

Abstract Data Type (ADT)

- Why ADT?
- What is ADT?
- ADT as objects

65

Why ADT?

- Information hiding
 - Localize changes: changes in one component does not affect other components
- Specifying the representations
 - Conceptual consolution
- Conquering complexity
 - Decompose problems into collections of interacting components

66

Abstract Data Types

- Late 1960's: Good programmers shared an intuition:
 - If you get the data structures right, the rest of the program is much simpler.
- Early 1970's:
 - **Structure**: representation bundled with operators
 - **Specifications**: abstract models, algebraic axioms
 - **Language**: modules, scope, user-defined types
 - **Integrity constraints**: invariants of data structures
 - **Rules for combining types**: declarations
 - **Information hiding**: protect properties not in specifications
- Routine practice now part of OO discipline

67

What is ADT?

- Model system as a collection of ADTs
 - Algebra specification (e.g. Larch)
 - “Larch: Language and Tools for Formal Specification”, J.V. Guttag, J.J. Horning, S.J. Garland, K.D. Jones, A. Modet & J.M. Wing, Springer-Verlag, 1993
 - Axiomatic (model-theoretic) specification (e.g. Z)
 - “The Z Notation, A Reference Manual”, J.M. Spivey, Prentice Hall, 1992

68

What is ADT?

- Each ADT includes:
 - Data objects
 - Operations on data objects (specified as functions with domain and range)
 - Essential properties of operations (specified as algebraic equations in FOL with equality)
- Two-tiered approach to software development
 - LCL: Larch Common Language (PL-independent notation for writing interface specification)
 - LSL: Larch Shared Language (PL-dependent notation for writing interface specification) LSL exists for SmallTalk, Module-3, C, C++, CLU, ...
- Large library of predefined specs for common data types
- Larch theorem prover for correctness of properties

69

An Example

```
stack(S, E): trait      /* A trait describes an ADT, I.e., data objects,
introduces             operations, properties of op & links to other ADTs */
  new: -> S             /* S is a stack, E is an element (e.g. int, str, bool) */
  push: S, E -> S
  top: S -> E exempting top(new)
  pop: s -> S exempting pop(new)
  isEmpty: S -> Bool
  asserts
    S generated by new, push
    forall stk: S, e: E
      top(push(stk, e)) == e
      pop(push(stk, e)) == stk
      isEmpty(new)
      ~ isEmpty(push(stk, e))
  implies
    linearContainer(push for insert, top for first, pop for rest)
```

70

ADTs as Objects

- Objects (also called managers) are responsible for
 - Preserving the integrity of their resources
 - Hiding the representations from other objects
 - Interacting through function and procedure invocations
- Encourage reuse (encapsulation)
- Interaction via procedure call
 - An object (client) should know the identity of another object (server)
 - Change of an object identity necessitates modification of all other objects that explicitly invoke it