# Class 0

## Philosophy of OOP

# Overview

- POP
- OOP
- OOP features
- Thinking in Object oriented terms

# Procedure Oriented Programming

- Extension to unstructured programming
- Developed to handle increasing complexity
- Provides functions, local variables, rich control structures and less reliant on GOTO
- **Code acting on data** (functions acting on data)
- Emphasis is on procedure or algorithms

# Problems with POP

- No control on data access
- Requires you to think in terms of the structure of the computer rather than the structure of the problem you are trying to solve. The programmer must establish the association between the machine model (in the “solution space,” which is the place where you’re modeling that problem, such as a computer) and the model of the problem that is actually being solved (in the “problem space,” which is the place where the problem exists).
- Inaccurate mapping leads programs that are difficult to write and expensive to maintain for large and complex problems.

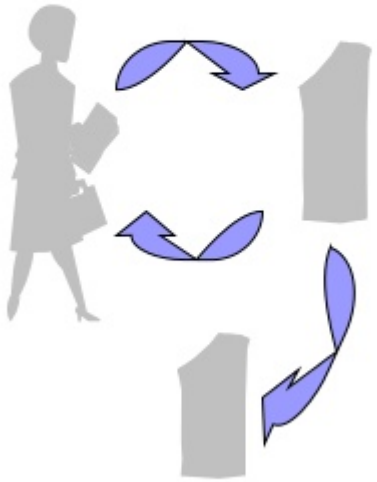
# Object Oriented Programming

- Programs are organized around data, with the key principle being "**data controlling access to code**".
- Provides tools for the programmer to represent elements in the problem space.
- We refer to the elements in the problem space and their representations in the solution space as “objects.”
- Thus, OOP allows you to describe the problem in terms of the problem, rather than in terms of the computer where the solution will run.

# POP vs OOP

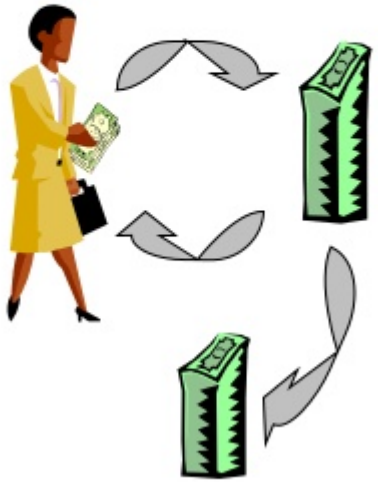
## Procedural vs. Object-Oriented

- Procedural



Withdraw, deposit, transfer

- Object Oriented



Customer, money, account

# Features of OOP

- Objects & Classes
- Data Abstraction
- Data Encapsulation
- Data Hiding
- Message Passing
- Inheritance
- Polymorphism
  - Overloading
  - Dynamic Binding

# Object

- An  Object represents a real world entity which can be physical or logical
- Every Object has state and behavior
- State refers to **properties**(static) of the object and its current values(dynamic)
- Behavior is how an object reacts, in terms of its state changes and message passing, **operations** performed on the object.

# Example

- Dog Object
- Properties
  - Breed
  - Colour
- Behaviour
  - Barks
  - Wag tail
  - Bites!!

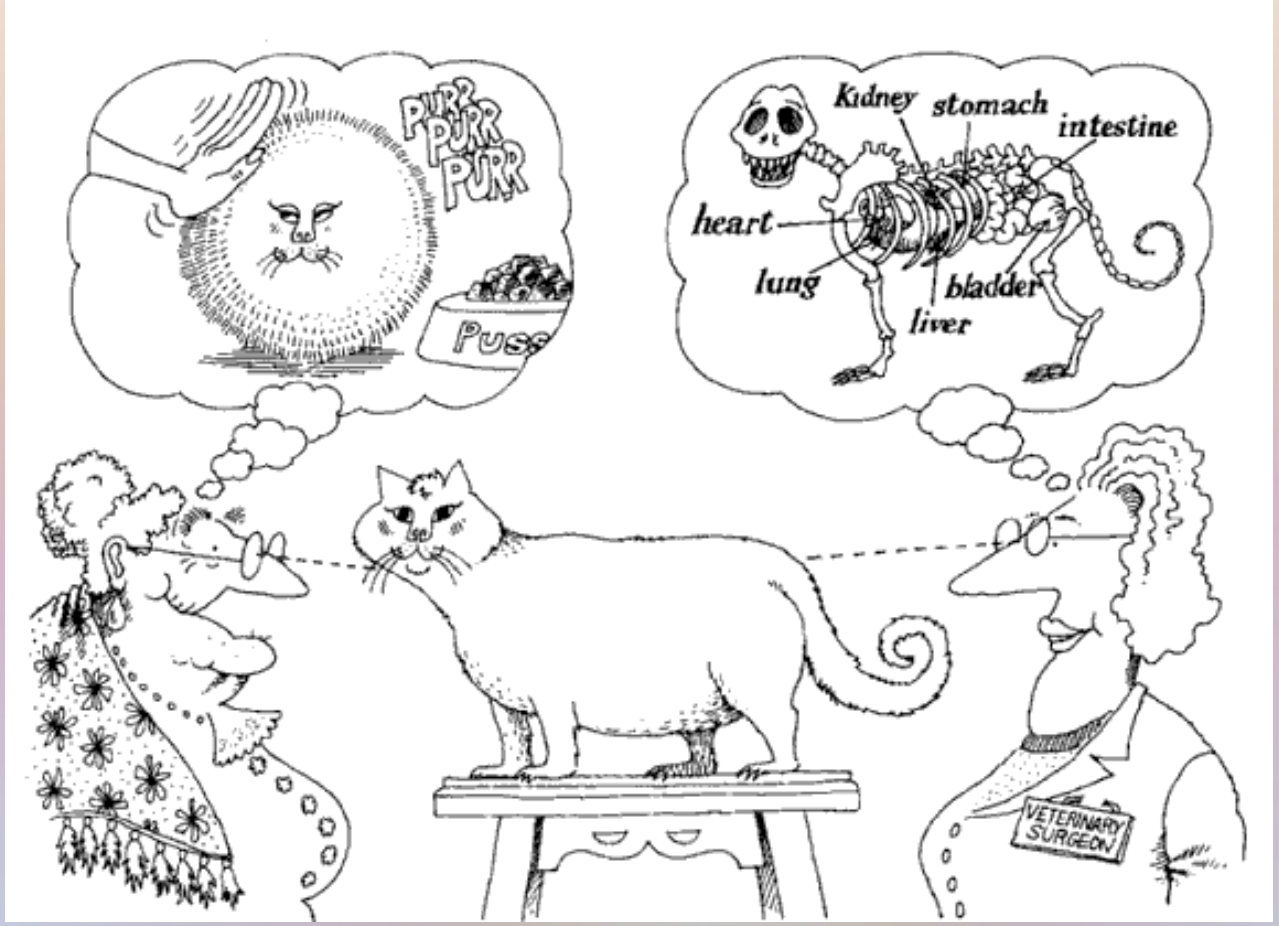
# Class

- It is a set of objects that share a common structure and a common behavior.
- It is a user defined data type that allows a collection of data members and member functions
- Class acts as a template for creating objects
- An Object is not a class but an instance of a class

# Data Abstraction

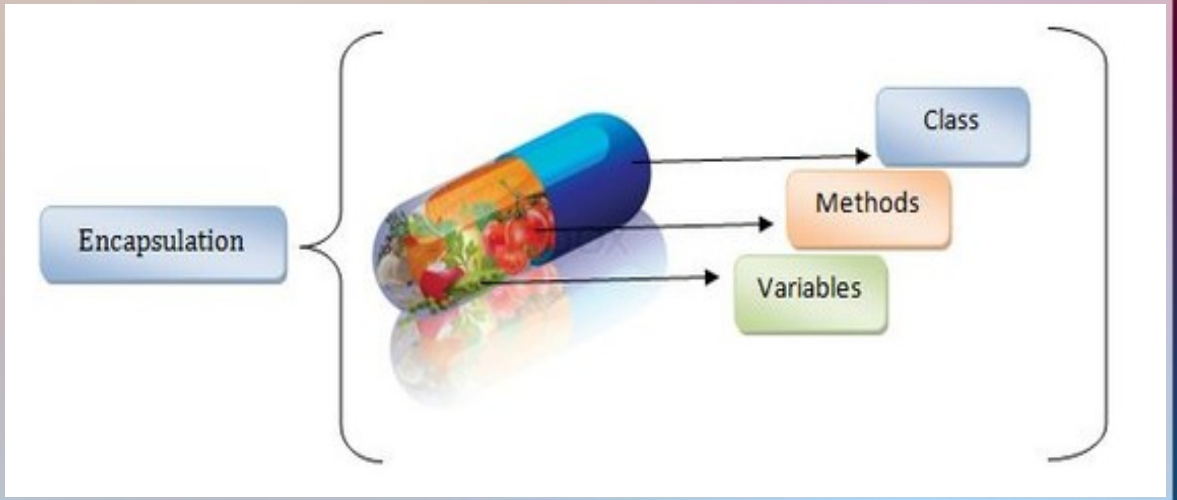
- A good abstraction is one that emphasizes details that are significant to the reader and suppresses details that are insignificant
- A means of coping with complexity
- Focuses on outside view

# Data Abstraction



# Data Encapsulation

- Encapsulation is the mechanism that binds together code and the data it manipulates, and keeps both safe from outside interference and misuse.
- Data is encapsulated or wrapped around with functions only which can access this data
- Brings in access control
- Visibility specifier



# Data Hiding

- Outcome of data encapsulation
- Controlling access to data from external members or entities
- Thereby data is secured
- Less likelihood of errors

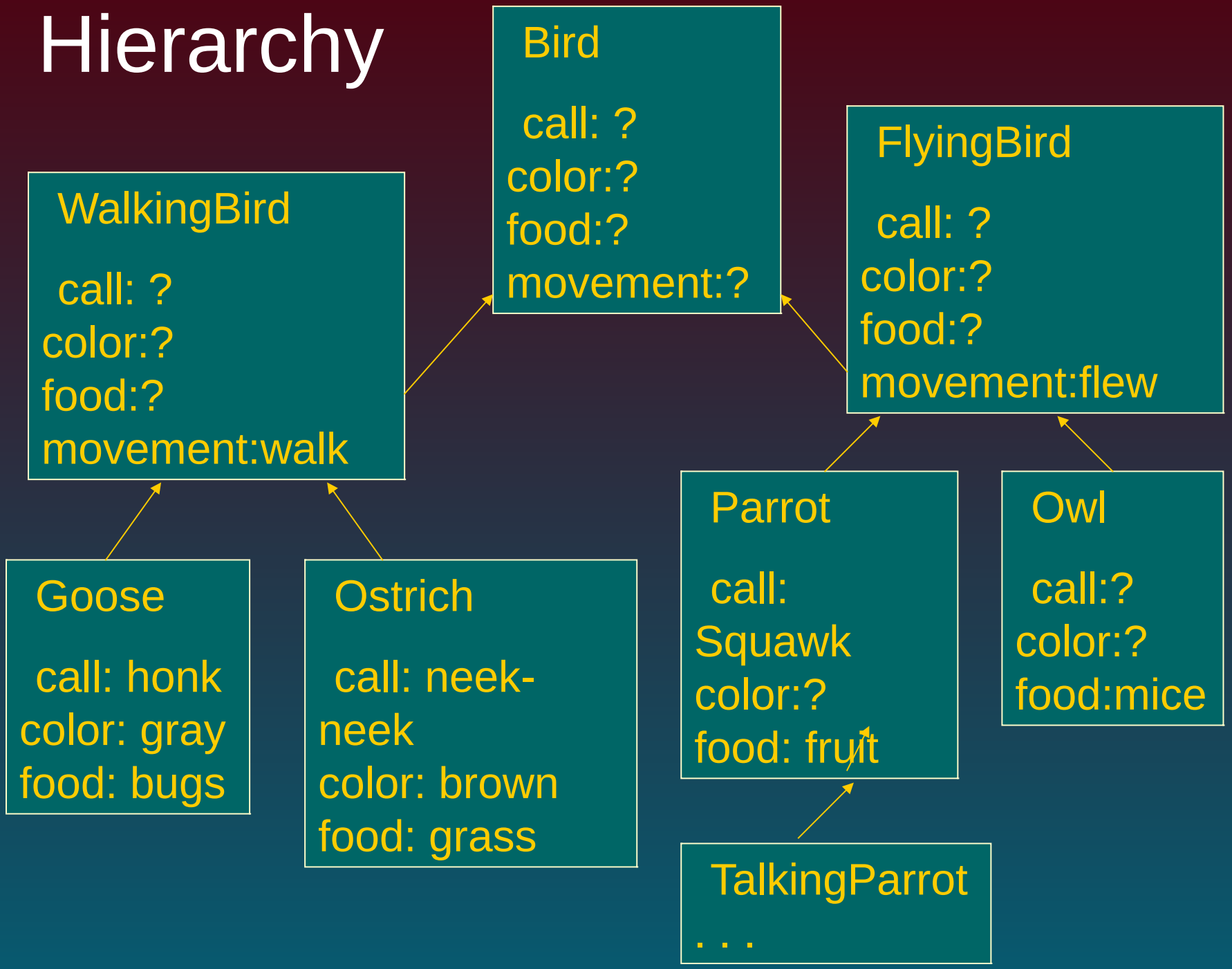
# Message Passing

- Way by which objects can interact with one another
- Objects invoke member functions of another objects

# Hierarchy

- Whole – Part relationship
- Generalization – Specialization
- Inheritance is the process by which one object can acquire the properties of another object.
- supports the concept of classification.

# Hierarchy


Bird
call: ?
color:?
food:?
movement:?
WalkingBird
call: ?
color:?
food:?
movement:walk
FlyingBird
call: ?
color:?
food:?
movement:flew
Goose
call: honk
color: gray
food: bugs
Ostrich
call: neek-neek
color: brown
food: grass
Parrot
call: Squawk
color:?
food: fruit
Owl
call:?
color:?
food:mice
TalkingParrot
. . .

# Polymorphism

- "one interface, multiple methods."
- polymorphism is the attribute that allows one interface to control access to a general class of actions.
- Polymorphism helps reduce complexity by allowing the same interface to be used to access a general class of actions.
- It is the compiler's job to select the specific action as it applies to each situation.

# Object Oriented Programming

***Object Oriented Programming is a method of implementation in which programs are organized as cooperative collection of objects, each of which represents an instance of some class, and whose classes are all members of hierarchy of classes united via inheritance relationships.***

# Thinking in an Object Oriented Way

- Understand the mini world from the problem description.
- Identify Role Players or Actors.
- Usually common nouns will be classes
- Usually proper nouns will be objects
- Attributes are properties (state)
- Verbs will be methods (behavior)

# Evolution of C++

- The programming language C++ evolved from C and was designed by Bjarne Stroustrup at Bell Laboratories in the early 1980s.
- Though several C++ compilers were available, C++ programs were not always portable from one compiler to another.
- In the early 1990s, a joint committee of the American National Standard Institution (ANSI) and International Standard Organization (ISO) was established to standardize the syntax of C++.
- In 1999, ANSI/ISO C++ language standards were approved. (last revision happened in 2011)

# Processing a C++ program