15CSE202 Object Oriented Programming
Lecture 2

Object Oriented Concepts

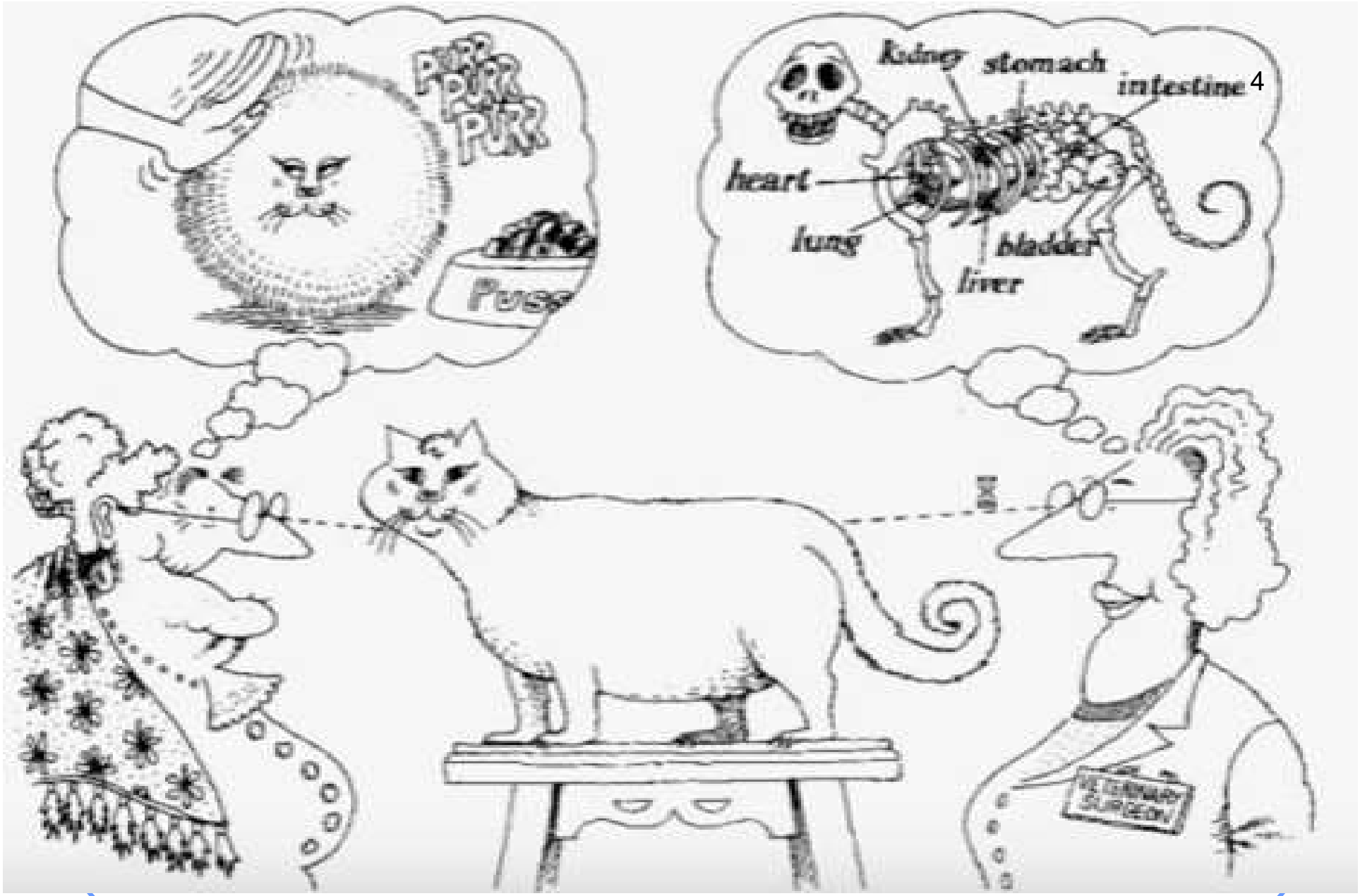
Nalinadevi Kadiresan
CSE Dept.
Amrita School of Engg.

Roadmap

- Abstraction
- Encapsulation
- Inheritance
- Modularity
- Strong Typing
- Concurrency
- Persistence

Abstraction - Modelling

Abstraction focuses upon the essential characteristics of some object; relative to the perspective of the viewer.



Abstraction focuses upon the essential characteristics of some object; relative to the perspective of the viewer.

Problems with Procedural Languages

struct stack

```
{   int top;
    char* store;} ;
```

struct queue

```
{   int front;
rear;
    char* store;} ;
```

push();

pop();

insert();

delete();

All functions are global!!

Problems with Procedural Languages

struct horse

{weight, color, Age;};

struct eagle

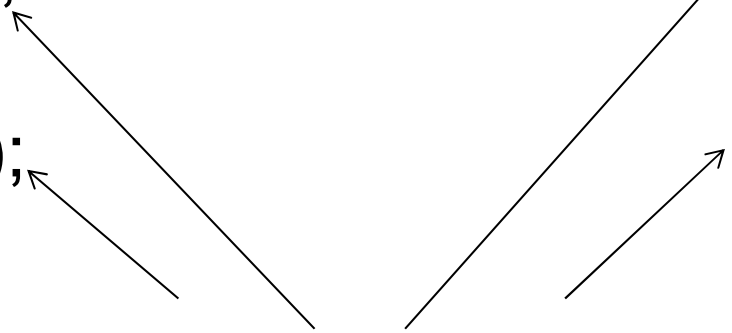
{ age, weight wingspan;} ;

gallop();

canter();

fly();

hunt();



All functions are global!!

Better Abstraction using OO Language

class stack

```
{ int top;
  char* store;
 ---------------
 push();
 pop();      };
```

class queue

```
{   int front;
rear;
    char* store;}
 ---------------
 insert();
 delete();   };
```

Class = State+ Behaviour

Better Abstraction using OO Languages

class Horse

```
{weight, color,
age;
-----------------
 gallop();
 canter();    };
```

class Eagle

```
 {  age, weight
wingspan;
-----------------
 fly();
 hunt();   };
```

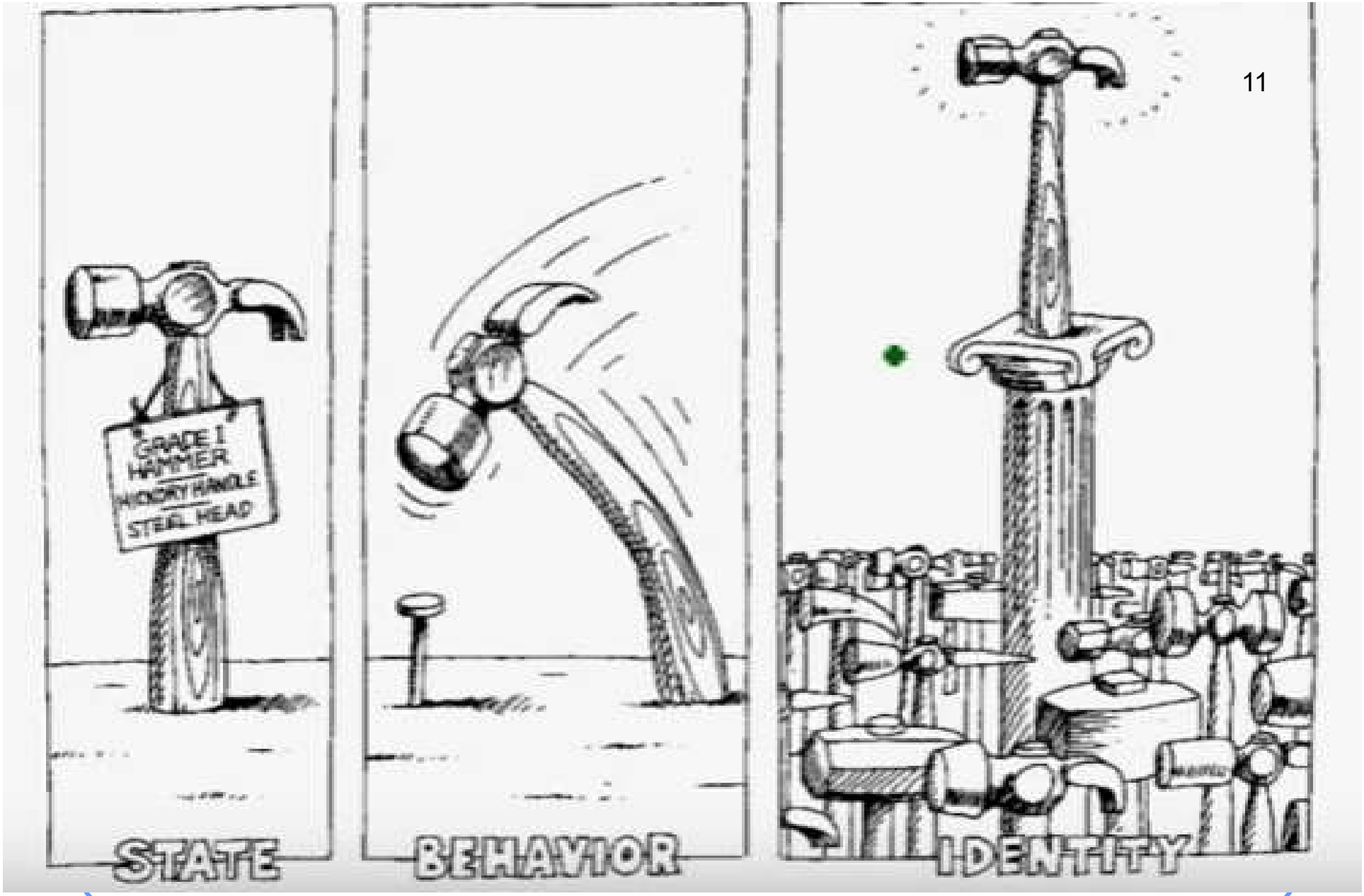
Class = State+ Behaviour

Class = State+ Behaviour

- State
  - data members
  - fields
  - properties
- Behaviour
  - member functions
  - methods

Objects

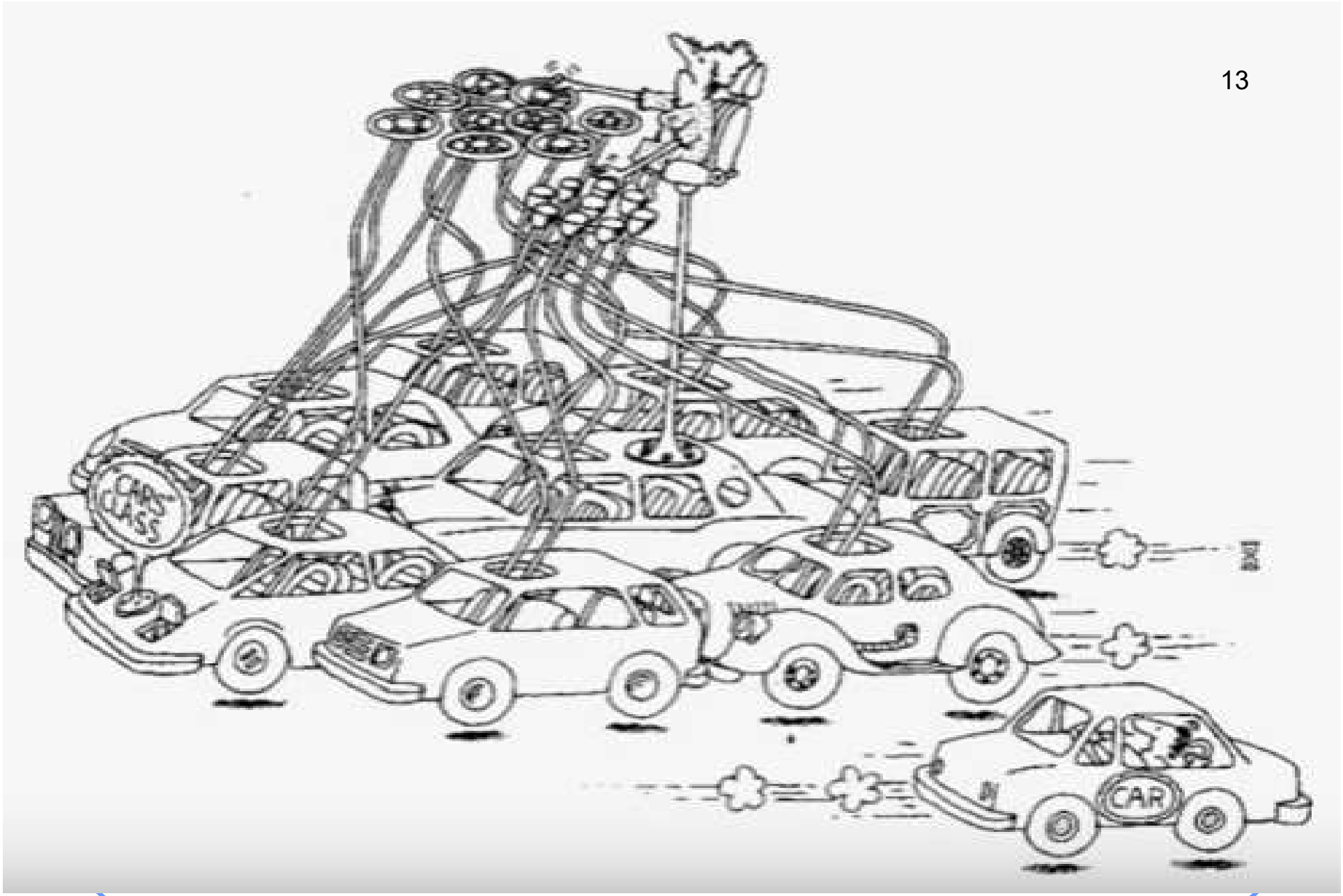
An object has **state**, exhibits some well defined **behaviour**, and has a unique **identity**.



An object has state, exhibits some well defined behaviour, and has a unique identity

Class

A class represents a set of objects that share a common structure and a common behaviour.



A class represents a set of objects that share a common structure and a common behaviour.

Objects in Real World

DOG

State	Name Color Breed Hungry
Behavior	Barking Fetching Wagging Tail

BICYCLE

State	Current Gear Current Speed Current Pedal Cadence (rhythm)
Behavior	Changing gear Changing pedal cadence Applying brakes

Objects in Real World

Try and identify the State & behavior of the two objects .

TABLE LAMP

State	?
Behavior	?

DVD PLAYER

State	?
Behavior	?

श्रद्धावान् लभते ज्ञानम्

TABLE LAMP

State	On/Off
Behavior	Turning On Turning Off

DVD PLAYER

State	On /Off Current Volume Current Station
Behavior	Turn on Turn off Increase volume Decrease volume, Seek Scan Tune

Roadmap

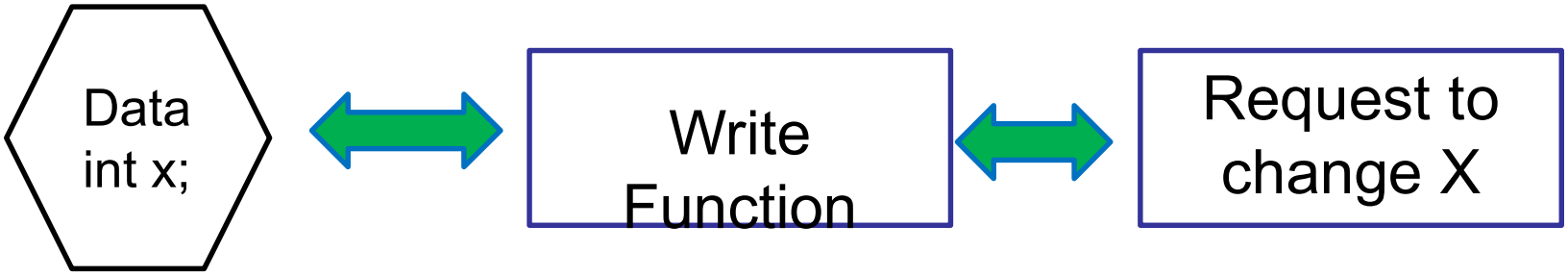
- Abstraction
- Encapsulation
- Inheritance
- Modularity
- Strong Typing
- Concurrency
- Persistence

Problems with my Public Garden

Solution 1: Build a high fence

Write functions to Safeguard Data

$0 \leq x \leq 100$



Private!
Hidden !

Write function Code

```
void modify_x(int newval)
{
If(newval>100 || newval <0)
        {PRINT ERROR AND EXIT}
else
      x = newval;
}
```

Read Function Code

```
int read_x()
{
return x;
}
```

Encapsulation

- FIRST LAW OF OOP: Data must be hidden, i.e.,PRIVATE
- Read access through read functions
- Write access through write functions
- For every piece of data, 4 possibilities

  >>read and write allowed

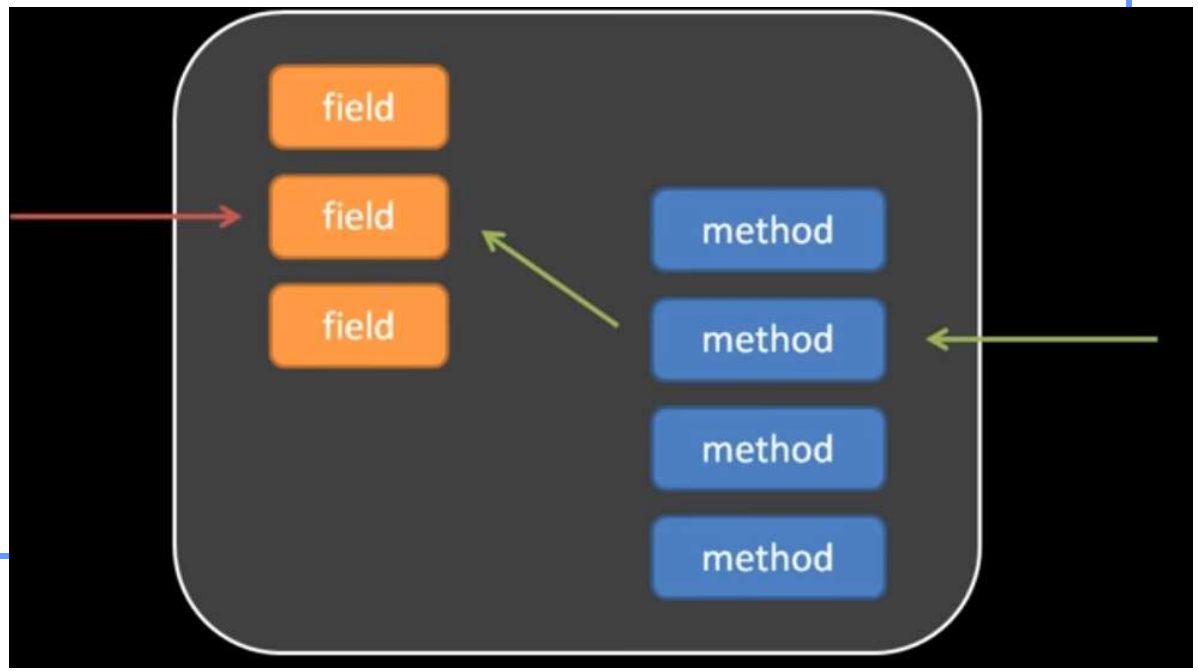
  >>read only

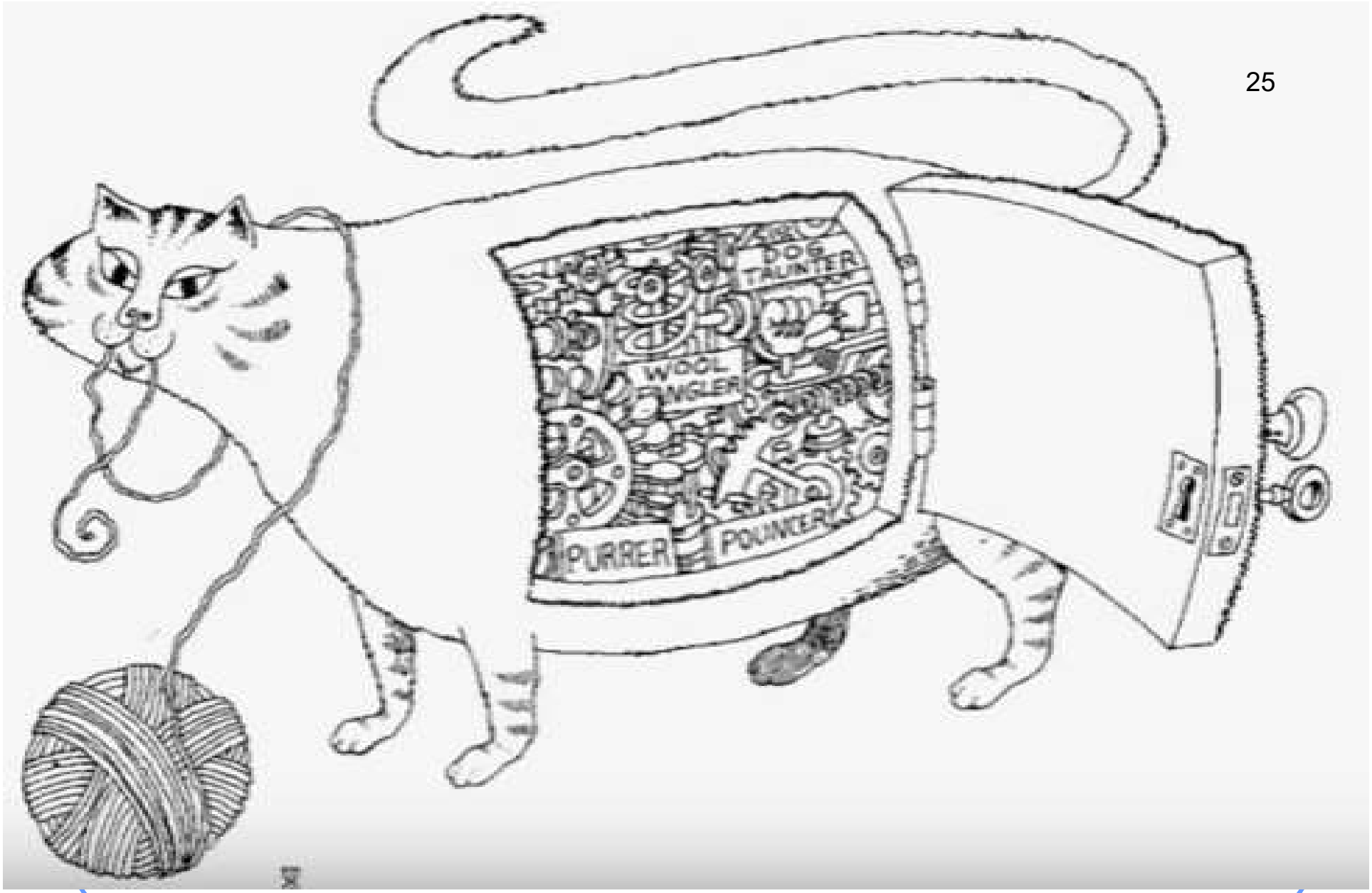
  >>write only

  >>no access

Encapsulation

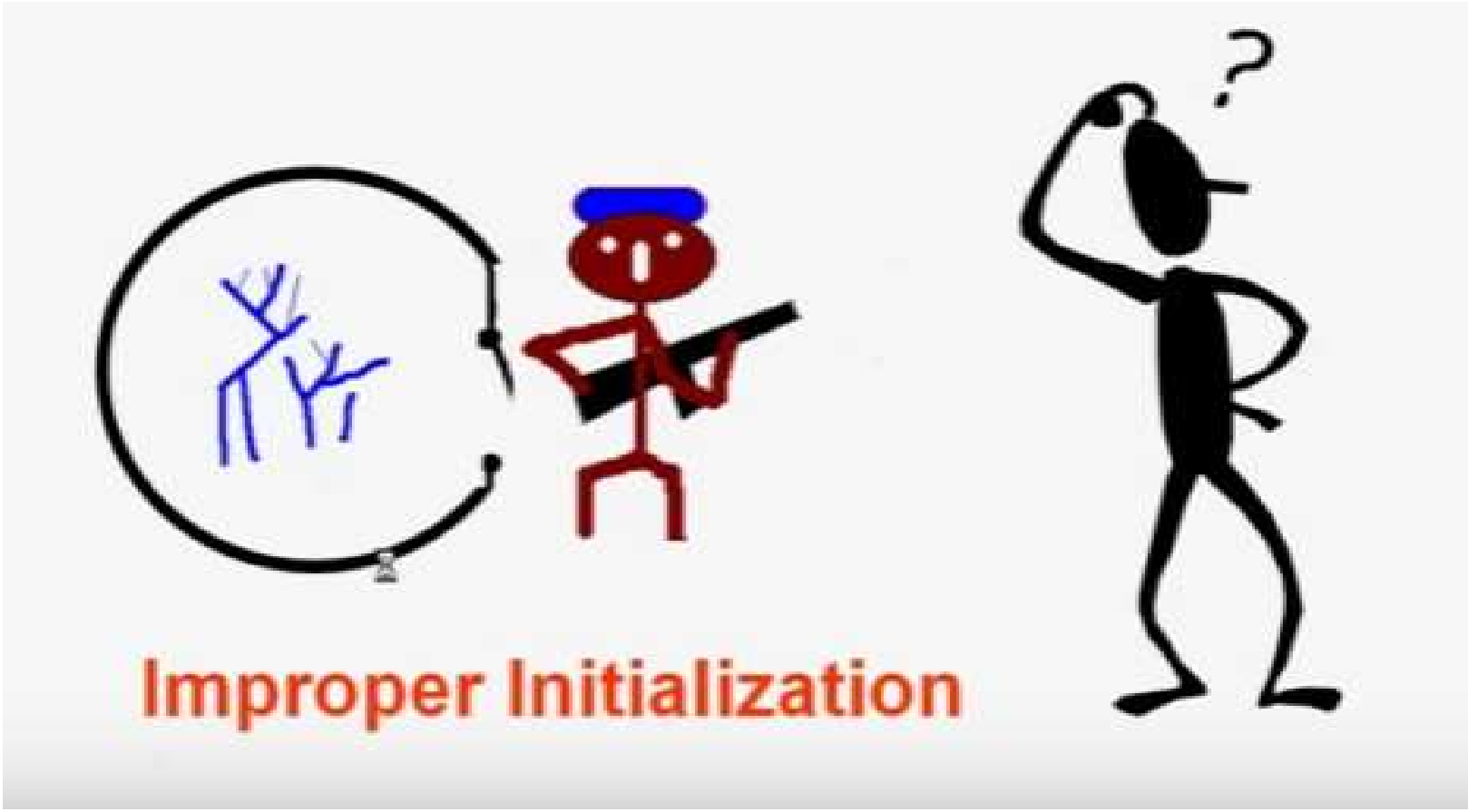
Encapsulation hides the details of the implementation of an object.





Encapsulation hides the details of the implementation of an object.

Problem: Garden has only weeds



Initialization of Objects

- Special Functions – CONSTRUCTORS ensure correct initialization of all data, automatically called at the time of object creation.

Resource Deallocation

- Special Functions – DESTRUCTORS

  ensure correct initialization of all resources before an object “dies” (goes out of scope).

Lifecycle of an Object

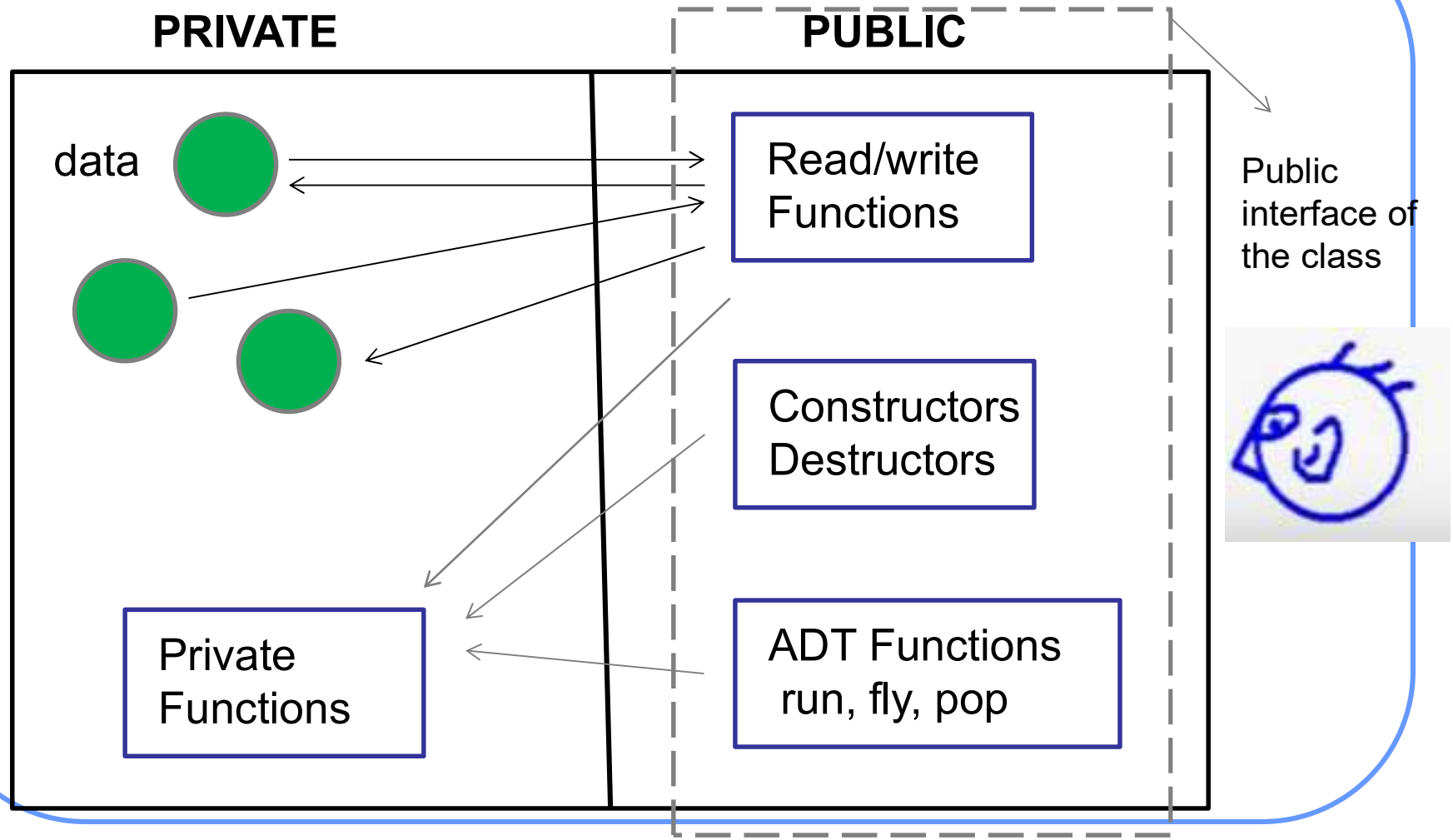
- Born healthy
  - using constructors
- Lives Safely
  - using read/write functions
- Dies cleanly
  - using destructors

Lifecycle of an Object

- Born healthy
  using constructors **BRAHMA**
- Lives Safely
  using read/write functions **VISHNU**
- Dies cleanly
  using destructors **MAHESHWARA**

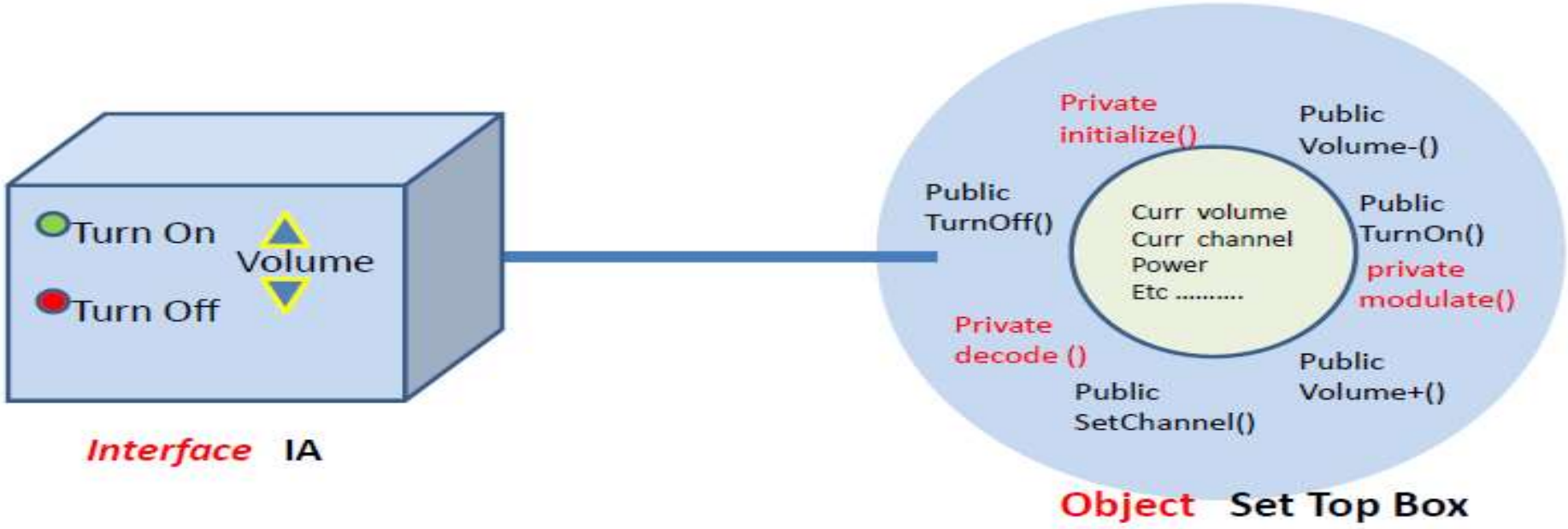
HOLY TRINITY of Object Oriented Programming

Anatomy of a Class

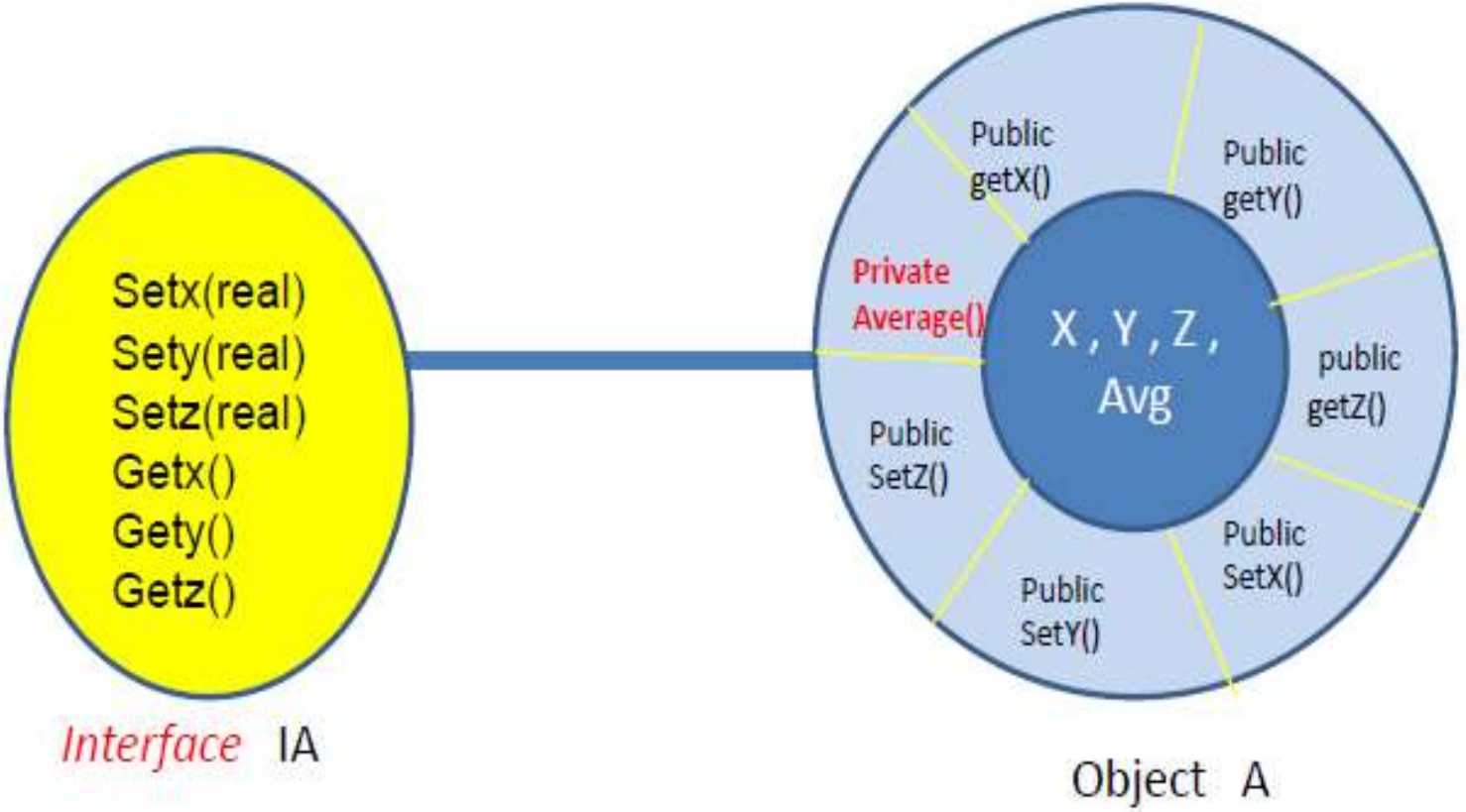


Objects – public interface

Other objects can change the state of an object by using only those methods that are exposed to the outer world through a public interface. This help in data security.



Objects – Public Interface



Roadmap

- Abstraction
- Encapsulation
- Inheritance
- Modularity
- Strong Typing
- Concurrency
- Persistence

Code Reuse - Inheritance

- class Stack has functions pop and push

  Can we add peek function without having the source code with us?

- Extending the functionality of a class

  OR

  Specializing the functionality of a class

  Stack

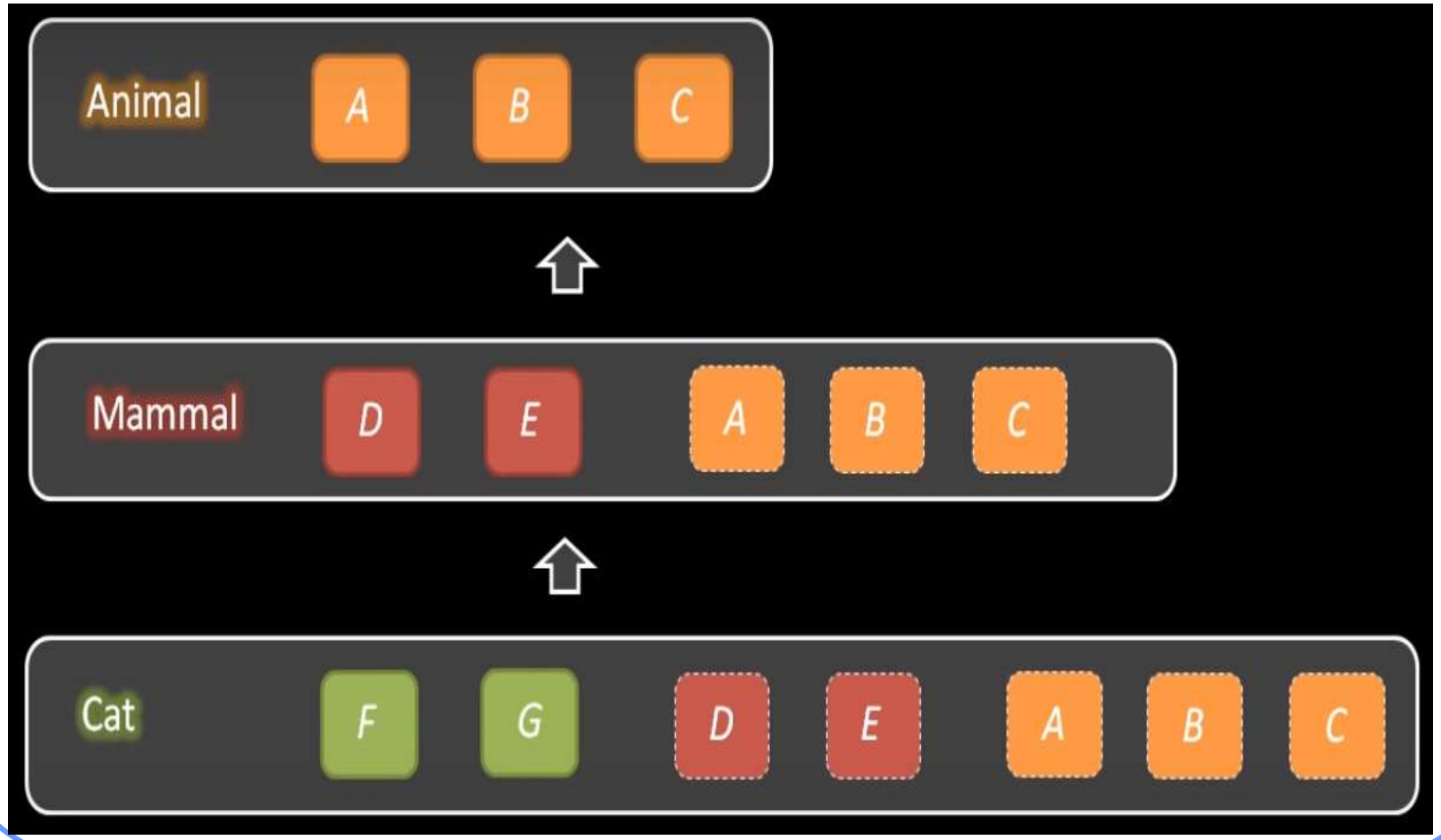
  Stack 1

Subclasses - Inheritance

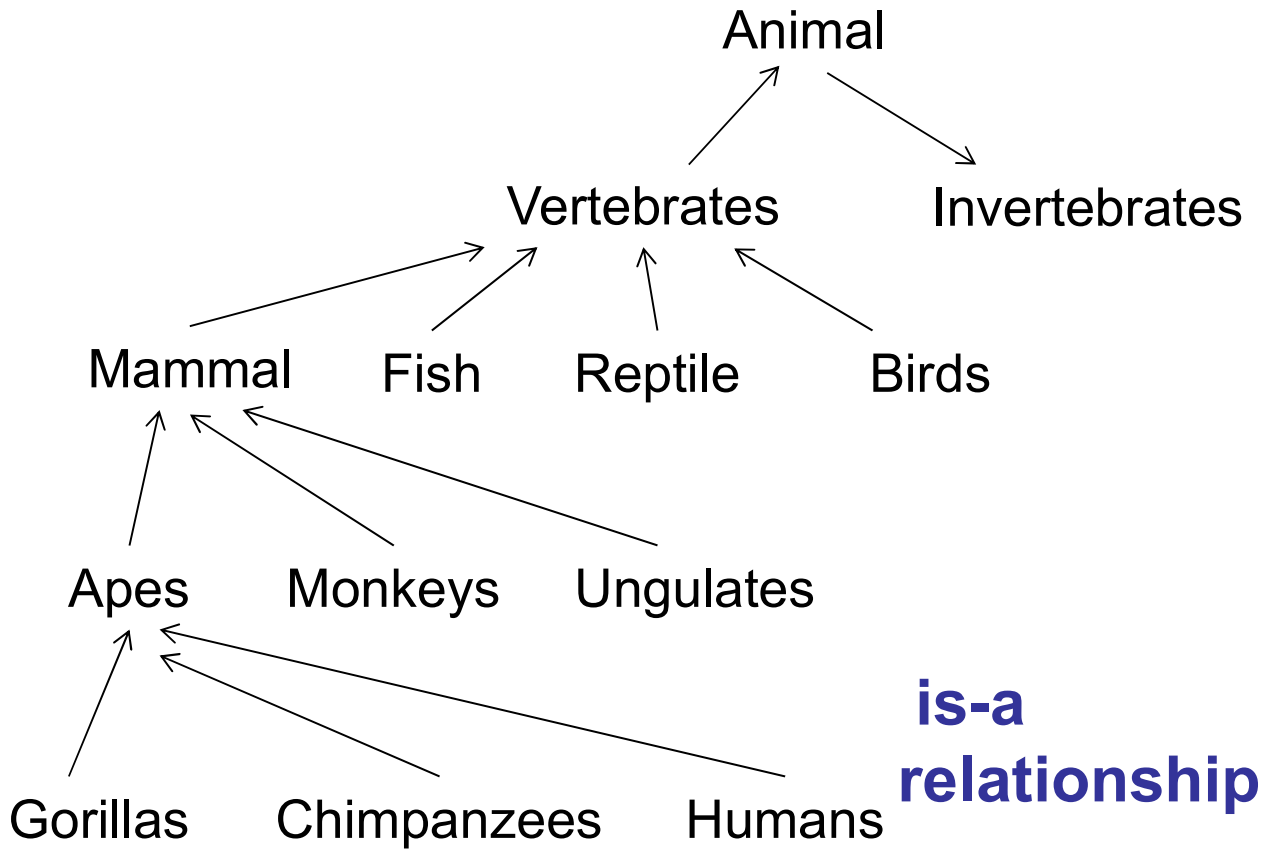
A subclass may inherit the structure and behaviour of its superclass.

A subclass may inherit the structure and behavior of its superclass.

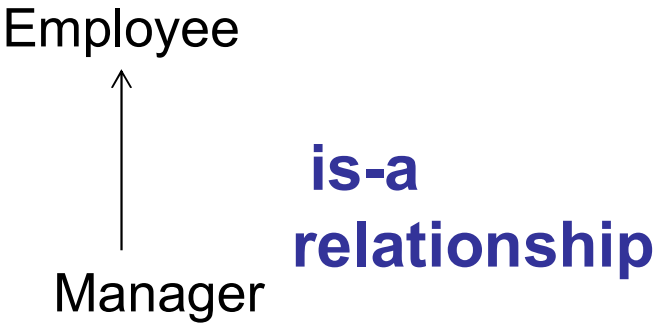
Inheritance



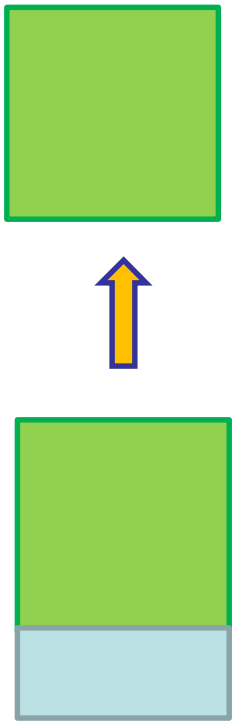
Inheritance Trees



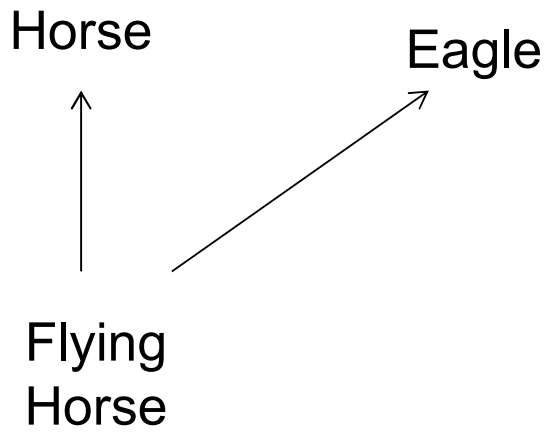
Inheritance Trees



Nomenclature

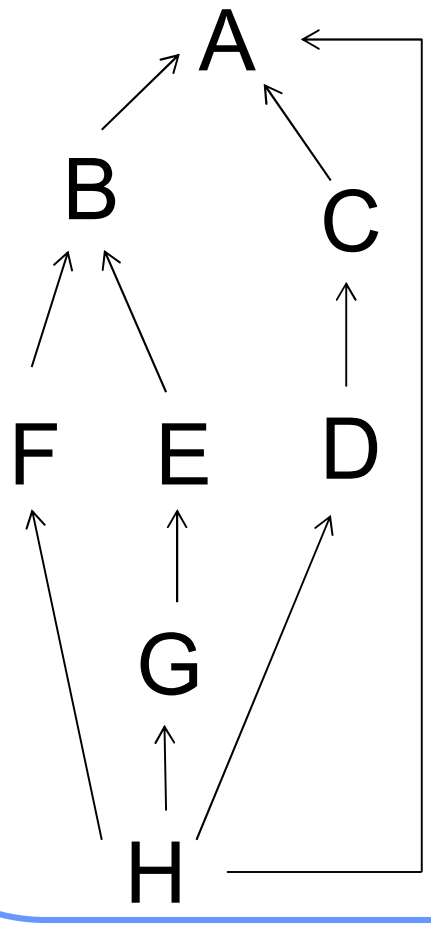
	Not recommended		BEST
	Superclass	Parent class	Base class
	Subclass	Child class	Derived class

Multiple Inheritance



One class having more than one base class

Graph Structure of Multiple Inheritance

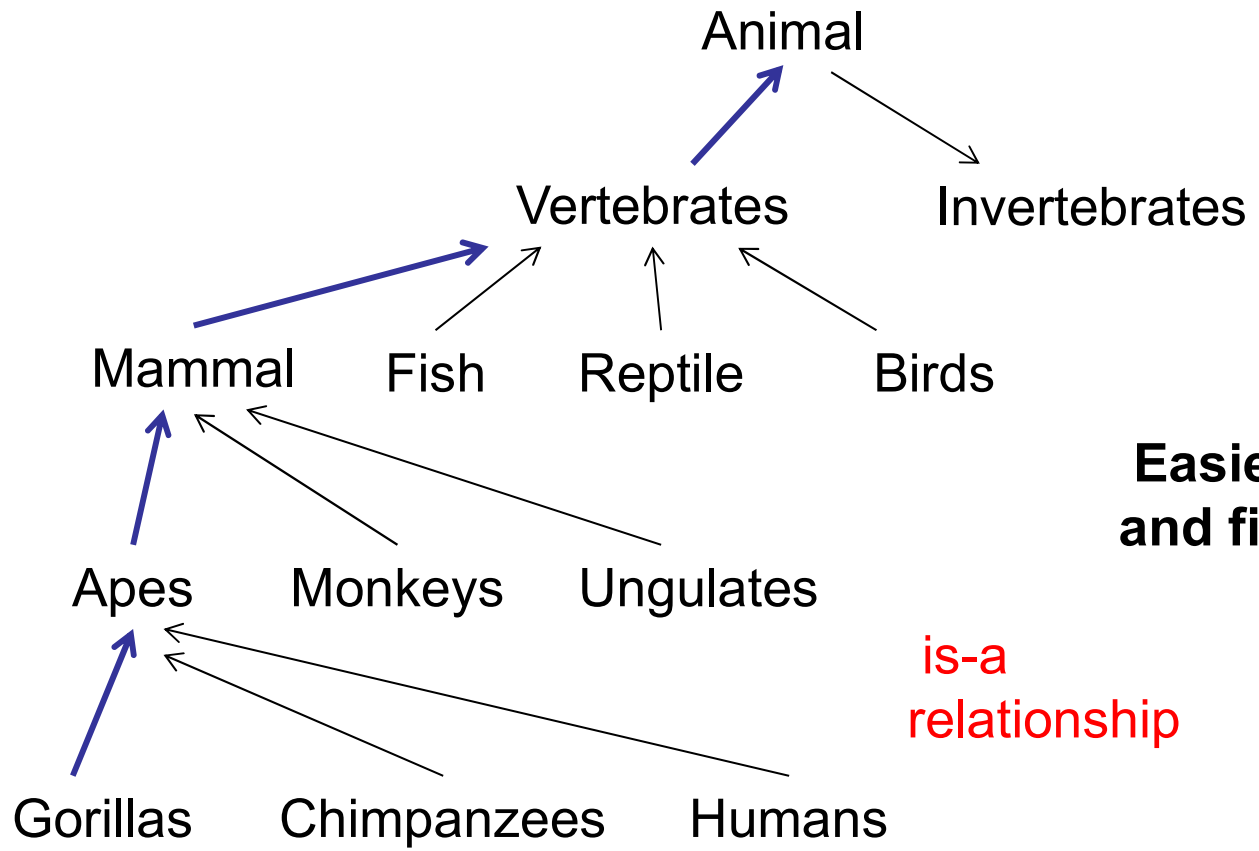


Graphs have more complicated structure than trees.

More difficult to maintain and fix bugs etc.

Ambiguity in function calls.

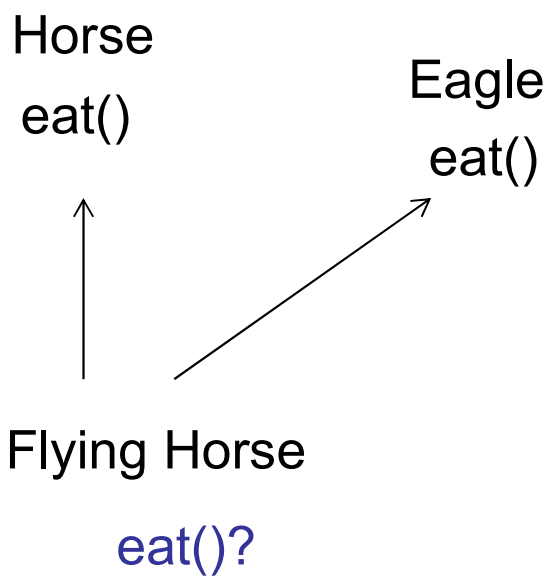
Inheritance Trees



Easier to maintain and fix bugs

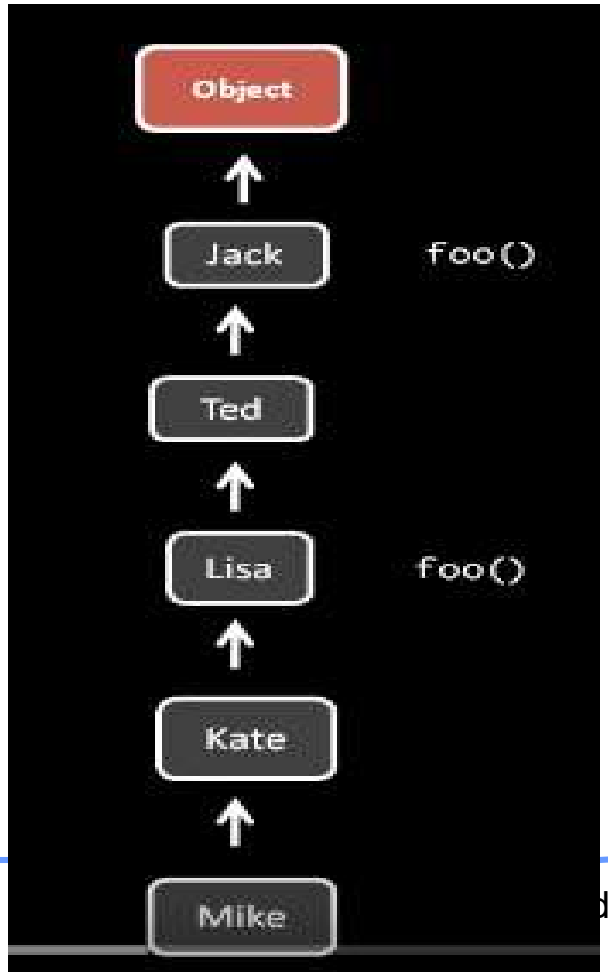
is-a relationship

Ambiguity in Multiple Inheritance



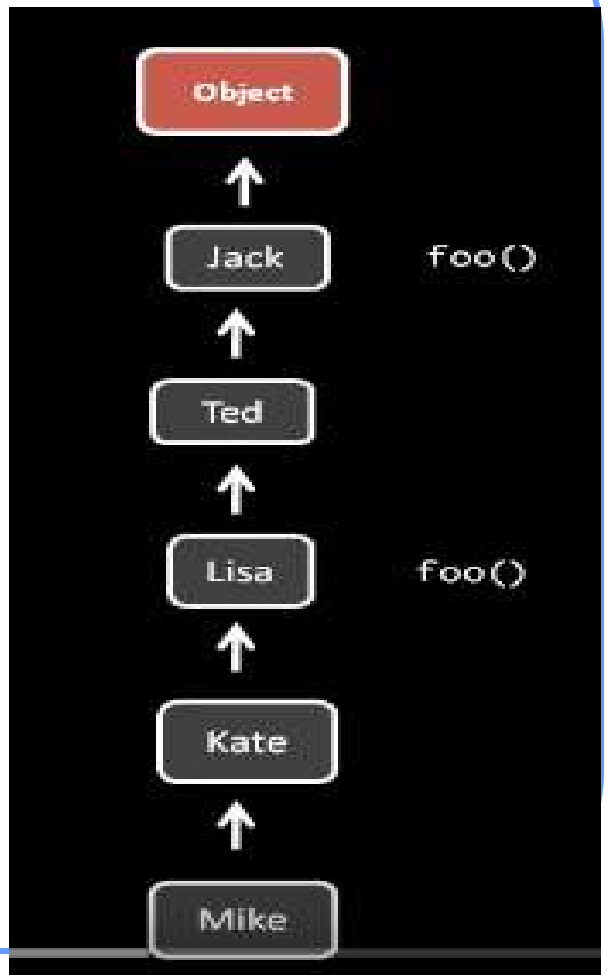
Overridding

- Redefining a inherited method



Polymorphism

- Depends on the type of object
- Eg. x.foo()



Roadmap

- Abstraction
- Encapsulation
- Inheritance
- Modularity
- Strong Typing
- Concurrency
- Persistence

Modularity

Modularity packages abstractions into discrete units.

- Classes
- Packages
- Domains

Modularity packages abstractions into discrete units.

Roadmap

- Abstraction
- Encapsulation
- Inheritance
- Modularity
- Strong Typing
- Concurrency
- Persistence

Strong Typing

Strong typing prevents mixing abstractions.

X = Y

is allowed only if X and Y are objects of the same class.

Weak typing: Allow some violations (e.g., in C)

Untyped: Dynamic Typing (LISP, javascript, etc.)

Strong typing prevents mixing abstractions.

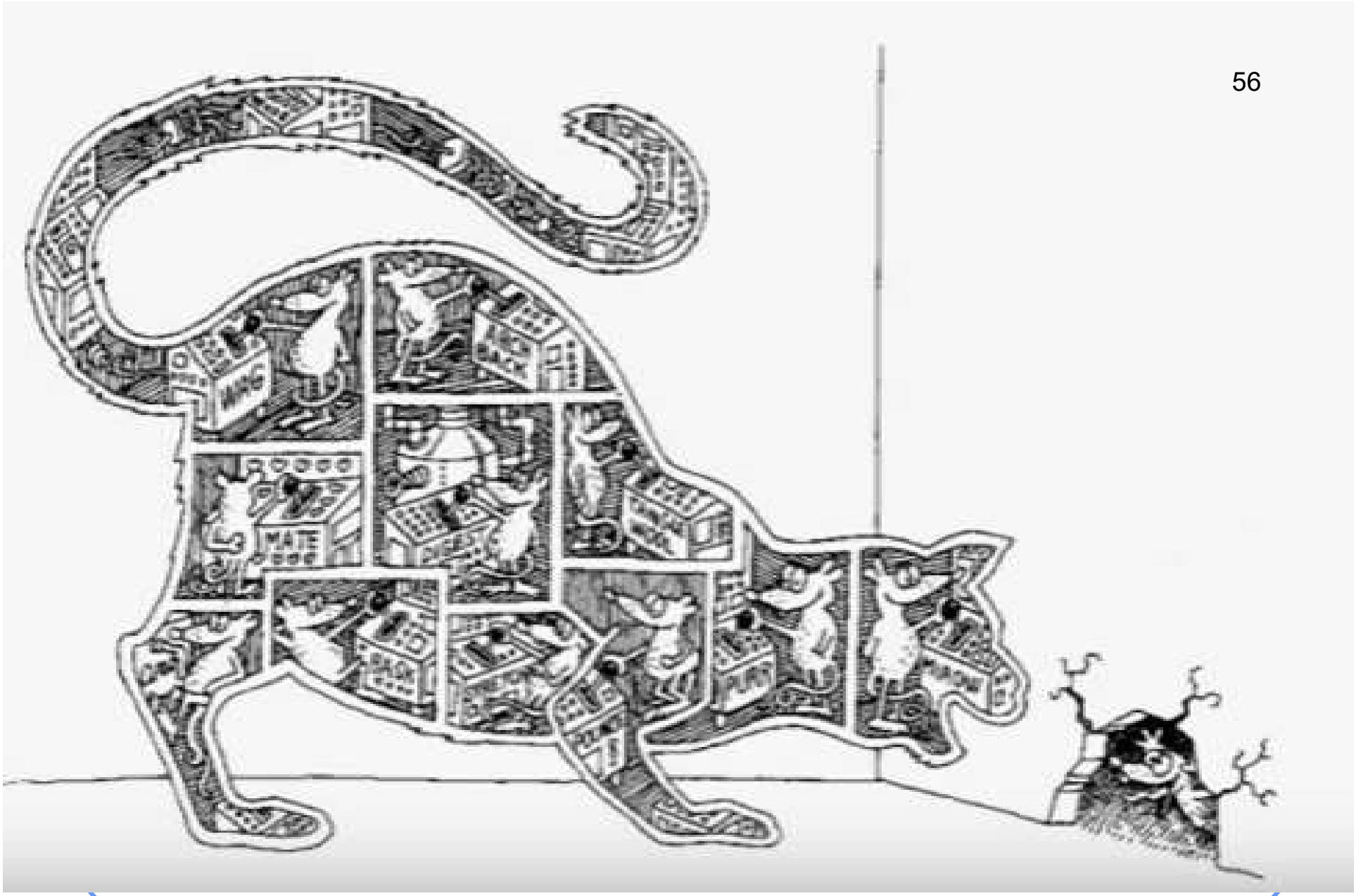
Roadmap

- Abstraction
- Encapsulation
- Inheritance
- Modularity
- Strong Typing
- Concurrency
- Persistence

Concurrency

Concurrency allows different objects to act at the same time.

Java's support of multithreading is a form of concurrency.



Concurrency allows different objects to act at the same time.

Roadmap

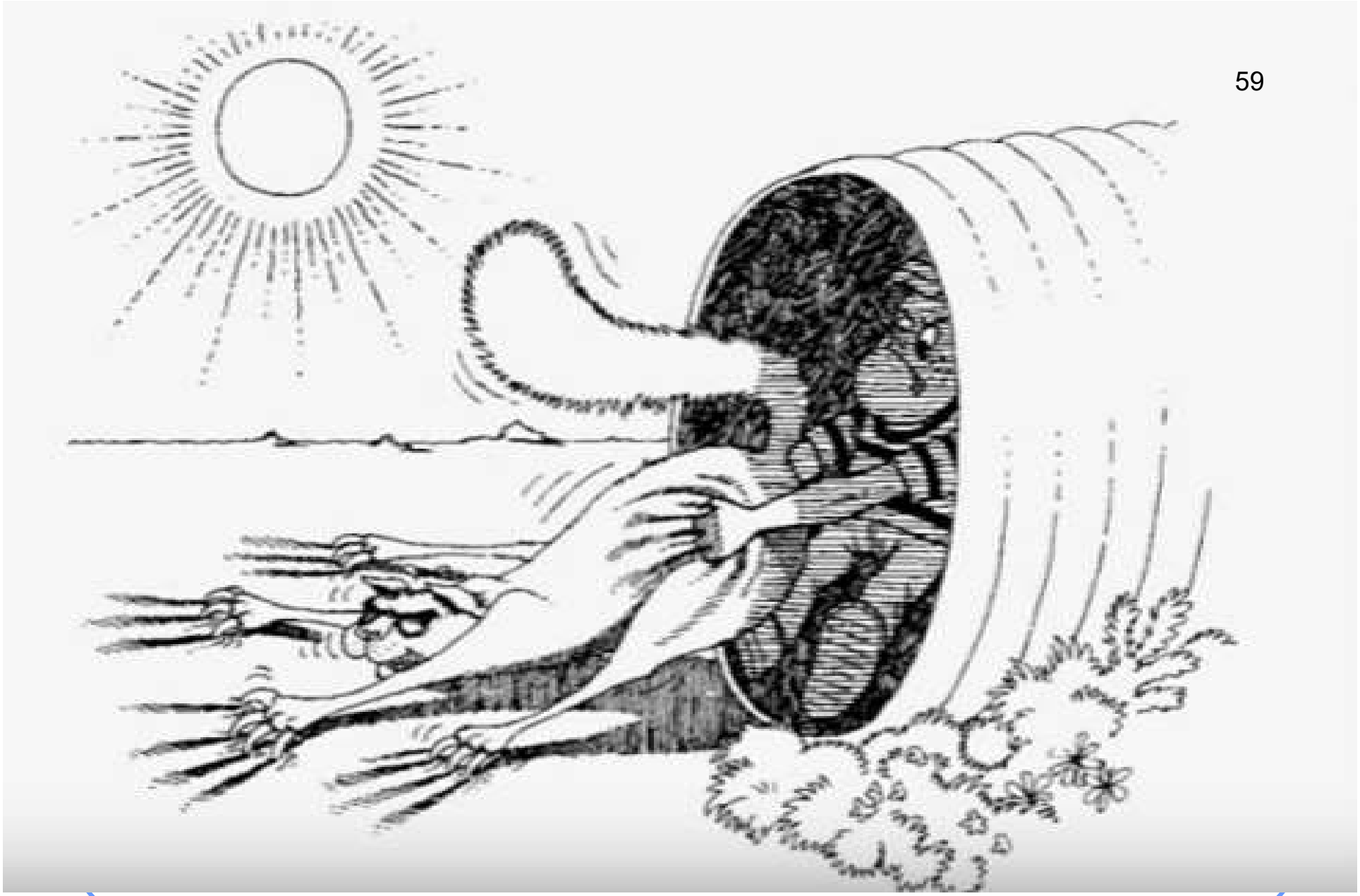
- Abstraction
- Encapsulation
- Inheritance
- Modularity
- Strong Typing
- Concurrency
- Persistence

Persistence

Persistence saves the state and class of an object across time and space, i.e., storage on permanent storage media.

Object Serialization is supported in Java.

Can also use mapping to RDBMS using some Object/Relational Mapping Scheme.



Persistence saves the state and class of an object across time and space

Recap

- Abstraction
- Encapsulation
- Inheritance
- Modularity
- Strong Typing
- Concurrency
- Persistence

Classification of programming languages according to its support of Object-Orientation

object-oriented programming
+

procedural and data-oriented	employ some of the basic imperative structures	no other structure or paradigm
• C++ •Ada 95	•Java •C#	•Smalltak

Dynamic Typed + Multiparadigm
- Javascript
- Python
- Ruby

Next Session will be

Overview of Java