Object Oriented Programs: Basic Concepts

Nick West
Drawbacks of “Structured” Design

- Working in Solution Domain

This is a Computer, but more natural to describe problem in terms of Objects

- Process Dominated
  - But Program = Software and Data!
  - Data structuring takes second place.

- Reusability Not Encouraged
  - But “The best way not to create bugs is not to write code!”
The World is Full of Objects!

• Consider:-
  – Power Supply:-
    • Just a source of 240V AC.
    • Not a complex of circuits, grids, transformers, power stations…
  – Telephone:-
    • Just connects to a remote phone.
    • Not a complex of circuits, exchanges, land lines, satellite links…
  – Simple Model (Data Abstraction)
  – Hidden Complexity (Data Hiding).

• PC Hardware
  • Bus
  • Memory
  • CPU
  • Screen
  • Disk Drive

All have Internal Complexity.

But relatively simple Interface.

Allows - Upgrading existing PC
- Building new PC.

Could this be done in Software?
Object Oriented Paradigm

- **Object:**
  - A miniture Program consisting of:-
    - Set of internal variables.
    - Set of functions to manipulate them.

- **Externally:-**
  - Only access functions.
  - Variables hidden; object controls access.

- **Immediate Advantages**
  - More Natural - to state problem in terms of objects. Also good for User Interface.
  - Data Hiding
    - As with SA/SD only specify function; implementation can change.
    - Unlike SA/SD data can change, leads naturally to data abstraction.
  - Locality or Data Encapsulation ensures code robust; can rigorously test all states and flow paths.
Classes and Inheritance

- **CLASSES**
  Sets of objects with similar properties form a class. Can create an arbitrary number of objects (called instances of class).

- **INHERITANCE**
  Can create a new class (subclass) developed from another (base class) by adding more variables and functions.

**Example: Window Manager**

Subclass
(application specific)

Base class
(for basics e.g. move, resize, iconise)