16 Nov 2017



"The object-oriented version of spaghetti code is, of course, 'lasagna code'. Too many layers." - Roberto Waltman

Key Points



- Both procedural and object-oriented programming aim to
 - Better modularity
 - Better reuse
- Object-oriented programming adds several techniques such as inheritance and polymorphism to make programming even easier

References



- Ian Sommerville, “Software Engineering”, 10th Edition, Addison-Wesley, 2015.
- Timothy C. Lethbridge and Robert Laganière, “Object-Oriented Software Engineering: Practical Software Development using UML and Java”, 2nd Edition, McGraw Hill, 2001.
- R. S. Pressman, Software Engineering: A Practitioner’s Approach, 10th Edition, McGraw-Hill, 2005.