Abstract Data Types
&
Bottom Up Design

Bottom Up Design

- Defining the objects (abstract data types) to be used in a program.
  - » **Defining the logical data structures and operations on objects which are anticipated to be required for some higher level algorithm.**

- Difficulty and art occur when you need to decide what
  - » **finding the relevant object types**
  - » **deciding on what operations the objects should have**

Bottom Up Design – 2

- Other issues
  - » **How to describe the object types**
  - » **How to describe the relations and communications between object types**
  - » **How to use object types to structure software**

Object Motto

Ask not first what the system does –
Ask what it does it to

On OO Software Construction

The software development method which bases the architecture of any software system on modules deduced from the types of the objects it manipulates (rather than the function or functions that the system is intended to ensure)

Historical Note

Languages have few builtin data types because in the early days it was thought there were few data types and relatively many functions.

Now we see that there are many data types and relatively few functions

Abstract Data Types

- Logical constructs which model, or represent, portions of the real world.
  - » **What are they?**
  - » **Of what use are they?**
  - » **What do you do with them?**
  - » **How are they represented in programs?**
  - » **How are they documented?**

Abstract Data Types – 2

- A major advantage is to partition the solution to a problem into independent parts or modules.
  - » **Each module can be used in different contexts with little or no modification**
    - > **Able to construct larger programs and more programs with fewer resources.**
  - » **More confident that the programs are robust, and fulfil your expectations and specifications.**
  - » **Each module can evolve independently as long as the interfaces remain unchanged.**

Example – 1

- An abstract data type for a motor
  - » **Objects**
    - > **on/off switch**
      motor speed setting
      forward/reverse gear
  - » **Operations**
    - > **Enquiry – on-off?**
      forward-reverse?
    - > **Read – speed**
    - > **Write (Modify) – change speed**
      turn on, turn off
      change direction.

Example – 2

- An abstract data type for playing cards
  - » **Objects**
    - > **each card, the entire deck**
      a hand, a trick, a deal
  - » **Operations**
    - > **Enquiry – full deck? number of cards in a hand?**
      number of tricks played?
    - > **Read – the cards in a hand, the high card in a trick**
    - > **Write – deal a hand, play a card**
    - > **Reorganize – shuffle deck, sort the card**

ADT definition

- A model of a set of objects together with a set of operations on them
  - » **ADT ::= < objects , operations >**
- Abstract ==> model subset of all possible properties
- Objects are nouns, operations are verbs
    - > **Only a first approximation but useful at the boundary with the world**
- Any collection of nouns and verbs is an ADT
  - » **Difficulty is to define good and useful ADTs**

ADTs & Abstraction

- Bank deals with customers and their accounts
- Bank does not care
  - » **How the customer arrived at the bank**
  - » **What the customer ate for breakfast**
  - » **What the customer is wearing**
- Abstractly, a customer is an object that sends messages deposit, withdraw and "what is the account balance?"

ADTs & Abstraction – 2

- An ADT is designed for users who require convenient and useful set of operations on objects that are complex

- The definition of an ADT is a specification of the properties that govern the abstract objects

On Objects

- Partition the objects into two sets
  - > **Guides the development of robust and useful programs**
- Set 1 – The Real Objects
  - » **Represent real world objects**
    - **– user data, the data in which a user is interested**
    - > **a car**
    - > **a bank account**
    - > **a data file**

On Objects – 2

- Set 2 – The Meta Objects
  - » **Descriptions of other objects (from meta meaning about)**
    – programmer is interested in them to build good models with finite resources such as memory and disk
    - > **sizes**
    - > **counts**
    - > **number of attributes**
- Representation
  - » **Data structures within a program**

On Operations

- Define all the primitives to manipulate the ADT objects
  - > **primitive as no finer grain operations available**
  - > **Users build more complex secondary operations as combinations of primitives**
- Partition into 5 groups
  - > **enquiry**
  - > **read**
  - > **write**
  - > **reorganize**
  - > **test**

On Operations – 2

- Objective
  - > **Design a complete, orthogonal set of operations**
  - > **User has a simple, complete control of objects**
  - > **Minimize side effects among operations**
- Sometimes provide a larger set than strictly necessary
  - > **Increase the efficiency of combinations of operations**
  - > **Simplify user manipulation of objects**
- Representation
  - > **Methods – functions & procedures**

Enquiry Operations

- Return status information about the objects – **meta data**
  - » **Nothing changes within the ADT**
    - > **How much data do we have?**
    - > **What is the maximum number we can store?**
    - > **How many records have been processed?**
    - > **Is the structure full?**
    - > **Is the structure empty?**

Read Operations

- Retrieve data values from the data structure – **user data**
  - » **User data does not change**
  - » **Meta data could change as a side effect**
- Example – 1
  - » **Read the current record**
    - > **Meta data does not change**

Read Operations – 2

- Example – 2
  - » **Read the next record**
    - > **Case 1: there is a next record**
      - **– meta data changes to indicate either**
        (a) one more record has been read, or
        (b) there is one less record to read
    - > **Case 2: there is no next record**
      - **– meta data "end of file" is returned, the status of the file does not change**

Write Operations

- Modify the data representing the real objects – may also modify meta objects
- Typical operations are
  - » **insert, delete, replace**
- Examples
  - » **Replace a record in a file**
    - > **user data changes**
    - > **meta data remains the same**
  - » **Append a record to a file**
    - > **user data changes**
    - > **meta data changes – size of file, record count**

Reorganization Operations

- Changes the physical relationships among real objects
  - » **sort a list of names**
  - » **shuffle a deck of cards**
- Make the other operations more efficient
  - » **sorting data**
  - » **balancing a tree**
- Reorganization operations may change meta objects
  - » **balancing a tree may change the height**

Test Operations

- Not strictly a part of an ADT – rarely required by users
- Useful for implementers and regression testing
- Use operations in the other groups to test and provide diagnostic information
- Can be used to show
  - » **the meaning of the other operations**
  - » **example operator use**