# Praktische Softwaretechnologie

Károly Bósa
(Karoly.Bosa@jku.at)

Research Institute for Symbolic Computation
(RISC)

# Literatures



- Xiaoping Jia, **Object-Oriented Software Development Using Java – Principles, Patterns, and Frameworks**, 2nd ed., Addison-Wesley, 2002. Practical Programming Homework. More than 50 Euro

- **Java Tutorial:**
  http://download.oracle.com/javase/tutorial/java/index.html

- Further books will be mentioned at the introduction of each Topic

# Introduction into the Object Oriented Programming

# Non-Structured Programming



- **Global data**
- **Only one main program**
- **Program flow branching by command *GOTO***
- **E.g.: typical beginner BASIC program**

```
…
50 IF A<>0 THEN GOTO 100
…
100 PRINT…
```

# (Block)Strucktured Programming



- **The program flow is controlled by program structures: *if-then-else, while*, etc.**
- **Global data**
- **Only one main program**
- **E.g.: a simple/beginner PASCAL program**

```
…
if a<>0 then begin
        …
end;
else begin … end;
…
```

# Procedural Programming



- The program code is wrapped into functional substructures (procedures, functions)
- The data are given among the program structures as arguments
- However the data are still global partially
- Accessing to the Global data is possible from every program structure
- The definition of the data structures are separated from the algorithmic program codes
- The contexts of the data structures and program structures are ambiguous; difficult to understand and reuse
- Typical (advanced) PASCAL program; a C program in one file

# Modular Programming



- **The algorithms and their dependent data are wrapped into modules**
- **The interfaces of the modules are well defined**
- **E.g.: Modula-2 and in C is also possible, in a C source file:**

```
static int i;
…
functions
```

# Objects



Grady Booch, Object-Oriented Design with Applications, Addison-Wesley,1991:

**An object has *state, behavior* and *identity*.**

# Objects



Grady Booch, Object-Oriented Design with Applications, Addison-Wesley,1991:

**An object has *state, behavior* and *identity*.**

- State = Data

# Objects



Grady Booch, Object-Oriented Design with Applications, Addison-Wesley,1991:

**An object has *state, behavior* and *identity*.**

- State = Data
- Behavior = Algorithms which use the data

# Objects



Grady Booch, Object-Oriented Design with Applications, Addison-Wesley,1991:

**An object has *state, behavior* and *identity*.**

- State = Data
- Behavior = Algorithms which use the data
- Identity = Distinguishably from other objects

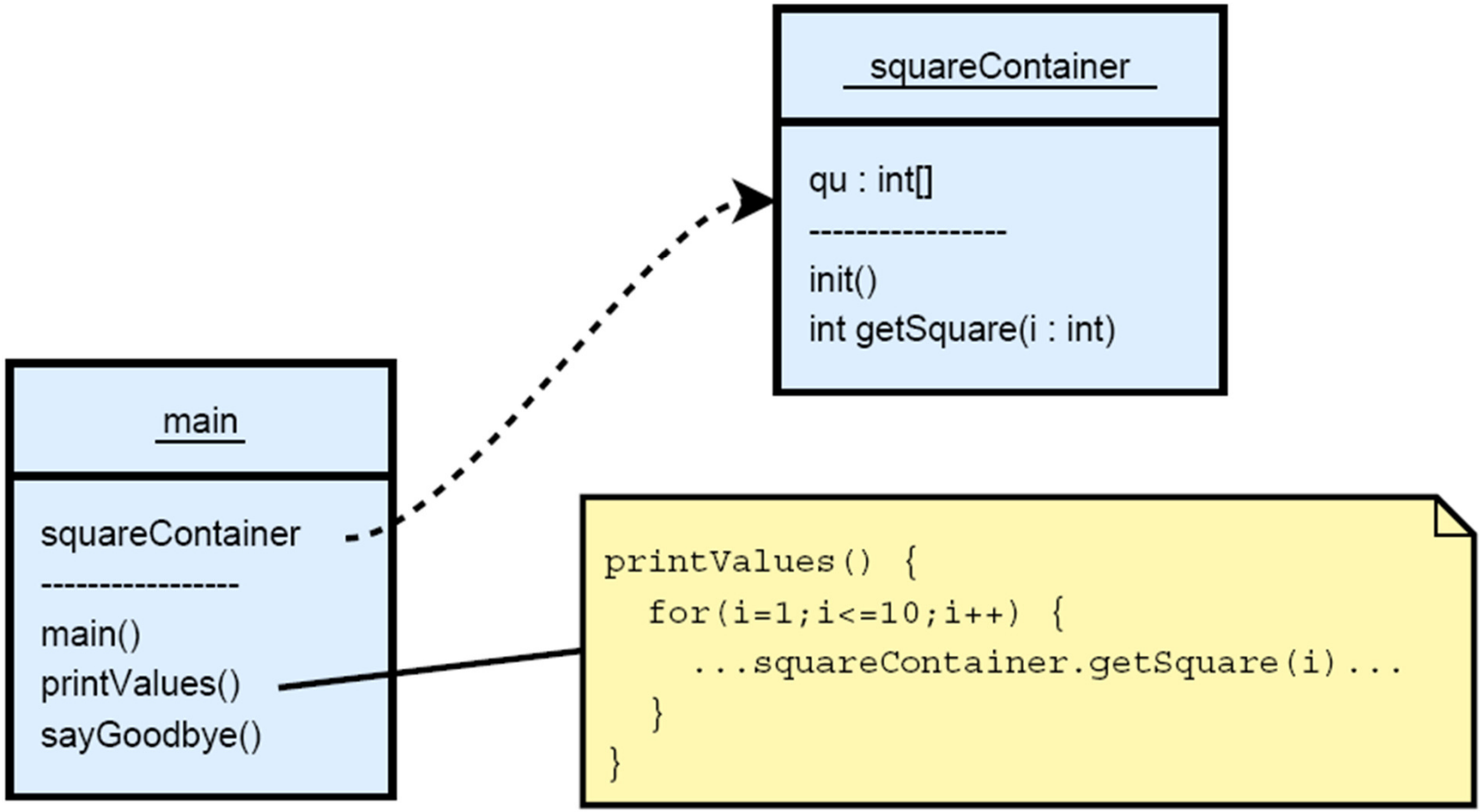
# Object Based Programming



- **The global state of a program consists of (the states of) numerous objects**
- **The objects interact with each other via messages**
- **These messages are realized as procedure/function calls, e.g.:**
  - sending message "*m*" to object *"o"* = calling procedure *"m"* of object *"o"*
  - Procedure *"m"* is able to modify directly the state of the objects *"o"* or to send another message to another object

# An Example for Objects





# Encapsulation



- Accessing to the field *"squareContainer.qu"* from outside (e.g.: from function "main") is not possible/desirable

- Accessing (changing/reading values) to the fields of an object is done typically though designated access points (*public* functions).

- **Advantages of this:**
  -avoiding side effects,
  -clear structures (storing the data and their algorithms together),
  -controlling the modification of the data, etc.

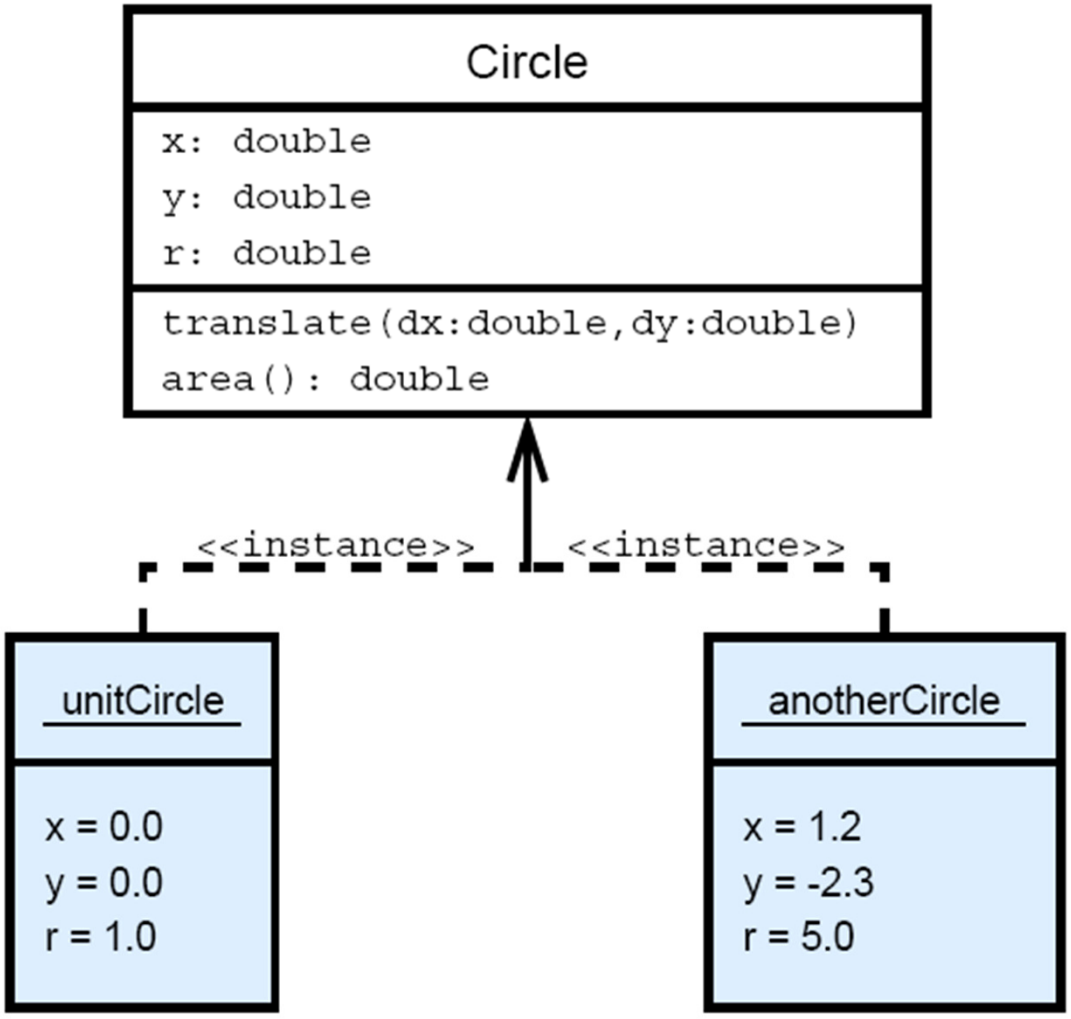
# Object Oriented Programming



- **Typing:** The objects belong to classes. Within a class each object has
  - the same data fields and
  - the same behavior (same functions).
- **Inheritance:** A class may inherit the data and behavior of (an)other class(es).
- **Polymorphism:** The same piece of program/function can work on different kind of objects.

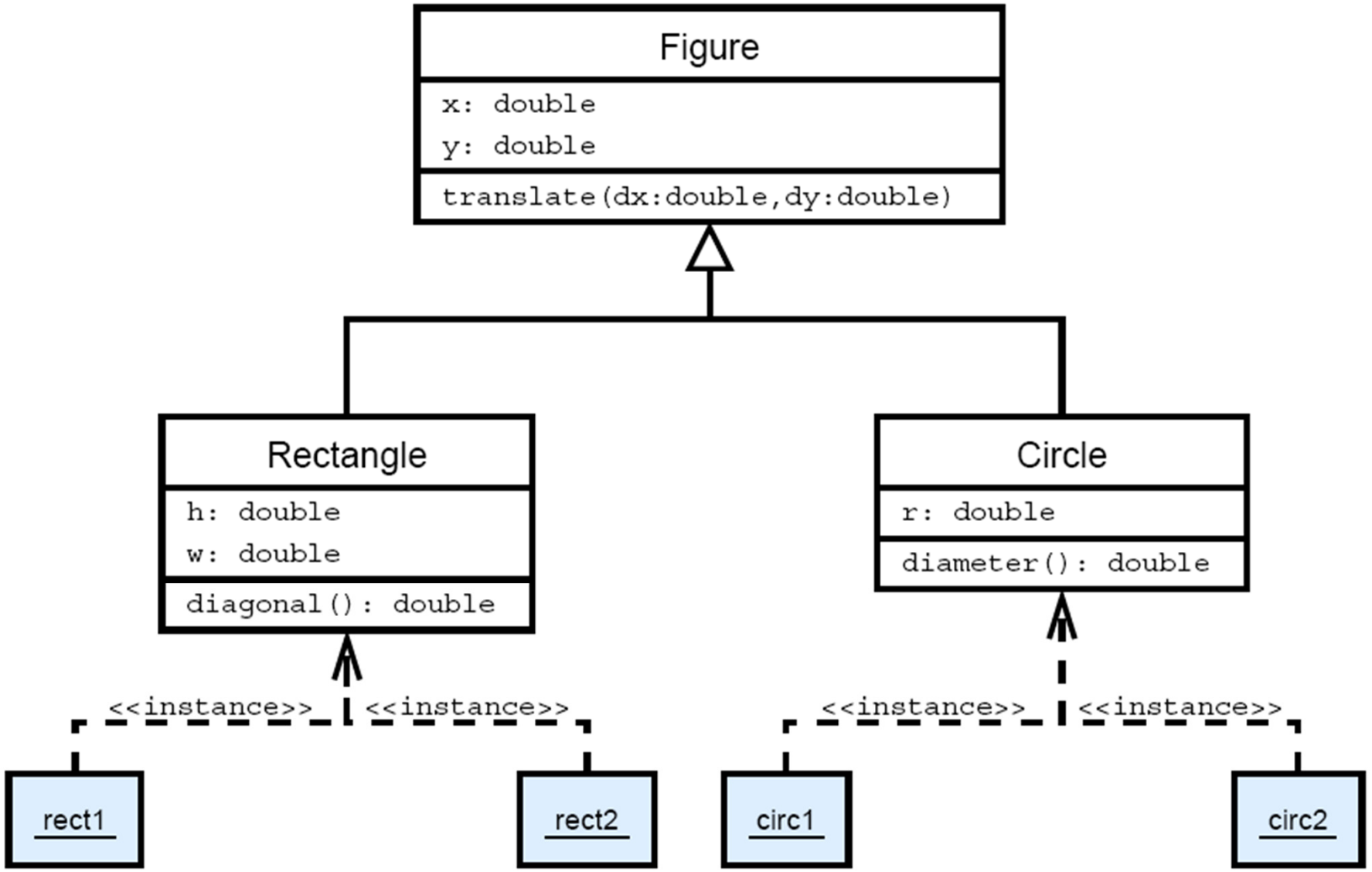
# An example for Classes





# An Example for Inheritance

Karoly.Bosa@jku.at



# An Example for Polymorphism



```
void m() {
  Figure f;
  Rectangle r = ...;
  Circle c = ...;

  f = r;      // Allowed , Rectangle is subclass of Figure
  f = c;      // Allowed , circle is subclass of Figure

  c = f;      // Not Allowed (!!!)
}
```

# Influence of Polymorphism for Behavior

Karoly.Bosa@jku.at

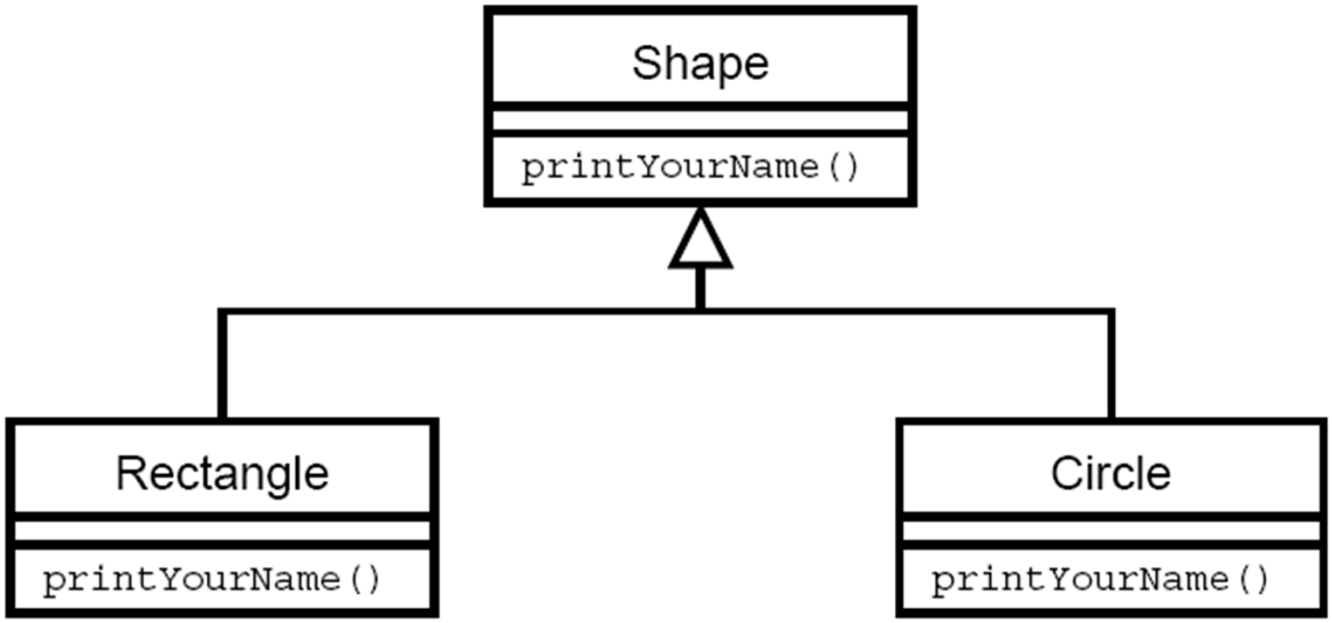
```
void diagTranslate(Figure f,double d) {
  f.translate(d,d);
}

Rectangle r = ...;
Circle c = ...;

diagTranslate(r,1.0);
diagTranslate(c,2.0);
```

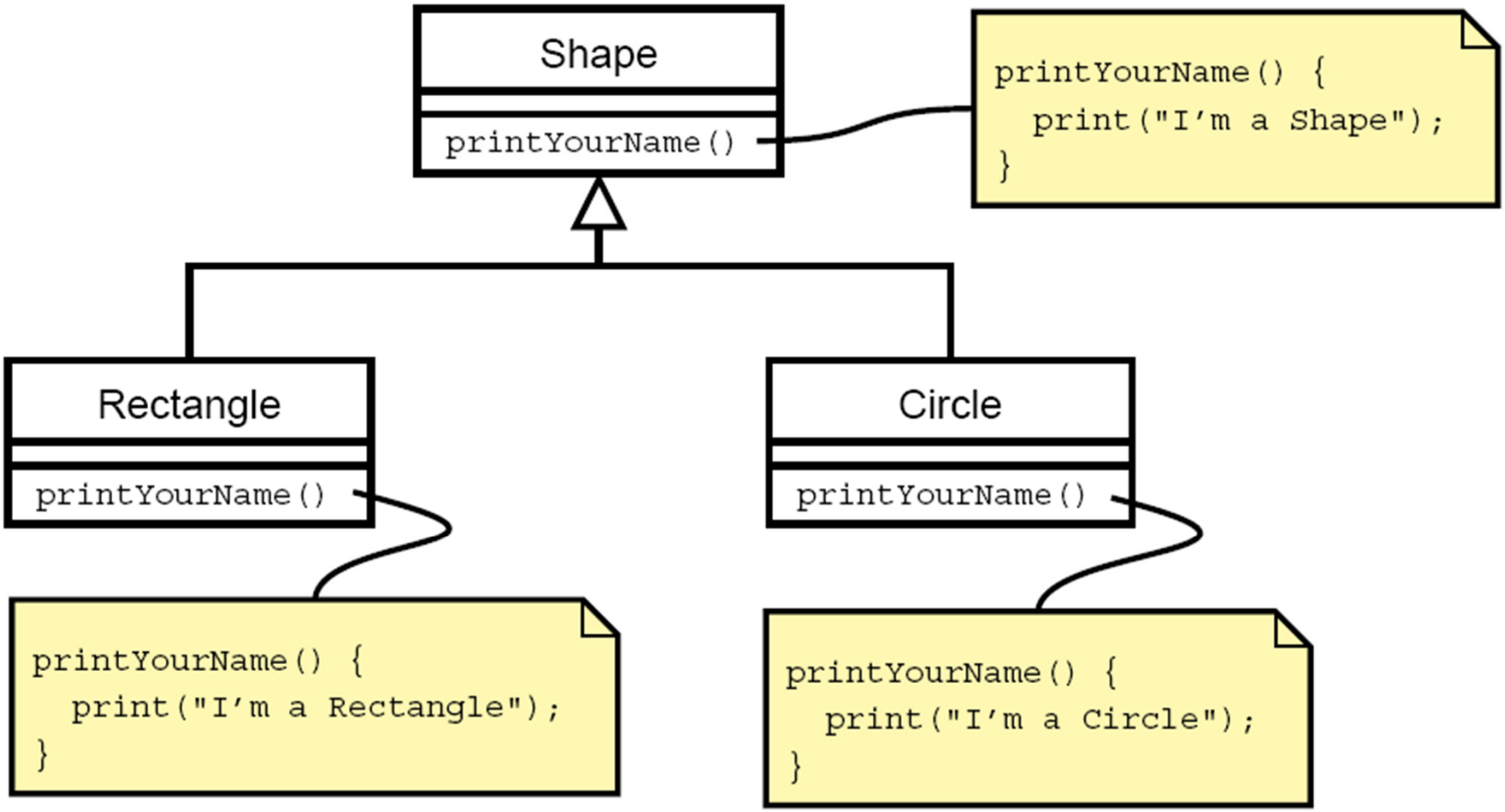
# Dynamic/Late Binding 1.





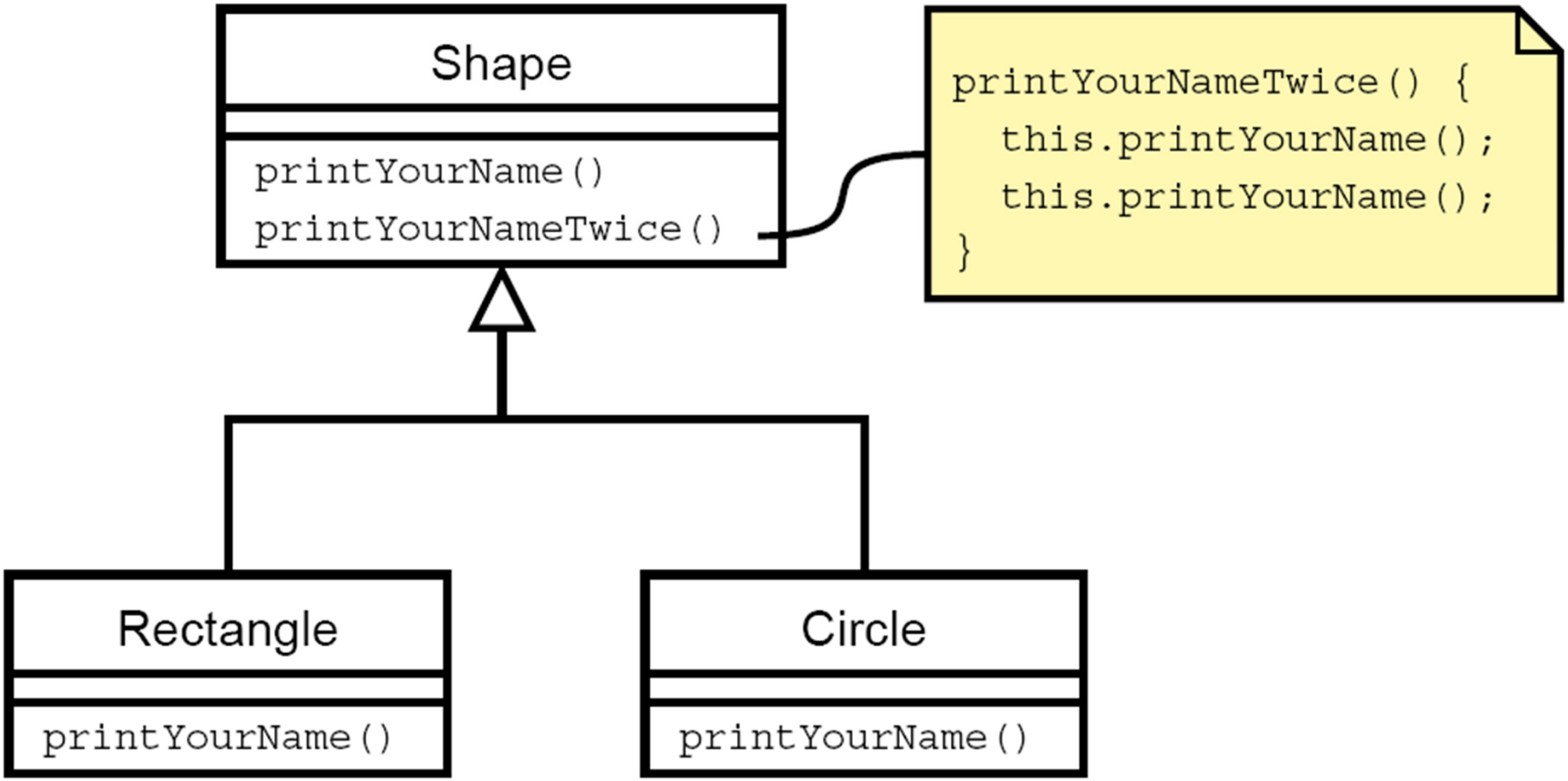
# Dynamic/Late Binding 1.





# Dynamic/Late Binding 2.





# Simula 67



**Simula 67** was the first OO language. It was developed in the years 60th in Oslo by Ole-Johan Dahl and Krysten Nygaard.

- It was mainly applied for simulations.
- It is an extension of Algol 60.
- Next to the standard types it uses classes and inheritance.
- The methods/behaviors have not been bound strictly to the objects yet.
- Practically it is used only in Europe.

# Smalltalk



The first “real/consequent” OO language. It was developed in the years of 70s at the Xerox PARC by Alan Kay and others.

- The influence of the Simula
- Everything is an object
- Already a development tool with GUI
- It is still used at present
- It had a strong influence for many other OO languages

# OO Expansions of other Prog. Languages

Karoly.Bosa@jku.at

- C
  - ➠ Objective C
  - ➠ C++
- PASCAL/Modula-2
  - ➠ Oberon
  - ➠ Modula-3
- FORTRAN
  - ➠ Fortran 2003
- Ada
  - ➠ Ada 95
- LISP
  - ➠ CLOS
- Standard ML
  - ➠ Objective CaML
- Haskell
  - ➠ O'Haskell

# Definiton of OO?



There is not an accurate definition which is accepted by everyone, some example on Ward's Wiki:

http://www.c2.com/cgi/wiki?WelcomeVisitors

http://www.c2.com/cgi/wiki?DefinitionsForOo

http://www.c2.com/cgi/wiki?NobodyAgreesOnWhatOoIs

http://www.c2.com/cgi/wiki?ObjectOrientedForDummies

# Definition of Kirsten Nygaard



Kristen Nygaard (1926-2002) was one of the developer of Simula 67, which was the first OO language. His definition is:

> A program execution is regarded as a physical model, simulating the behavior of either a real or imaginary part of the world.

# Defintion of Alan Kay



Alan Kay was one of the developer of Smalltalk, which is a very successful OO language. He requires the following essential elements for an OO language:

- Polymorphism
- Data encapsulation
- Inheritance
- Every type is an object type
- The object types compose a hierarchy with a single root

# What is OO?



- There is a lot of differences.
- But a lot of common issues among the OO language as well.
- In this lecture we will focus on the Java language

# Exercise I.

# Exercise I.



- Installing of Java or Finding an Java installation on an available Computer
- Writing a “Hello World” program with Text Editor
- Compiling the program from command line
- Executing the program from the command line

# Exercise 1



- Installing of Java or Finding an Java installation on an available Computer
- Writing a “Hello World” program with Text Editor
- Compiling the program from command line
- Executing the program from the command line
- Not necessary to understand (yet) ;-)
- See the guidance for this exercise on the web page of the lecture.