Copyright Warning

COMMONWEALTH OF AUSTRALIA

Copyright Regulations 1969

WARNING

This material has been reproduced and communicated to you by or on behalf of the University of Sydney pursuant to Part VB of the Copyright Act 1968 (**the Act**).

The material in this communication may be subject to copyright under the Act. Any further copying or communication of this material by you may be the subject of copyright protection under the Act.

Do not remove this notice.

Copyright © ViSLAB, The University of Sydney

Object Oriented Design (Reuse)

Week 6-1

Copyright © ViSLAB, The University of Sydney

OOP led Reuse

Copyright © ViSLAB, The University of Sydney

OOP led Reuse

- Many OOP Lang (Java, .NET)

Copyright © ViSLAB, The University of Sydney

OOP led Reuse

- Many OOP Lang (Java, .NET)
  - reuse byte-code (not source code)

Copyright © ViSLAB, The University of Sydney

OOP led Reuse

- Many OOP Lang (Java, .NET)
  - reuse byte-code (not source code)
  - reusable software modules

Copyright © ViSLAB, The University of Sydney

OOP led Reuse

- Many OOP Lang (Java, .NET)
  - reuse byte-code (not source code)
  - reusable software modules
    - Class Libraries

Copyright © ViSLAB, The University of Sydney

OOP led Reuse

- Many OOP Lang (Java, .NET)
  - reuse byte-code (not source code)
  - reusable software modules
    - Class Libraries
    - Frameworks

Copyright © ViSLAB, The University of Sydney

OOP led Reuse

- Many OOP Lang (Java, .NET)
  - reuse byte-code (not source code)
  - reusable software modules
    - Class Libraries
    - Frameworks
    - Components

Copyright © ViSLAB, The University of Sydney

OOP led Reuse

Copyright © ViSLAB, The University of Sydney

OOP led Reuse

- Another form of reuse

Copyright © ViSLAB, The University of Sydney

OOP led Reuse

- Another form of reuse
  - Design Pattern

Copyright © ViSLAB, The University of Sydney

OOP led Reuse

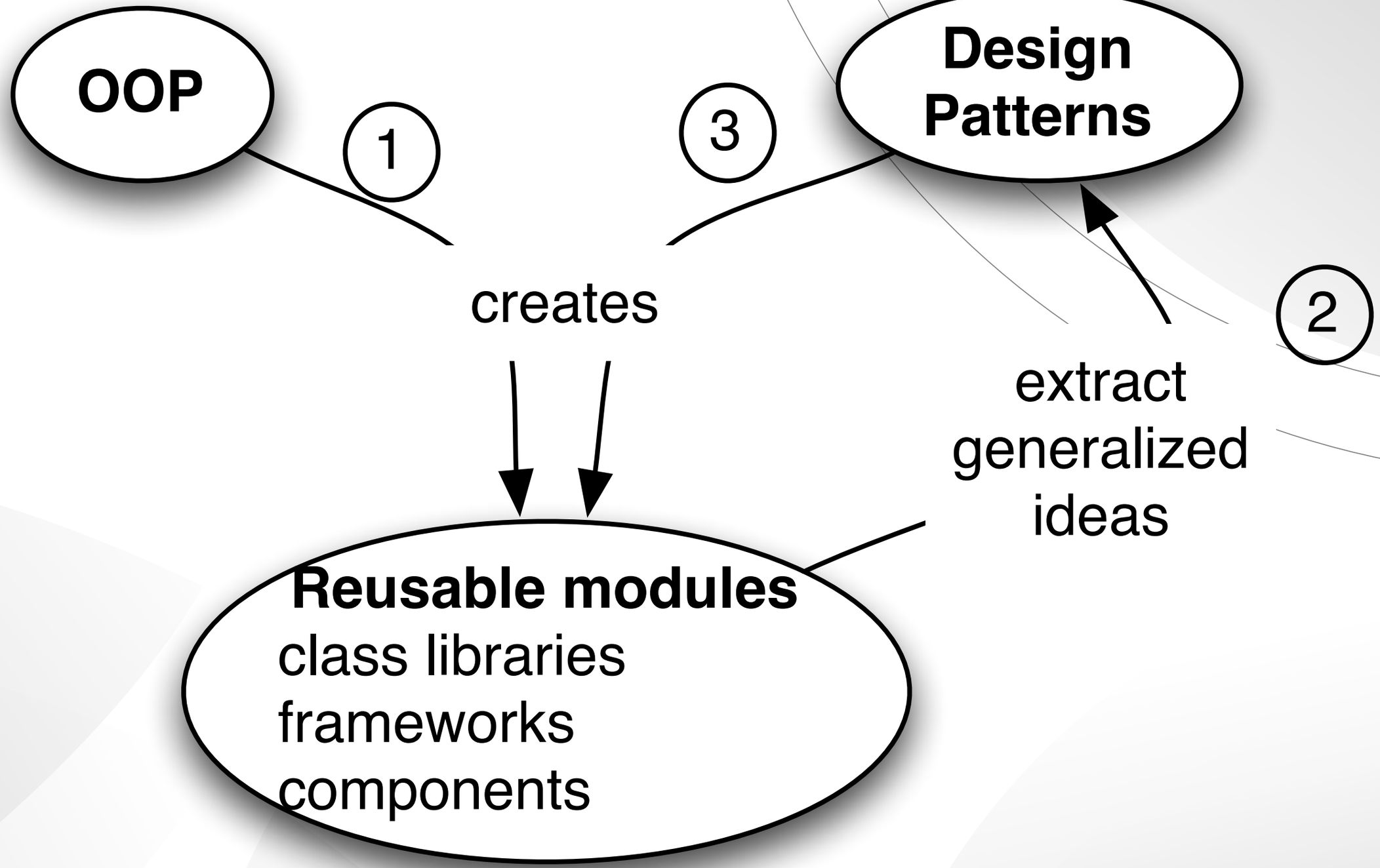
- Another form of reuse
  - Design Pattern
  - not concrete implementation...you cannot use it as modules

Copyright © ViSLAB, The University of Sydney

OOP led Reuse

- Another form of reuse
  - Design Pattern
  - not concrete implementation...you cannot use it as modules
  - documented form of general rules in prog.

Copyright © ViSLAB, The University of Sydney

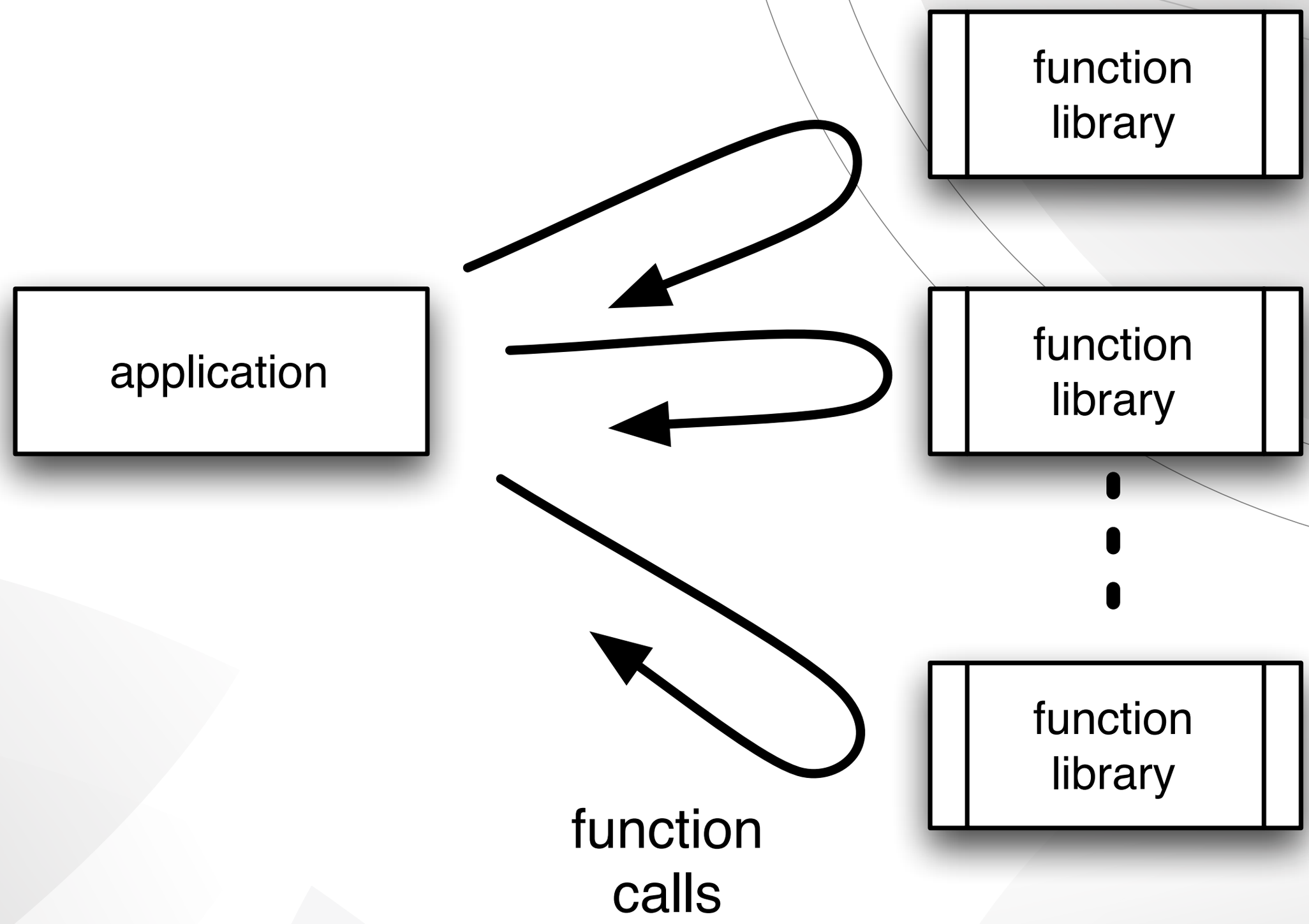


Copyright © ViSLAB, The University of Sydney

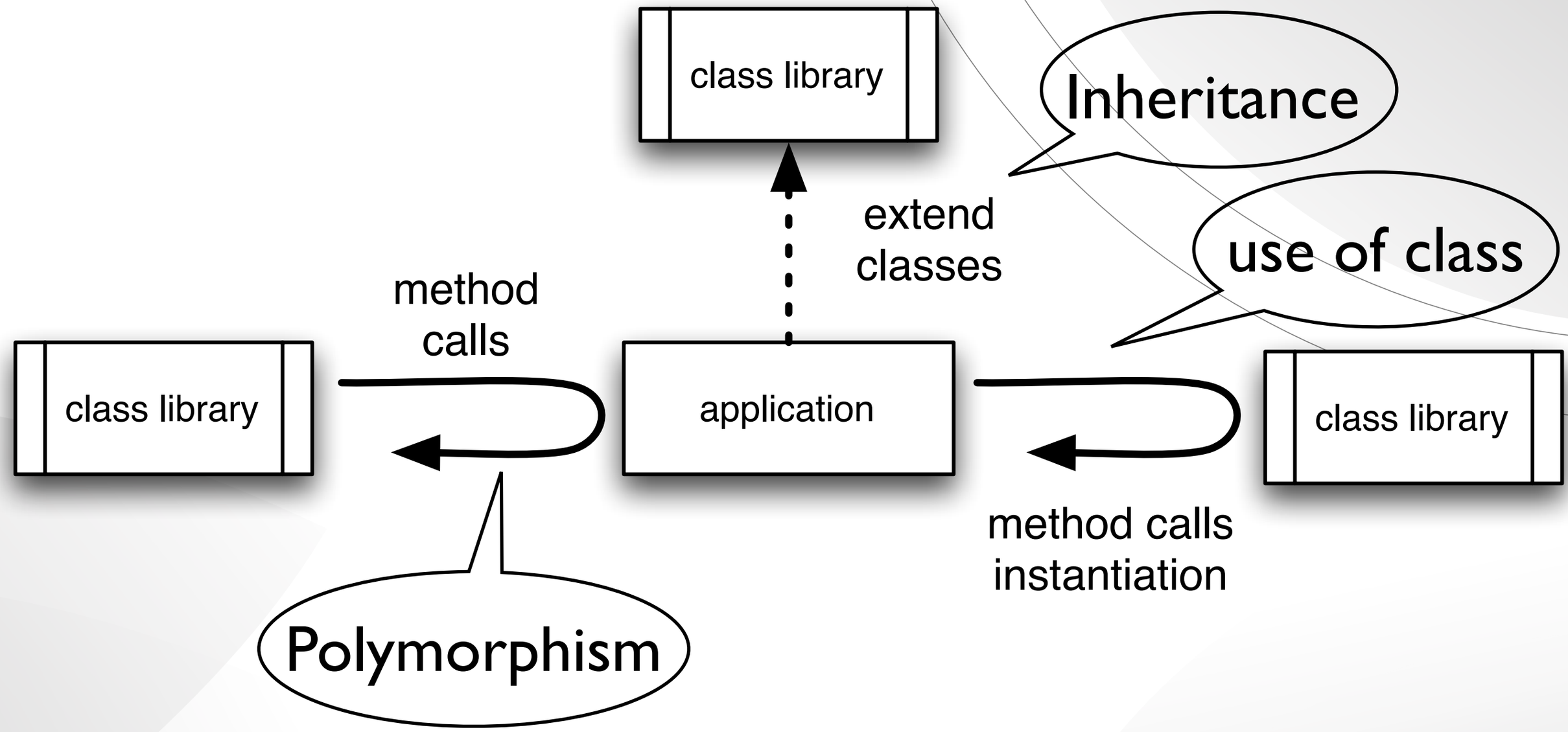
Class Libraries

- before class libraries
  - function libraries
  - subroutine libraries
- with OOP....class is the basic unit of S.W. modules
  - hence....class libraries
  - but assume the OO-based use.

Copyright © ViSLAB, The University of Sydney



Copyright © ViSLAB, The University of Sydney



Copyright © ViSLAB, The University of Sydney

extending lang.

- function libraries
- class libraries
  - extend prog lang's capabilities
    - not mods to lang spec.

- Java, .NET comes with tons of class libs.
- careful small lang spec updates.

Copyright © ViSLAB, The University of Sydney

Framework

- My first encounter...(around 1988).
  - NeXTStep / OPENSTEP frameworks

- Now..everywhere..

Copyright © ViSLAB, The University of Sydney

Framework

- similar to class libraries...but
- different in its purpose how they're used.

- class libraries
  - build using OO mechanism
  - do not specify how they should be used and their purposes.

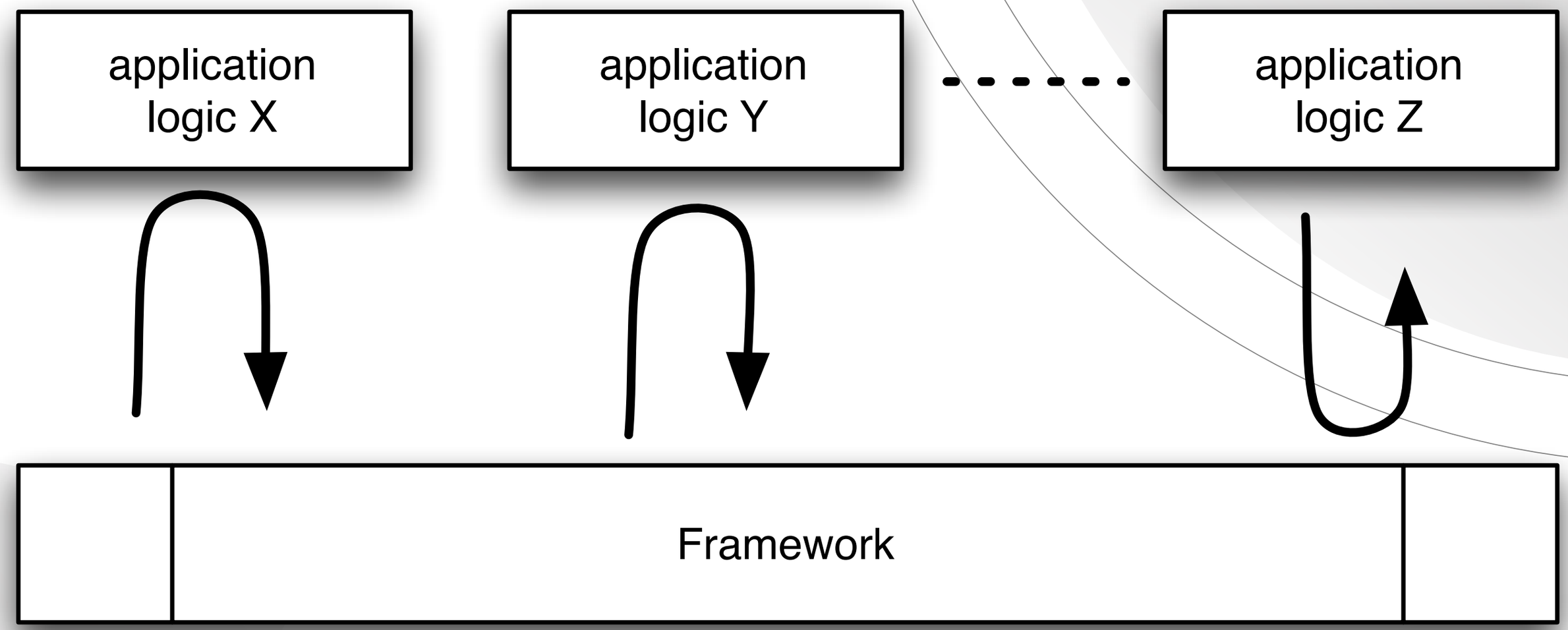
Copyright © ViSLAB, The University of Sydney

Framework

- half baked part of an application
- its behaviours have predefined purposes
- how they should be used....defined

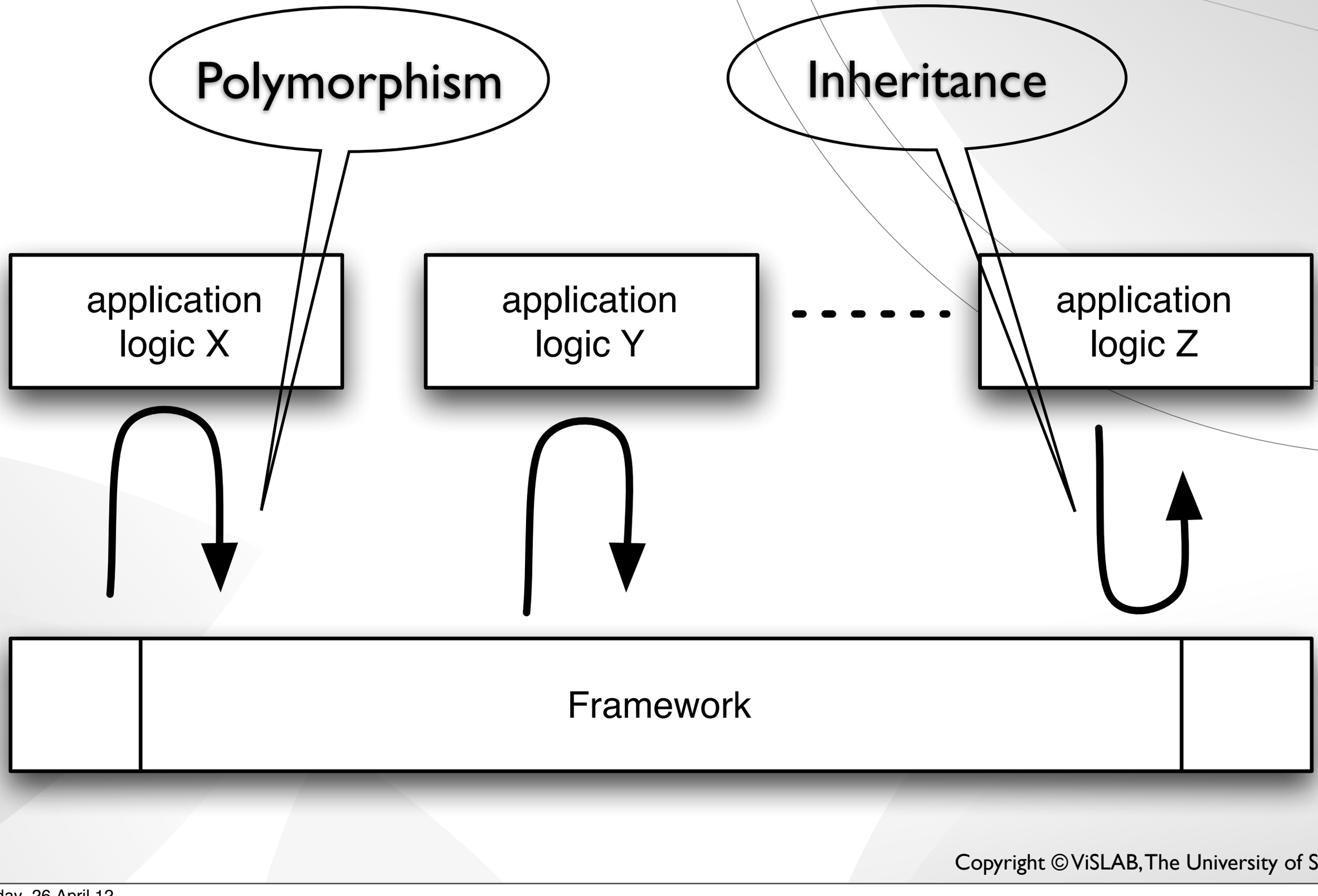
- your application will provide concrete/specific logic to fit into the framework.

Copyright © ViSLAB, The University of Sydney

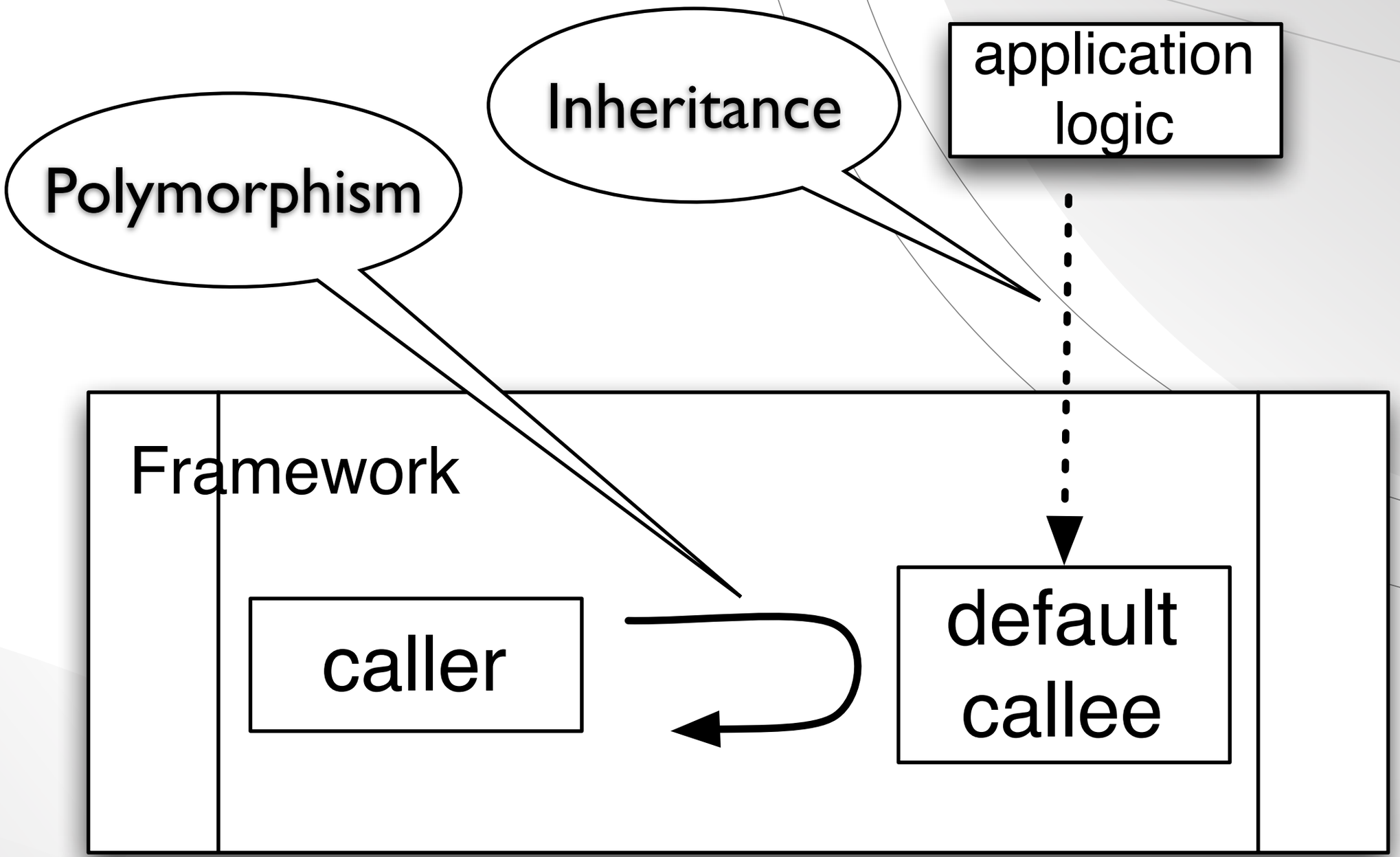


Copyright © ViSLAB, The University of Sydney

application specific logic



Copyright © ViSLAB, The University of Sydney



Copyright © ViSLAB, The University of Sydney

Framework example

- Java Applet
  - you create a class ..inherits Applet class
  - implement
    - init
    - start, etc.
  - your applet will run in a web page.

Copyright © ViSLAB, The University of Sydney

Things to watch out

- duplication of names
  - class names
  - method names
- C++ : namespace
- Java : package

Copyright © ViSLAB, The University of Sydney

Components

- more close to "off the shelf" modules
- typically distributed in binary forms (including bytecode)
- larger granularity
- self-contained
- no need for internal details, etc.

Copyright © ViSLAB, The University of Sydney

Component examples

- JavaBeans, Enterprise JavaBeans
  - deployable java components
  - independently developed
- Visual Basic's control components (GUI)
  - supports drag&drop

Copyright © ViSLAB, The University of Sydney

Components

- usually see where you can distribute executable modules.
  - Windows : proprietary environment
  - Java, .NET: cross-platform, bytecode
  - C++: ?

Copyright © ViSLAB, The University of Sydney

Design Patterns

- building
  - class libraries, frameworks, component libraries.....
- realise that there are many COMMON WAYS to build reusables
- regardless of platforms/dev. environment.

Copyright © ViSLAB, The University of Sydney

Design Patterns

- are not
  - a set of sourcecode
  - a set of pre-compiled executables
- are
  - WRITTEN document describing how you could design software modules/systems

Copyright © ViSLAB, The University of Sydney

Design Patterns

- originates in
  - Architectural Design Concept
  - Christopher Alexander
  - *A Pattern Language: Towns, Buildings, Construction* (1977)
  - *The Timeless Way of Building* (1979)

Copyright © ViSLAB, The University of Sydney

Design Patterns

- does not tell you how to implement in a particular OOP lang.
  - may have reference implementation.

- "Design Patterns" (1995)
  - Erich Gamma, Richard Helm, Ralph Johnson and John Vlissides
  - GoF (Gang of Four)

Copyright © ViSLAB, The University of Sydney

Creational		
	Abstract Factory	generation of related components
	Builder	build a complex instance
	Factory Method	delegate instance creation to a subclass
	Prototype	create an instance by copy
	Singleton	only one instance

Copyright © ViSLAB, The University of Sydney

Structural		
	Adapter	use over a wrapper
	Bridge	separate functional hierarchy and implementation hierarchy
	Composite	recursive container and components
	Decorator	recursive decorator and decoratee
	Facade	provide a simple entry point to interact with many
	Flyweight	minimise resource wasting through sharing
	Proxy	act on behalf of ...

Copyright © ViSLAB, The University of Sydney

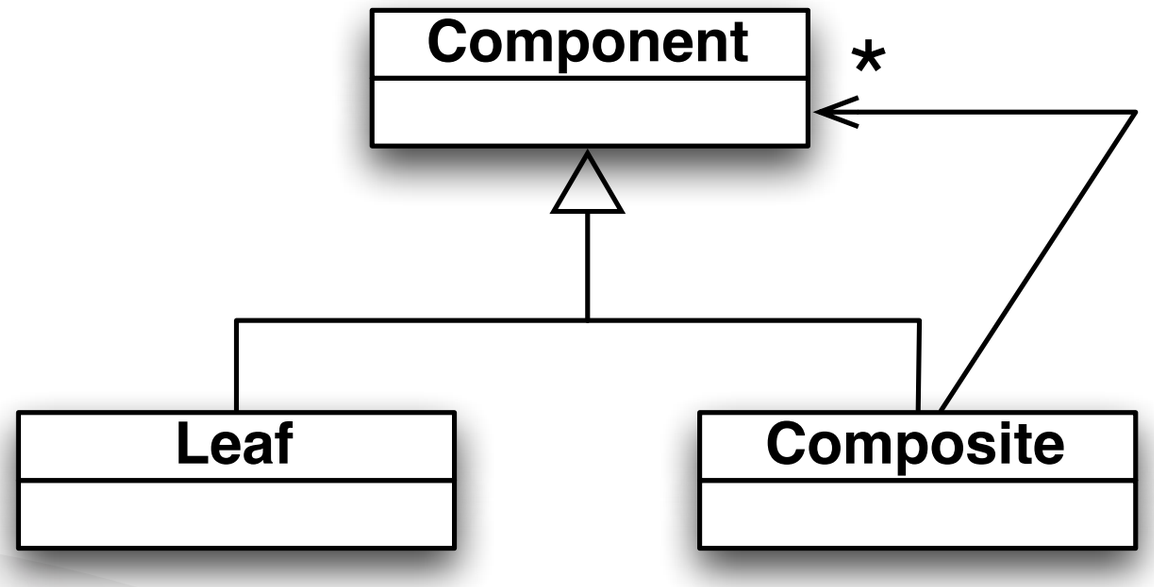
Behavioral		
	Chain of Responsibility	keep delegating
	Command	oder something to objects
	Interpreter	represents grammatical rules
	Iterator	count/visit one by one
	Mediator	provide a unified consultation point
	Memento	keep object's status for later use.
	Observer	notify the change of state

Copyright © ViSLAB, The University of Sydney

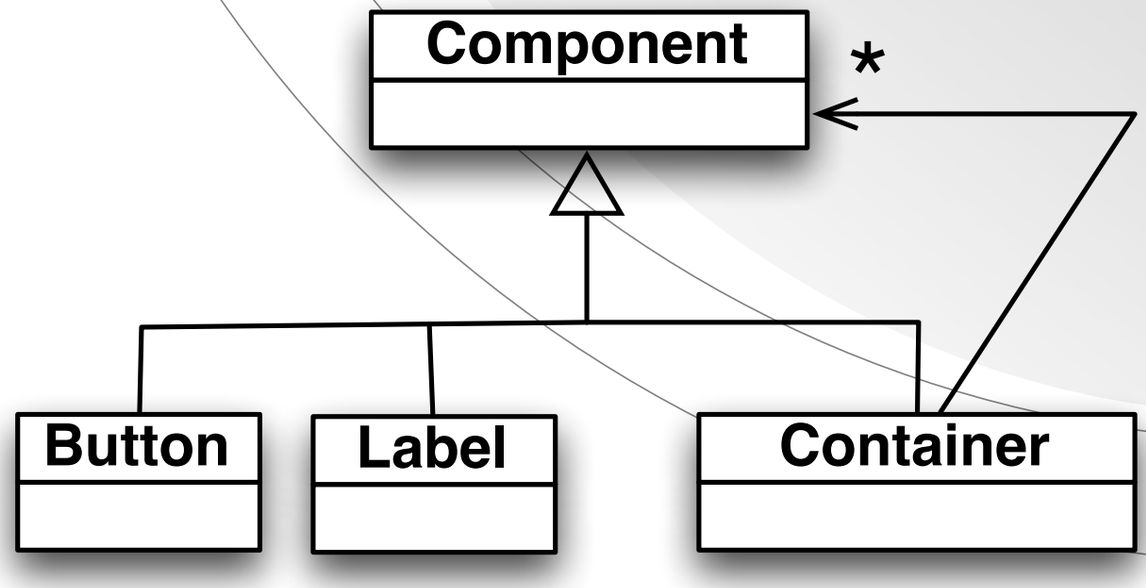
Behavioral		
	State	represent “status” using class
	Strategy	replace an entire algorithm depending on....
	Template Method	delegate actual process to subclasses
	Visitor	do something by acting/ visiting on a series of objects

Copyright © ViSLAB, The University of Sydney

example



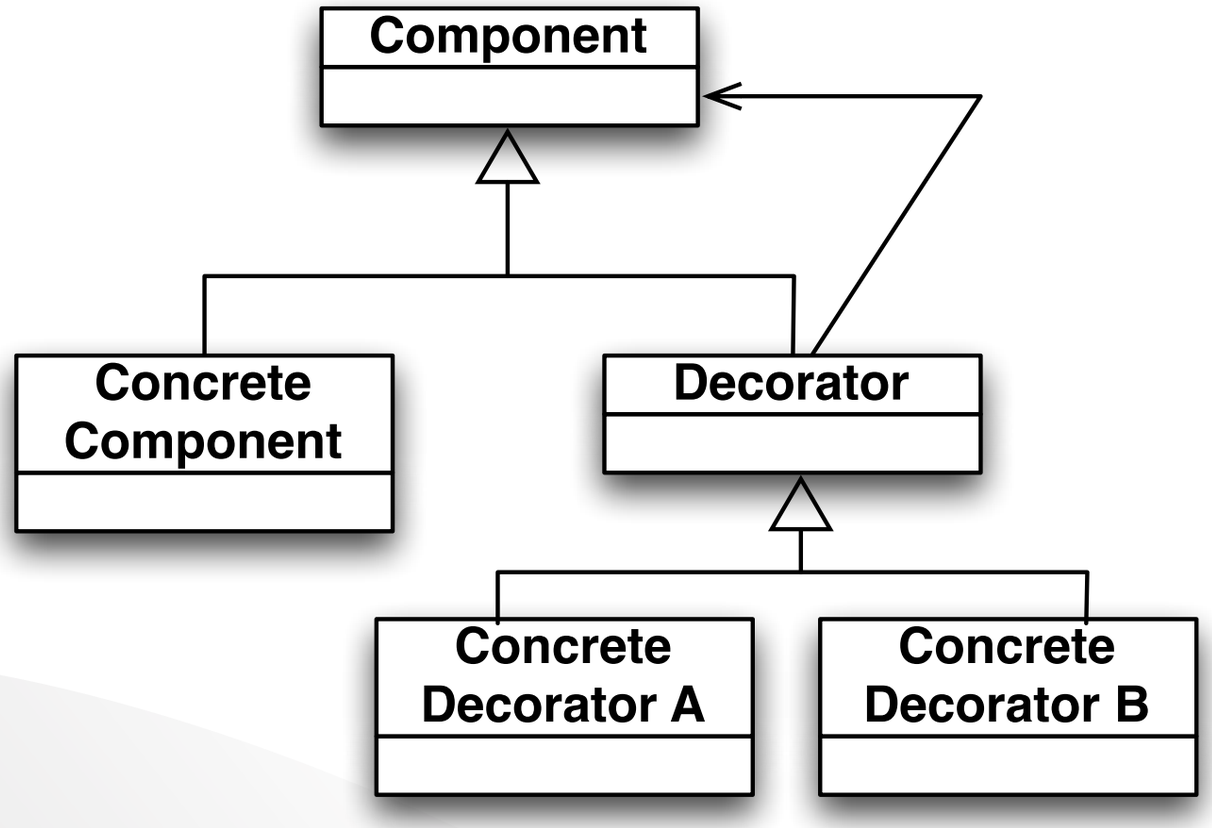
Composite Pattern



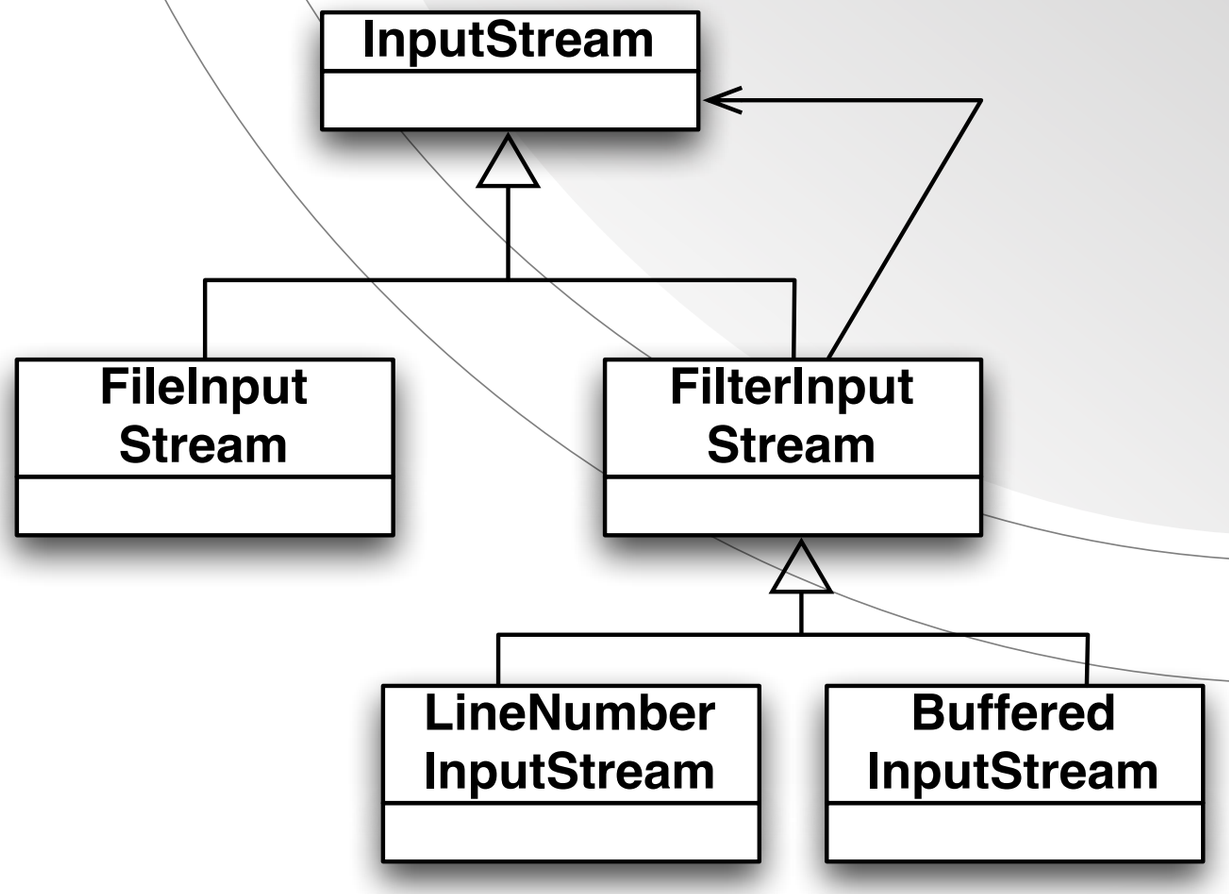
java.awt package

Copyright © ViSLAB, The University of Sydney

example



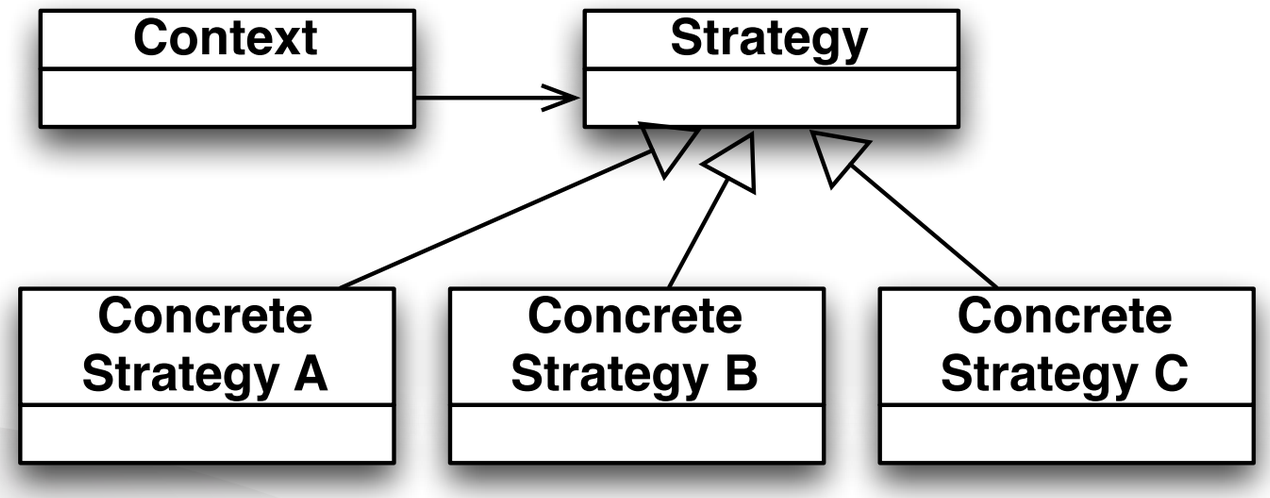
Decorator Pattern



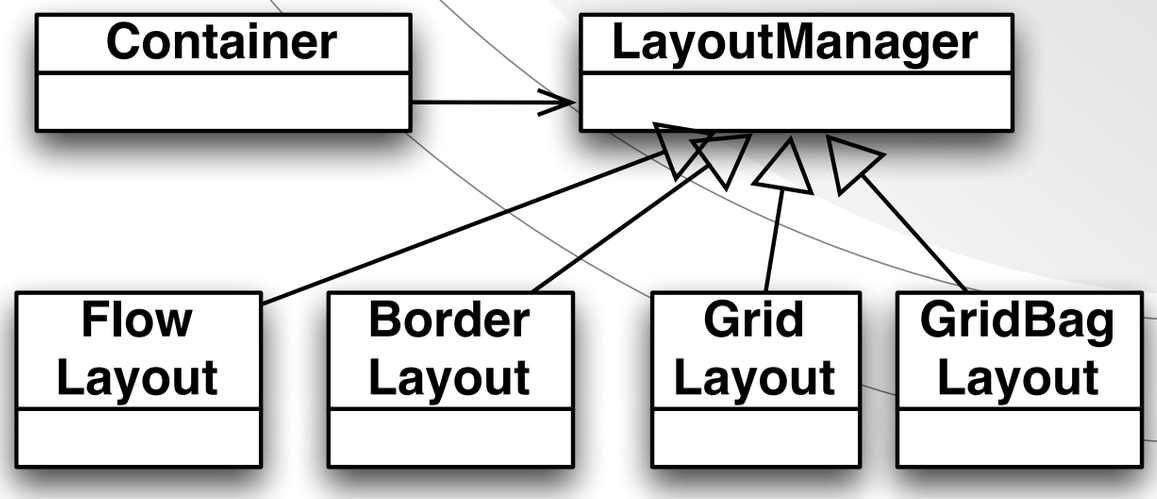
java.io package

Copyright © ViSLAB, The University of Sydney

example



Strategy Pattern



Java.awt package

Copyright © ViSLAB, The University of Sydney

Some class has names of pattern

- Iterator
- Observer, etc.

Copyright © ViSLAB, The University of Sydney

There are may patterns

- GoF...Design Patterns has 23 patterns...but
- there are more...and may be more..

Copyright © ViSLAB, The University of Sydney

learning patterns

1. associate
   - “Pattern Name”,
   - “its purpose”,
   - “form” (structure...UML)
2. use them to study others’ work (library/ frameworks, etc)
3. use them for your own design
4. use to exchange ideas with others

Copyright © ViSLAB, The University of Sydney

- D.P originates in Architectural Patterns
- C.S. has
  - named data structures and algorithms
    - list, linked-list, stack, heap
    - binary search, quick sort, bubble sort,...
- but D.P is more sophisticated
  - based on OO.

Copyright © ViSLAB, The University of Sydney

Other Patterns

Name	Description
Specific D.Ps.	J2EE pattern, EJB pattern, multithread pattern, etc
Analysis	system analysis, requirement analysis describe problems
Architectural	Overall software/system structure
Idiom	P.Lang techniques
Refactoring	change internal without changing funcitonality
Process	System dev. process patterns
Anti patterns	list of don't do it!

Copyright © ViSLAB, The University of Sydney

Use D.P. to be Lazy but High Quality Code

- appropriate use of D.P. will allow you to
  - write less code
  - build re-usable modules
- all happening at Design Stage!

Copyright © ViSLAB, The University of Sydney