Prana A. Rusdi
prana.rusdi@gmx.de

# Algoritma dan Pemrograman

# Objectives

- To understand the logic of programming
- To be able to draw flowchart diagram
- To understand the basic of programming
- To be able to implement the program

# Definition

- “A set of well-designed instructions or procedures carried out to accomplish a specific task”

# Example

- Creating an e-mail account
  - Clicking on a link
  - Input username and password
  - Input personal data (optional)
  - Verifikasi
  - License agreement

# Example

- Buying an online ticket
  - Schedule selction
  - Data input
  - Confirmation
  - Verification
  - Payment
  - Printing ticket

# Algorithm

- To produce a well-structured scheme
- To take system from one state to another
- Motto: “*Keep it simple*”
- Often described in pseudo-code
- To produce an effective and efficient software

# Algorithm

- Pseudo-code is an artificial and informal language that helps programmers develop algorithms.
- Example
  - If a student's grade is greater than 60, print pass, else print fail
  - If grade > 60

    print “Pass”;

    Else

    print “Fail”

# Components

- Loop (Iteration & Recursion)
- Condition
- List, Queue & Stack
- Sorting

# Loop

- For

```
for (expression){
/* some code here
}
```

- Example

```
for (i=0; i<5; i++){
 Print (“Hello”);
}
```

# Loop

- While

```
while (expression){
/* some code here
}
```

- Example

```
while (i<5){
 Print(“Hello”);
 i++;
}
```

# Condition

```
if (booleanExpression) {
    statement (s)
}
```

or

```
if (booleanExpression) {
    statement (s)
} else {
    statement (s)
}
```

# Condition

```
public class MainClass {

  public static void main(String[] arg) {
    int a = 0;
    if (a == 0) {
      System.out.println("in the block");
      System.out.println("in the block");
    } else {
      System.out.println("a is not 0");
    }
  }
}
```

# Condition

```
switch (expression) {
case value_1 :

      statement (s);
      break;
case value_2 :
      statement (s);
      break;
   .
   .
   .
case value_n :
      statement (s);
      break;
default:
      statement (s);
}
```

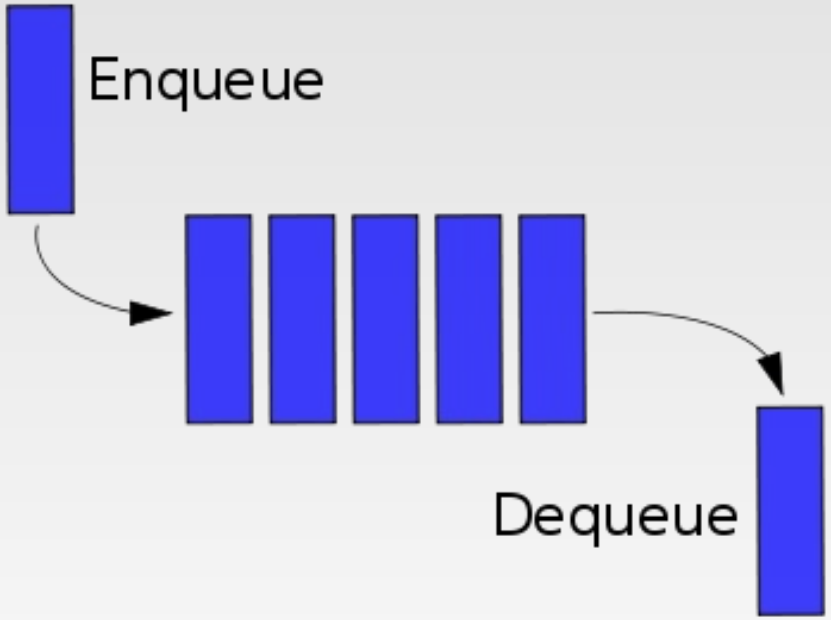
# Condition

```
public class MainClass {

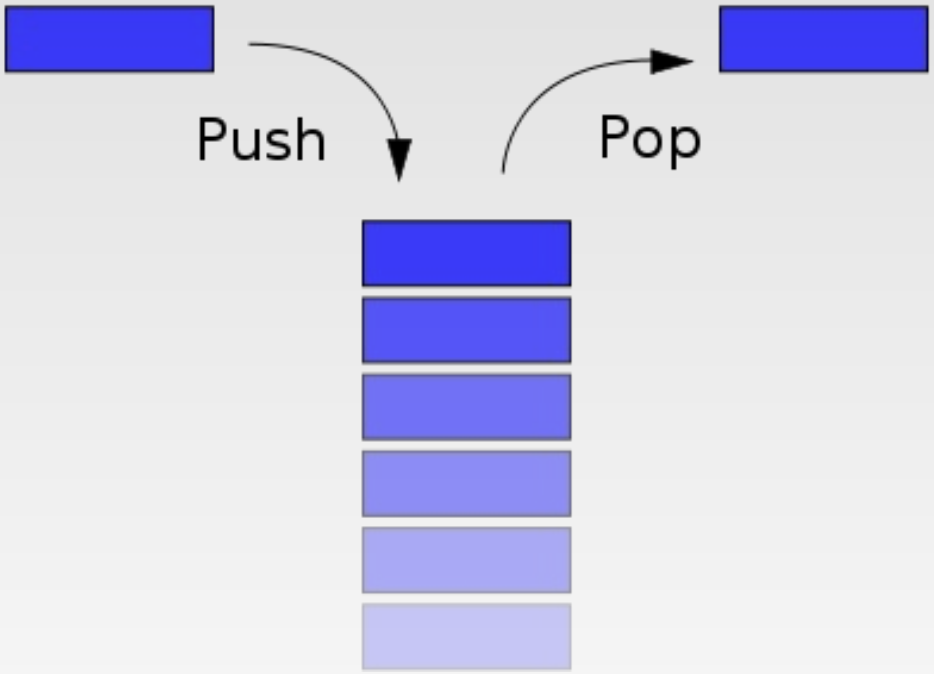
  public static void main(String[] args) {
    int i = 1;
    switch (i) {
    case 1 :
        System.out.println("One.");
        break;
    case 2 :
        System.out.println("Two.");
        break;
    case 3 :
        System.out.println("Three.");
        break;
    default:
        System.out.println("You did not enter a valid value.");
    }
  }

}
```

# Queue



# Stack



# Complexity of Algorithm

- Complexity affects performance
- Perform complexity vs. performance scaling

# Step-by-step Programming

- Anylze problems
- Define solution
- Define sequences
- Draw a flowchart
- Draw a model (UML)
- Develop a software
- Deploy testing

# Problem analysis

- Define root problem
- Problem breakdown
- Define actor(s)
- Define input
- Define output

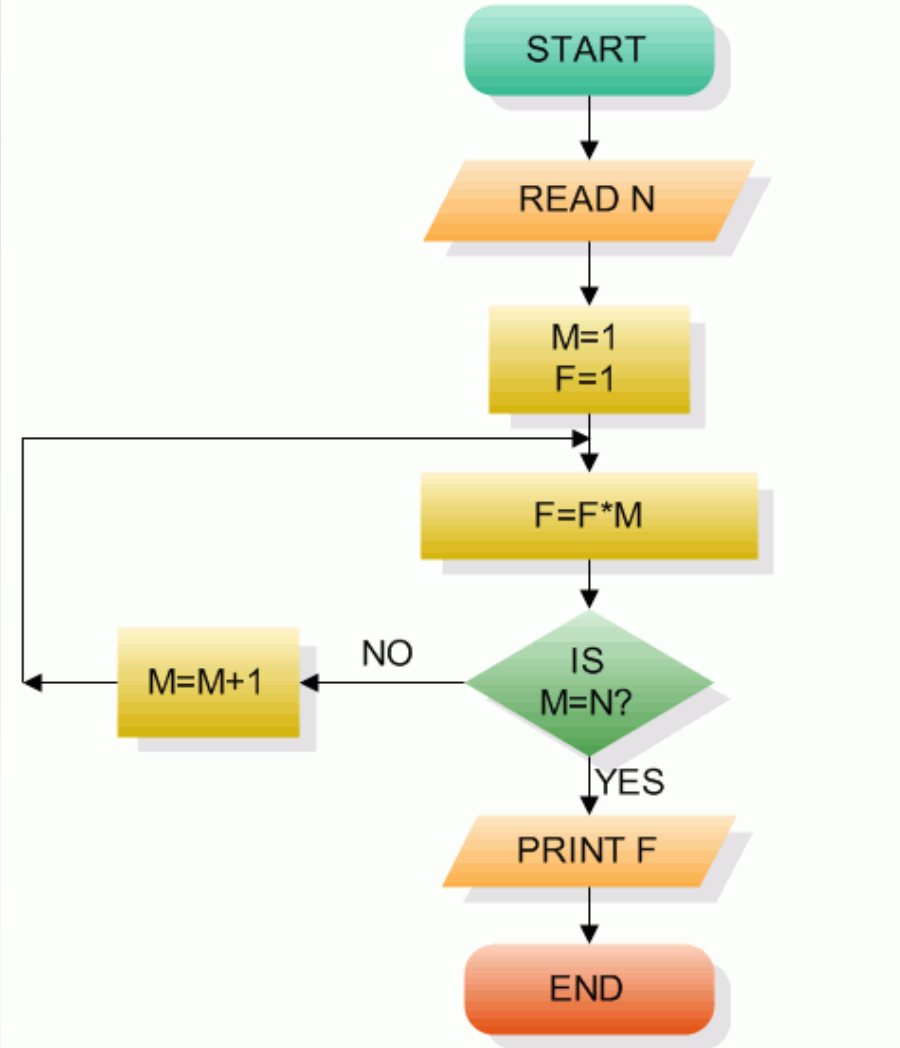
# Defining solution

- List all possibilities
- Evaluate every possibilities
- Define method

# Defining sequence

- Define *starting point*
- Define end-point
- List sequences
- Draw flowchart

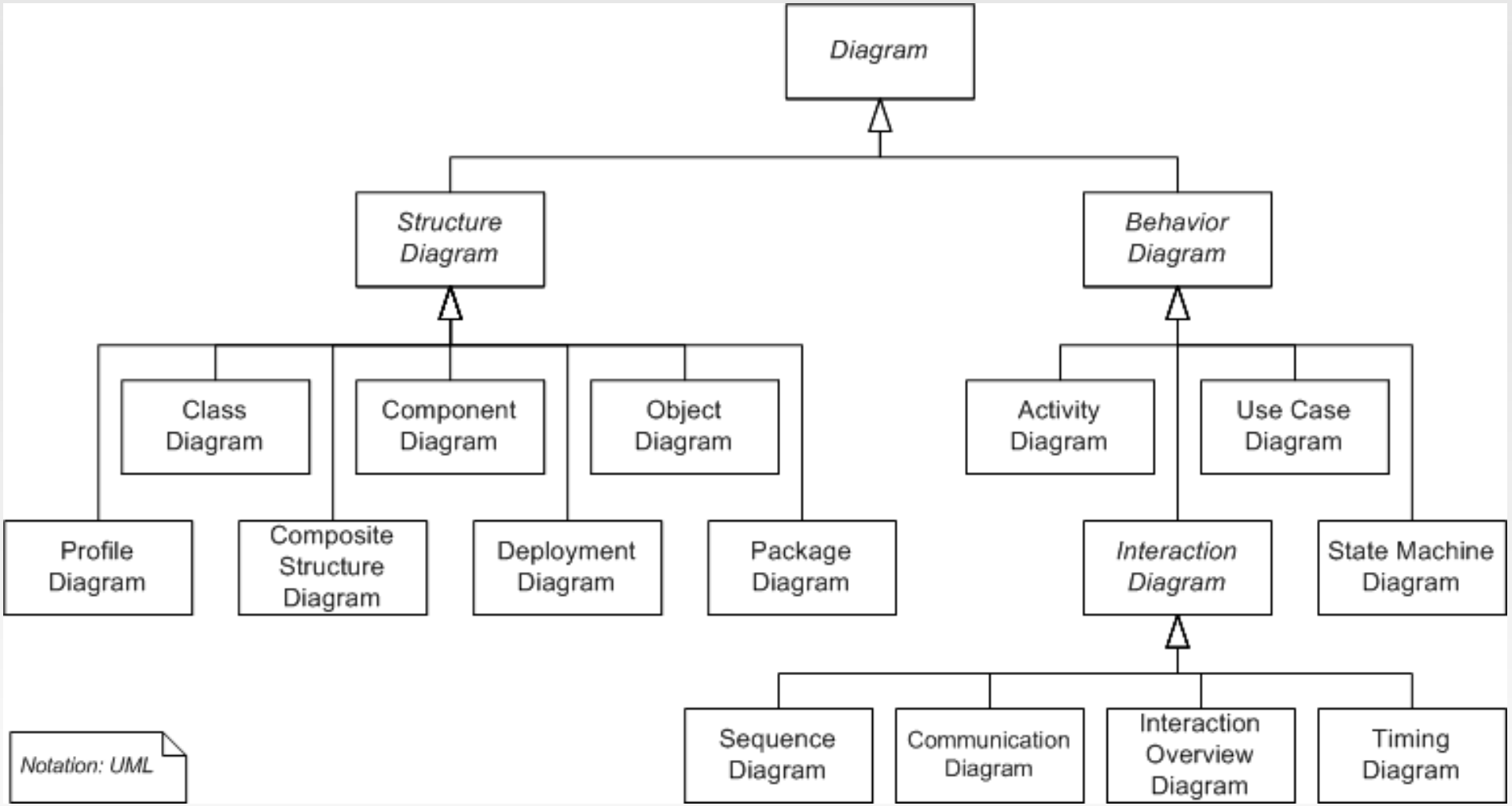
# Flowchart



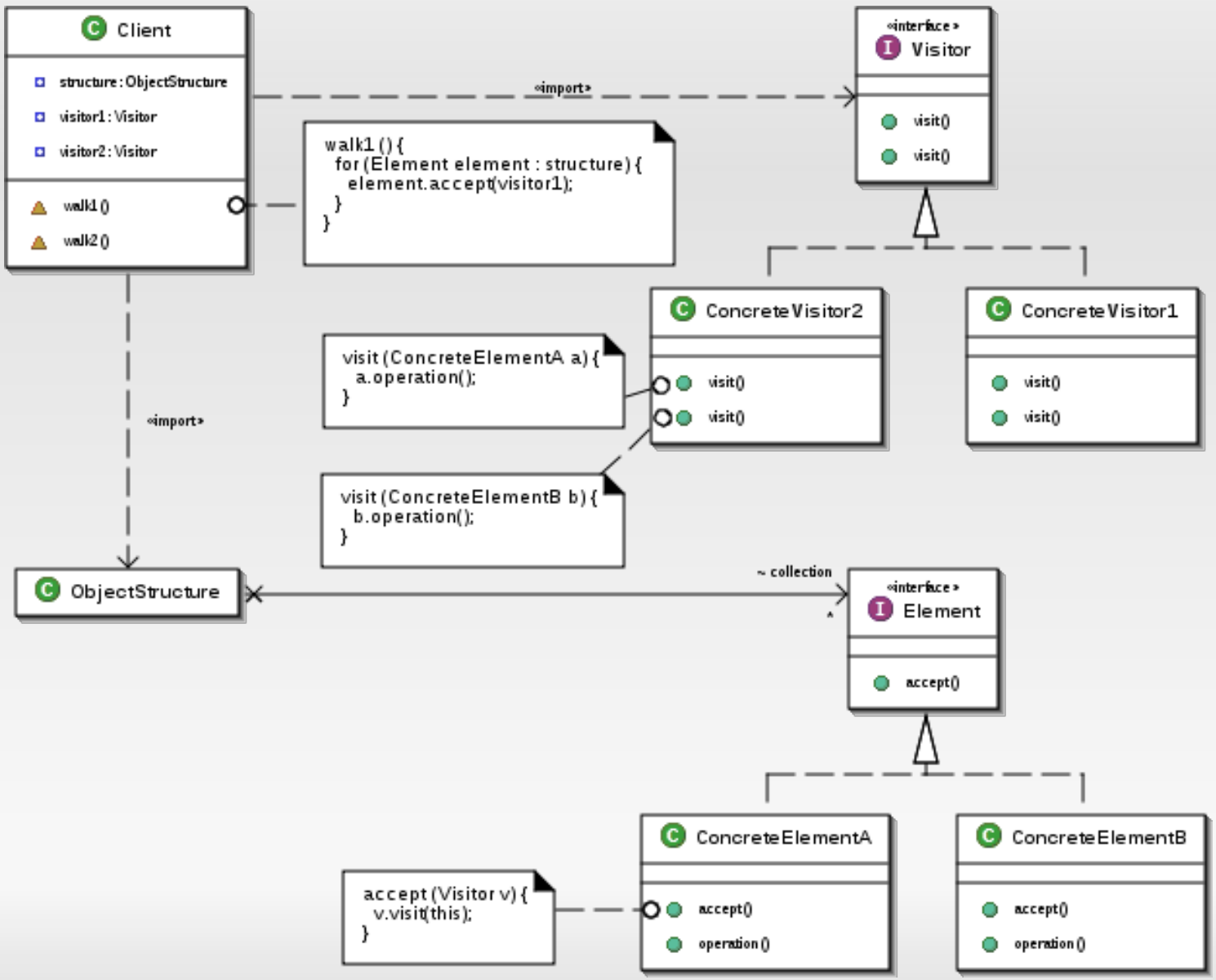
# Model

- “Blueprint of a system consists of interacting objects”
- Basic concept
  - Structure (Actor, class, object, etc.)
  - Behavior (Event, method, message, etc.)
  - Relationship (Aggregation, composition, etc.)

# Diagram


Diagram
Structure Diagram
Behavior Diagram
Class Diagram
Component Diagram
Object Diagram
Activity Diagram
Use Case Diagram
Profile Diagram
Composite Structure Diagram
Deployment Diagram
Package Diagram
Interaction Diagram
State Machine Diagram
Sequence Diagram
Communication Diagram
Interaction Overview Diagram
Timing Diagram
Notation: UML

# UML


Client
structure : ObjectStructure
visitor1 : Visitor
visitor2 : Visitor
walk1 ()
walk2 ()
«import»
walk1 () {
for (Element element : structure) {
element.accept(visitor1);
}
}
«interface»
Visitor
visit()
visit()
ConcreteVisitor2
visit()
visit()
ConcreteVisitor1
visit()
visit()
visit (ConcreteElementA a) {
a.operation();
}
visit (ConcreteElementB b) {
b.operation();
}
«import»
ObjectStructure
~ collection
«interface»
Element
accept()
ConcreteElementA
accept()
operation()
ConcreteElementB
accept()
operation()
accept (Visitor v) {
v.visit(this);
}

# Developing a system

- Consider skill and resource
- Consider the complexity
- Consider the end-user

# Testing

- To ensure that the software works
- To ensure that the software does what it should
- Error finding, bug fixing

# Summary

- Algorithm helps developer to produce an efficient and effective software
- Algorithm often described in pseudo-code
- Algorithm represented in flowchart
- Algorithm's complexity affects performance