Computer Programming

Object-Oriented Programming

Willy Picard
Department of Information Technology
The Poznan University of Economics
<picard@kti.ae.poznan.pl>

Agenda

- Lecture Goal(s)
- From Italy to Indonesia
- Interfaces, Classes, and Objects
- Attributes and Methods
- Encapsulation
- Inheritance and Polymorphism
- Conclusions

Lecture Goal(s)

Lectures Overview

Fundamental Concepts

- 1: Introduction
- 2: Basic data structures & Statements
- 3: Object-oriented programming I
- 4: Object-oriented programming II
- 5: Object-oriented programming III
- 6: Complex data structures
- 7: Threads & Exception handling

Today's Goal

To provide programming knowledge about object-oriented (OO) programming

From Italy to Indonesia

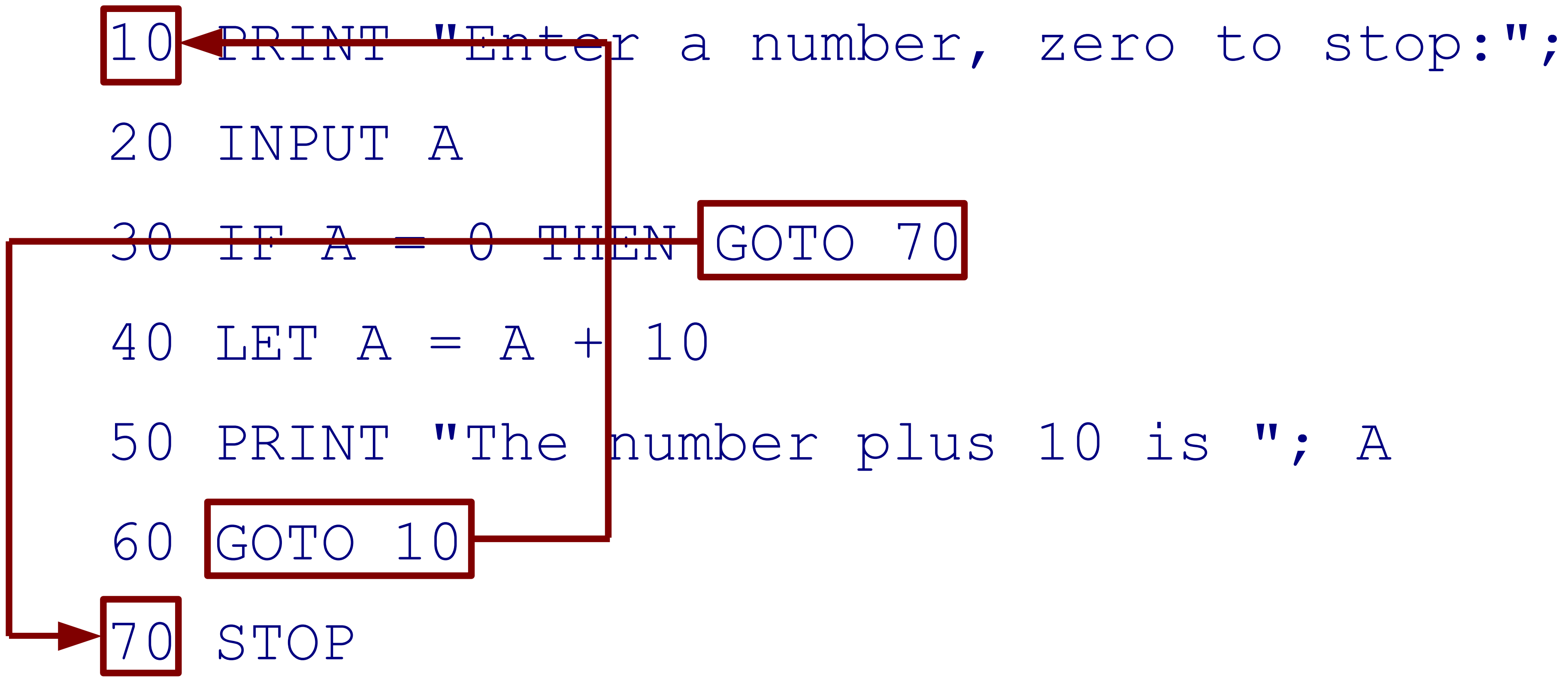
Italy

Spaghetti Programming

- Assembler, BASIC
- Intensive use of branching statements
- For `goto` lovers
- Unstructured data

Maintainability: *

BASIC Example



```
10 PRINT "Enter a number, zero to stop:";
20 INPUT A
30 IF A = 0 THEN GOTO 70
40 LET A = A + 10
50 PRINT "The number plus 10 is "; A
60 GOTO 10
70 STOP
```

The PASCAL Way

- Modularization
  - Procedures
  - Functions
- Structured data
- Functions and procedures coupled with data structures

Maintainability: **

PASCAL Record Example

```
program RECORD_INTRO (output);
      type  date = record
                     month, day, year : integer
                   end;
      var   today : date;
      begin
            today.day     :=   25;
            today.month   :=   09;
            today.year    := 1983;
            writeln('Todays date is ',
                     today.day,':',
                     today.month,':',
                     today.year)
      end.
```

PASCAL Procedure Example

```
program ADD_NUMBERS (input, output);
  procedure CALC_ANSWER ( first, second : integer );
    var   result : integer;
    begin
      result := first + second;
      writeln('Answer is ', result )
    end;

    var   number1, number2 : integer;
    begin
      writeln('Please enter two numbers to add');
      readln( number1, number2 );
      CALC_ANSWER( number1, number2)
    end.
```

WHY OO?

PASCAL Function Example

```
program ADD_NUMBERS (input, output);
  function SUM ( first, second : integer ): integer;
    begin
      SUM := first + second
    end;

    var   sum, number1, number2 : integer;
    begin
      writeln('Please enter two numbers to add');
      readln( number1, number2 );
      sum := SUM( number1, number2)
      writeln('Answer is ', sum)
    end.
```

The C Way

- Separation of
  - declarations
  - definitions
- Headers
- Libraries

Maintainability: ***

C Example

In myMath.h

```
int add(int i, int j);
```

In myMath.c

```
#include “myMath.h”
int add(int i, inj) { return i+j };
```

In myProg.c

```
#include "myMath.h"
#include <iostream.h>
#include <cstdlib>

int main( int argc, char* argv[]){
  int a = atoi(argv[1]); int b = atoi(argv[2]);
  int sum = add (a, b);
  cout << a << "+" << b << "=" << sum;
}
```

Limitations of the C Language

- Coupling between
  - procedures/functions
  - data structures
- Code reuse
- Spread code

Object-Oriented Programming Languages

- History
  - Nygaard and Dahl, Norwegian Computer Center
  - Simula 67
- Current OOPLs
  - C++
  - Objective C
  - Smalltalk
  - Eiffel
  - Common LISP Object System (CLOS)
  - Object Pascal
  - Ada

Indonesia

SOUTH CHINA SEA
BRUNAI
MALAYSIA
SUMATRA
KALIMANTAN
SULAWESI
MALUKU
PAPUA NEW GUINEA
IRIAN JAYA
JAVA
BALI
TIMOR
ARAFURA SEA
INDIAN OCEAN
TIMOR SEA

Interfaces, Classes, and Objects

Interface Definition

An interface defines a set of related functionalities (a behavior)

Interface Example

- Car interface
  - Enter the car
  - Start the car
  - Speed up
  - Break
  - Turn
  - Park the car
  - Stop the car
  - Exit the car

Interfaces in Java

- Syntax

```
interface <name>{
      ...
}
```

- Example

```
interface ICar {
      ...
}
```

Class Definition

An class defines the implementation of a set of related functionalities (a behavior)

Class Example

- Car class
  - e.g Citroen C3
  - Enter the car
  - Start the car
  - Speed up
  - Break
  - Turn
  - Park the car
  - Stop the car
  - Exit the car

Car Implementation Example

- Gas pedal
- Steering wheel
- Current speed

- Speed up
  - Press the gas pedal

Classes in Java

- Syntax

```
class <name>{
    ...
}
```

- Example

```
class CitroenC3 {
    ...
}
```

Class Classical Definition

An class is a bundle of variables and methods to operate on these variables

Object Definition

An object is an instance of a class

Object Example

- Car Object
  - A given car
  - e.g. The Citroen C3 with registration plates “PO TATO”
  - e.g. leather steering wheel
  - e.g. sport gas pedal
  - e.g. 30km/h

Classes and Interfaces

A class which implements an interface must define all methods declared in the interface

Classes and Interfaces in Java

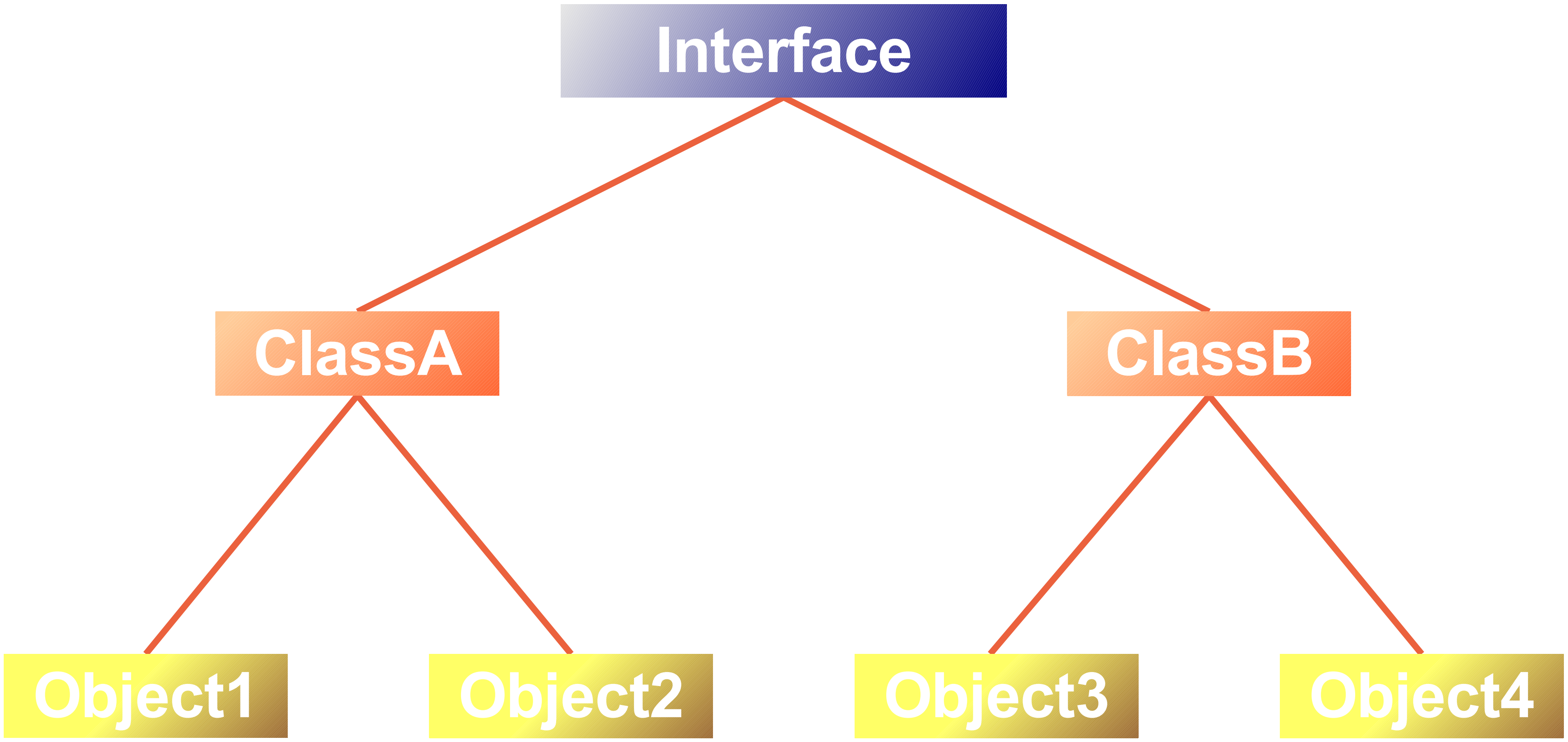
- Syntax

```
class <className> implements <interfaceName>{
    ...
}
```

- Example

```
class CitroenC3 implements ICar{
    ...
}
```

Summary



Conclusions

Conclusions

- Rule 1
  - Use interfaces
- Rule 2
  - Use interfaces
- Rule 3
  - Use interfaces

C Language vs. OOPLs

▸ Coupling between	*classes*
▸ procedures/functions	
▸ data structures	
▸ Code reuse	*inheritance*
▸ Spread code	*classes, inheritance*
▸ Description vs. definition	*interfaces, encapsulation*

Example

```
package pl.poznan.ae.compProg;
import java.util.*;

public class Sorter {
  private List _words;

  public void sort(String[] words){
    _words = Arrays.asList(words);
    Collections.sort(_words);
  }
  public String getSortedWords(){
    String sortedString = "";
    for (int i = 0; i< _words.size(); i++){
      sortedString += _words.get(i);
    }
    return sortedString;
  }
 public static void main(String[] args){
    Sorter sorter = new Sorter();
    sorter.sort(args);
    System.out.println(sorter.getSortedWords());
 }
}
```

See you next week

Object-Oriented Programming II

The Return