경희대학교
Kyung Hee University

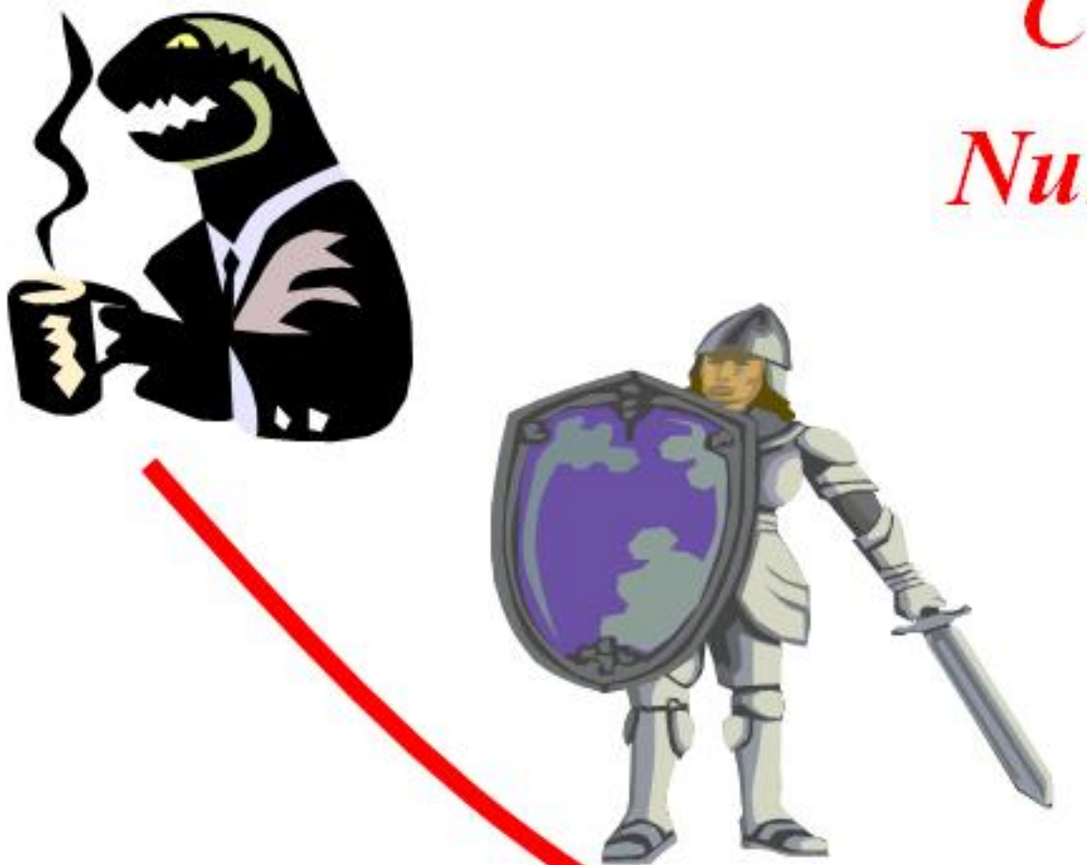
Advanced Object Oriented Programming

Introduction to Object-oriented Programming

Seokhee Jeon

Department of Computer Engineering
Kyung Hee University
jeon@khu.ac.kr

History of S/W Engineering



Complexity of problem,
Number of programmers

Initial Stage

- Simple problem
  - focus on actions
- One or small number of programmers
  - Enough time
  - Low salary (low cost)

Unstructured Programming

Transient Stage

- Large problem
  - still focus on actions
- Tens to hundreds of programmers
  - Tight deadline
  - Reasonable, but somehow high cost

Modular or Structural Programming

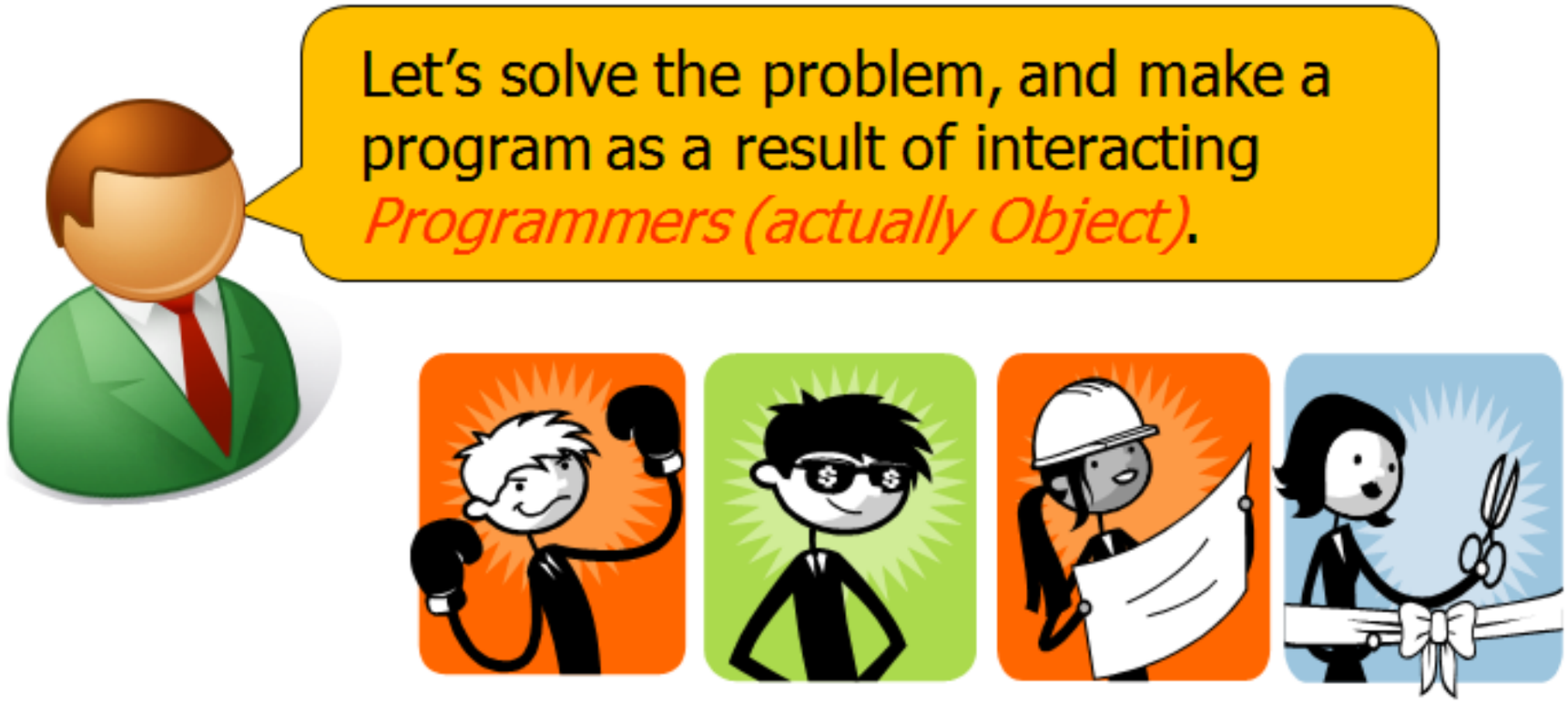
Current Stage

- Huge problem
  - data is more important than action
- Tens to hundreds of programmers
  - Always A.S.A.P (As Soon As Possible)
  - Too high cost → S/W reusability is critical (S/W as a component)

Object Oriented Programming

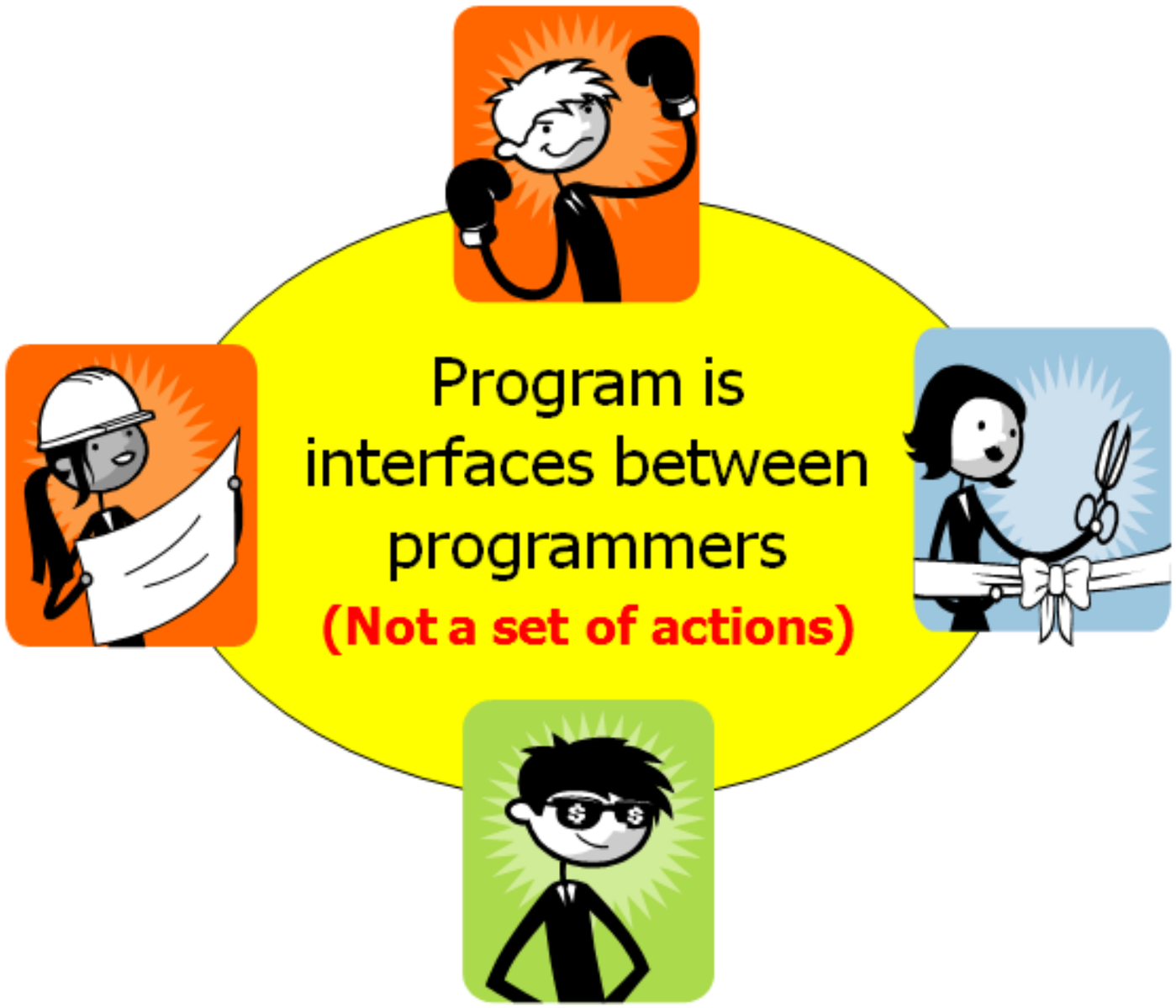
S/W Revolution

- Collaborated Work (Programmer's point of View)
  - **I don't know "WHO YOU ARE (data)".**
  - **I don't know "HOW YOU DO (your internal behavior)"**
  - **BUT, Please do right thing.**
  - **AND, Let's use mutually agreed interface.**



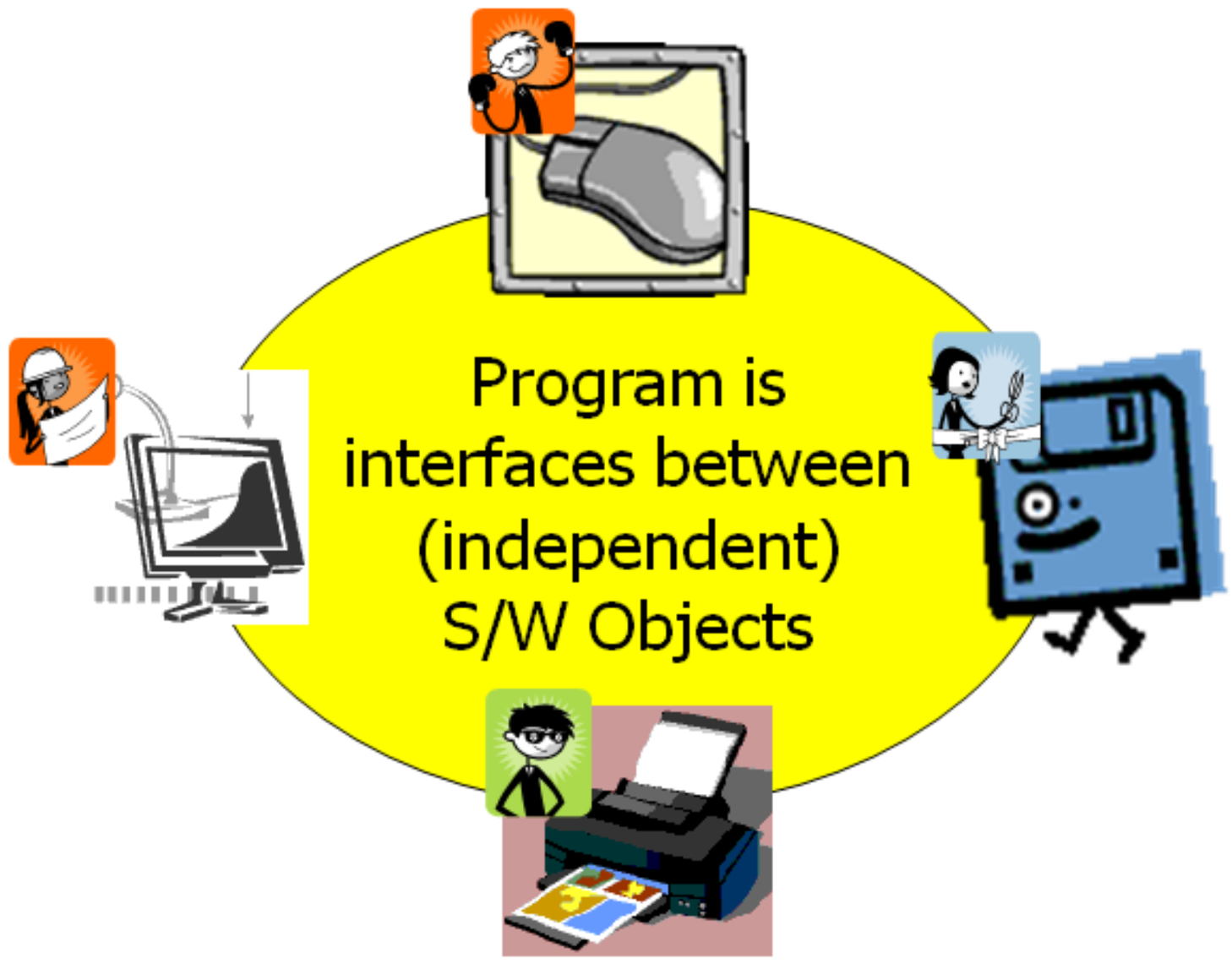
OOP

- Collaborated Work (Programmer's point of View)



OOP

- Collaborated Work (Program's point of View)



S/W Object

- What is S/W object?
  - S/W object example for ***CAT***

STATE

Name

Breed

Weight

Age

Asleep

BEHAVIOUR

Sleeps a lot

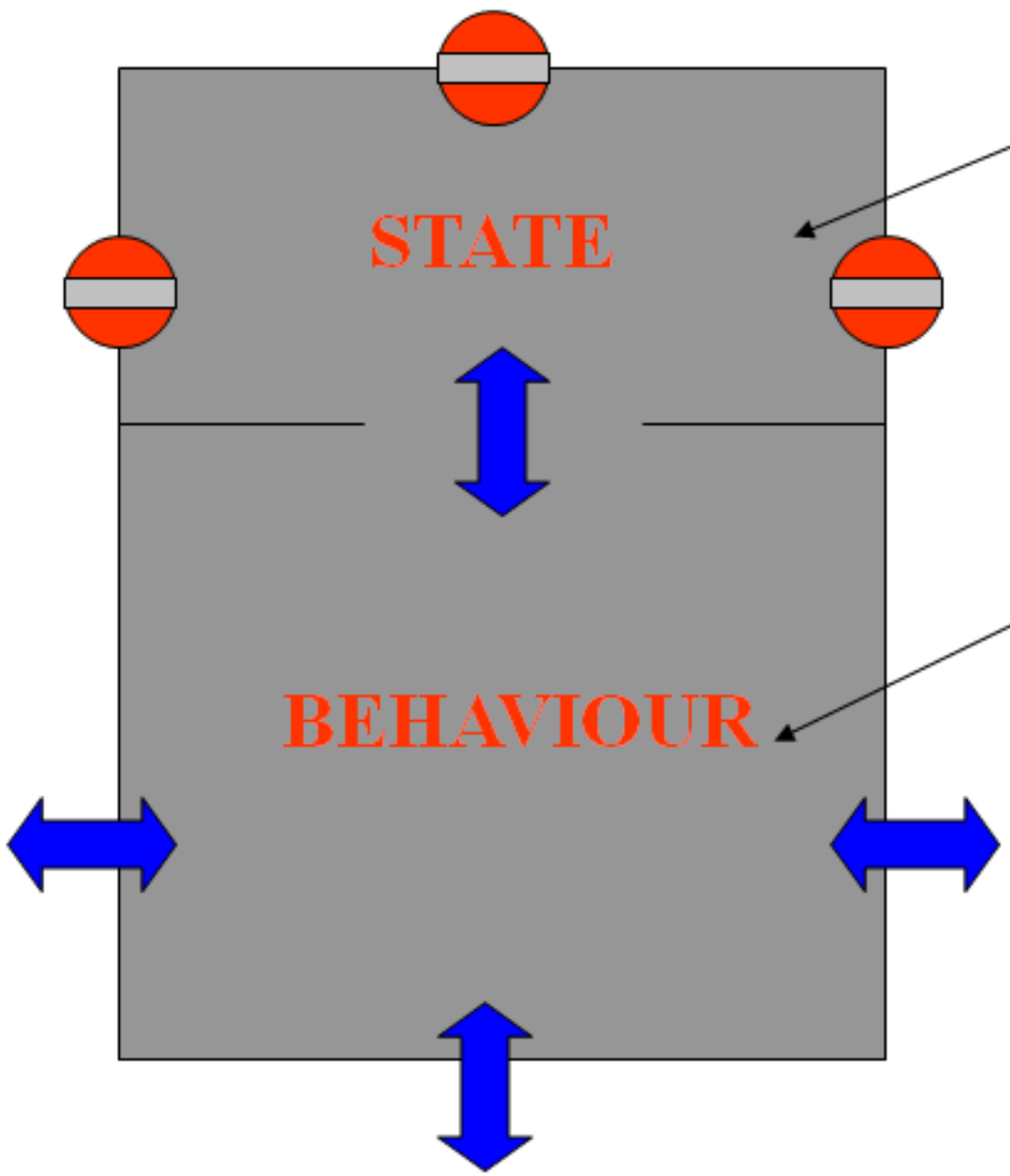
Scratches furniture

Catches mice

Fights other cats

S/W Object

- Implementation of S/W objects

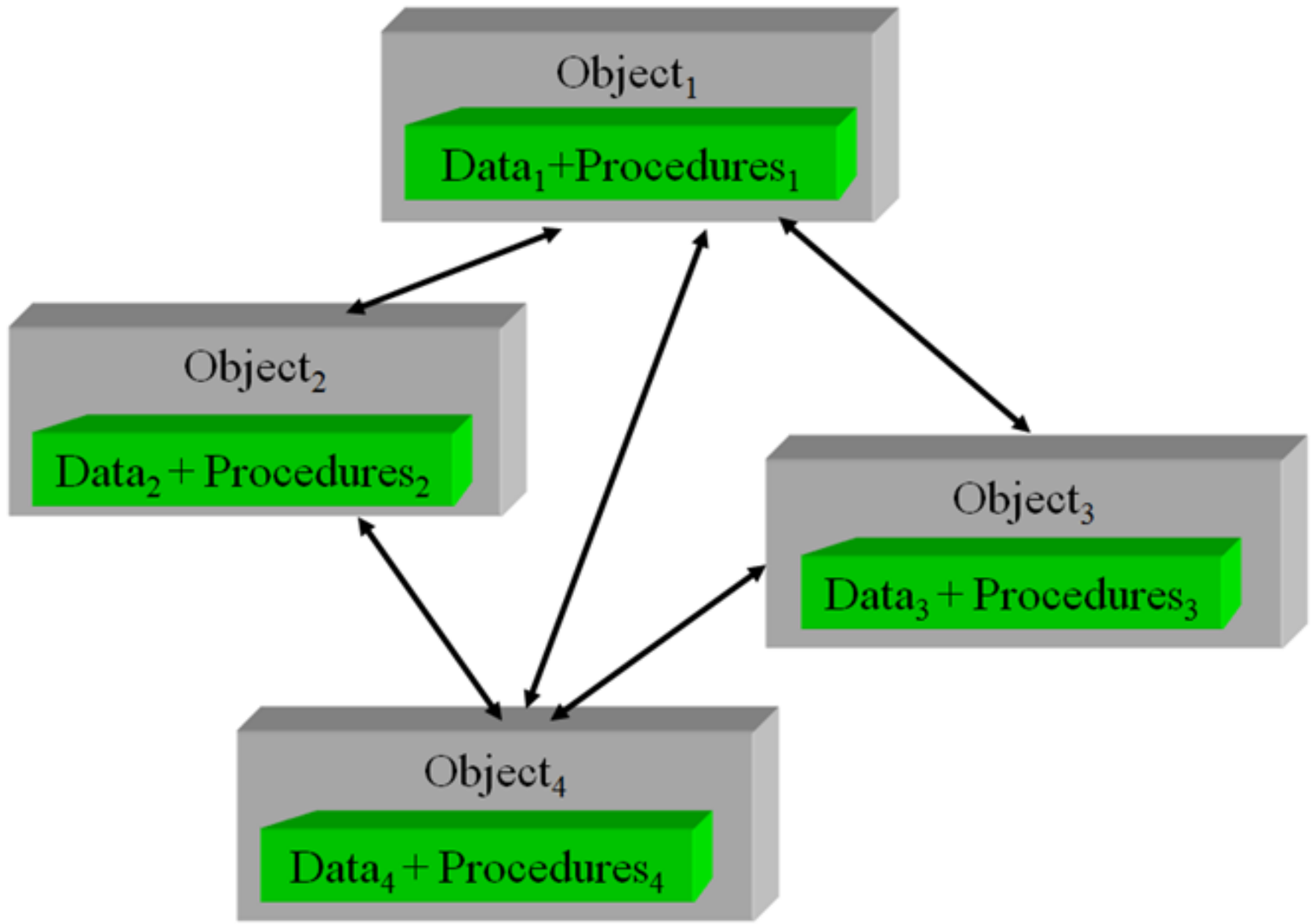


State is usually held as local variables (also called properties), quantities not visible outside the object (data hiding)

Behaviour controlled by *method functions* or subroutines which act on the local variables and interface with the outside.

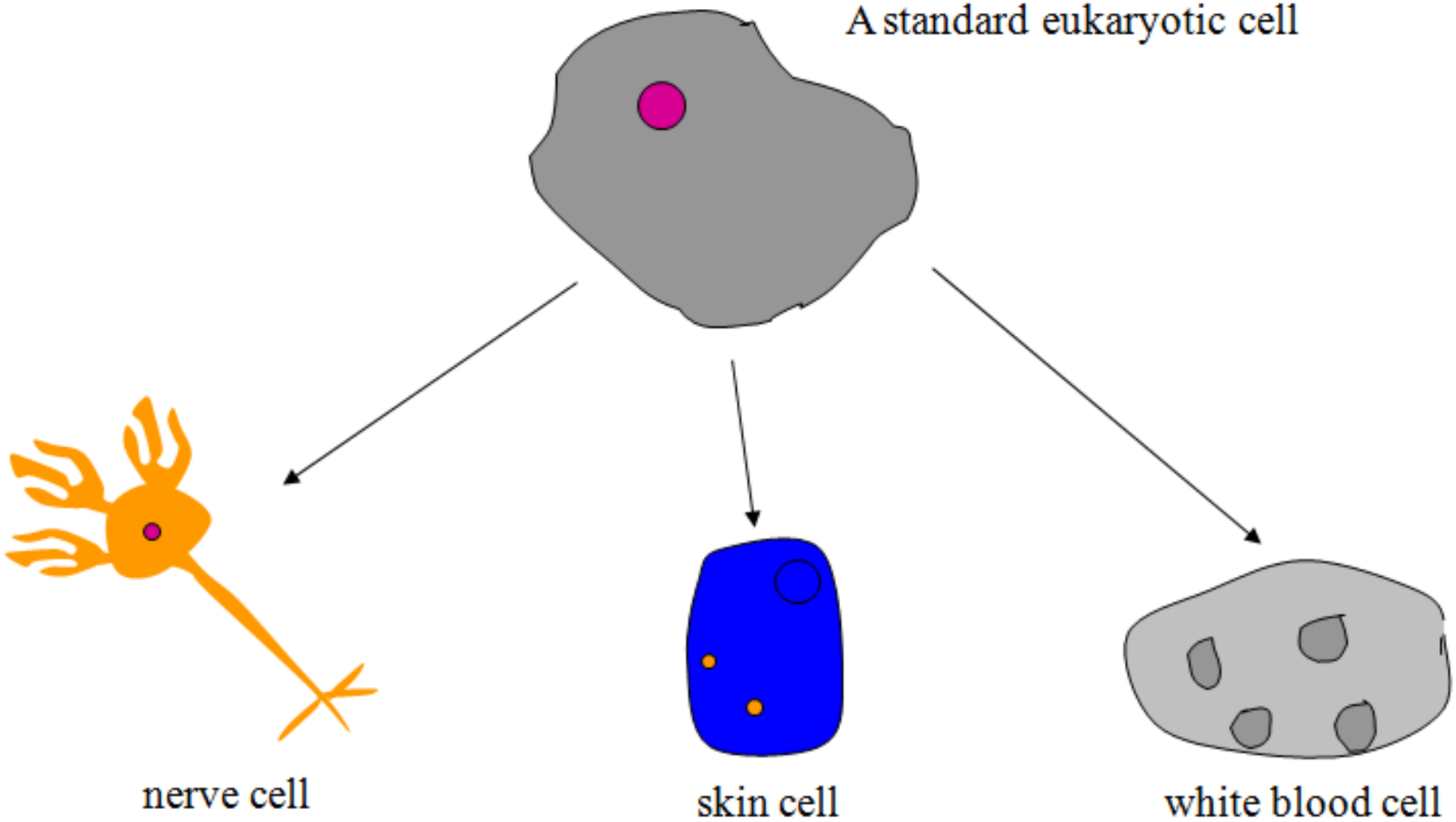
S/W Object

- Implementation of S/W objects



S/W Object

- Key feature of S/W Object – ***Inheritance***
  - Ability to derive one object from a more general class of related objects



Summary

- Objects provide a powerful and natural approach to representing many problems
- Features such as **inheritance** allow already written objects to be re-used – program modification easier.

- Certainly more difficult than conventional programming
  - Some concepts hard, even for experienced programmers
  - Implementation of objects often use complicated syntax/semantics
- OOP not famous for efficiency (memory or execution time)
  - C++ once famous for being slow, now much better
  - Java still famous for being slow

Let's start about Class

- Access identifiers - Private & Public : **WHY?**

Let's start about Class

- People have interests for "How to use TV"
  - Turn on/off
  - Volume up/down
  - Change channels
  - …

Let's start about Class

- Just few people have interests for its internal devices and actions

Let's start about Class

- So what?

"Hides most data and actions of TV"

"Just shows simplified actions to the (TV's) user"

Through these simplified actions,

hided data and actions are managed.

Let's start about Class

- So what?

"Hides most data and actions of TV"

"Data Hiding"

"Just shows simplified actions to the (TV's) user"

Through these simplified actions,

hided data and actions are managed.

"Encapsulation"

Questions?