

Terminology

CIS*2520 Summer 2006

Generality, Specification

- Generality
 - the extent to which a device can be applied to many similar tasks rather than a few specific tasks
- Specification
 - a detailed description of the essential properties of an object
 - e.g. ADT:
 - a definition/description of the data type together with a set of operations defined on it
 - e.g. programming:
 - a complete description of a program/module's required behaviour

CIS*2520 - Summer 2006

D. McCaughan

Implementation, Interface

- Implementation
 - an object built to satisfy a specification
 - e.g. programming:
 - a module written to behave as specified
- Interface
 - the way in which an object interacts with its environment
 - e.g. programming:
 - a description of how a programming module is invoked from its external program environment (e.g. parameter passing conventions) along with a list of any dependencies that exist between the two (e.g. external variables used by the module)

CIS*2520 - Summer 2006

D. McCaughan

Pre/Post-Conditions

- Preconditions of a subprogram
 - a set of conditions that must be true before a sub-program is invoked
 - if one or more of the preconditions fail when a subprogram is invoked, it is not guaranteed to perform its intended task
- Postconditions of a subprogram
 - a set of conditions that will be true when a subprogram terminates provided that the subprogram preconditions were true when the subprogram is invoked

CIS*2520 - Summer 2006

D. McCaughan

Decomposition, Module

- Decomposition
 - a separation into discernable parts, each of which is simpler than the whole
- Module
 - a separately compiled program unit that can be used to collect related constants, types, variables and subprograms
 - *public* module elements can be accessed freely
 - *private* module elements can only be accessed by the source module

CIS*2520 - Summer 2006

D. McCaughan

Decomposition (cont.)

- Functional Decomposition
 - breaking up overall program *behaviour* (functions) into subprograms simplifying higher level algorithmic description
- Modular Decomposition
 - breaking up program structure into modules
 - each has a well-defined scope of operation (thus simpler)
 - program can be expressed in terms of use of and interactions between modules
- Data Decomposition
 - breaking up data structures into components, each of which is simpler and relatively independent of the aggregate data structure

CIS*2520 - Summer 2006

D. McCaughan

Information Hiding

- Information Hiding (encapsulation)
 - embedding the details of a program element inside the element in such a way that when it is applied, its inner details do not have to be known
 - ideally, the inner details are so well hidden that not only are they not needed, they are not even available (usually thought of as “invisible” to the outside)

CIS*2520 - Summer 2006

D. McCaughan

Black Box Model

- Black Box Model
 - viewing an object in terms of its interface, ignoring all internal details, pretending the internal environment is completely hidden inside a black box
 - e.g. programming
 - viewing a program unit in terms of its interface while ignoring its inner details (e.g. local variables and algorithms)

CIS*2520 - Summer 2006

D. McCaughan

Data Values

- Atomic Data Value
 - data considered as a simple non-decomposable entity (as we choose to see them)
 - e.g. the integer 7
 - if we choose to consider representation of an integer as a collection of 32 bits, then "7" is not atomic
- Data Type
 - identifies characteristics common to a group of objects
 - components:
 - 1) set of values (e.g the set of all integers, etc.)
 - 2) set of operations defined on these values (+, -, *, etc.)

CIS*2520 - Summer 2006

D. McCaughan

Data Types

- Atomic Data Type
 - represents data considered as a single non-decomposable entity
 - e.g. integer data type
 - values = {..., -2, -1, 0, 1, 2, ...}
 - operations = {+, -, *, /, ... }
- Structured Data Type
 - values can be decomposed
 - there exist rules for associating elements
 - e.g. `int example[10][10]`
 - we can decompose the array into individual int components
 - we know, for example, that the elements of the array occur in row-major order (i.e. `example[2][1]` occurs before `example[3][1]`)

CIS*2520 - Summer 2006

D. McCaughan

Abstract Data Types

- Abstraction
 - reasoning about the essential properties of an object, ignoring unimportant details
- Abstract Data Type
 - specifies a data type (atomic or structured) and specifies a set of operations defined on this type *ignoring implementation details*
 - specified using data structures typically available in high level languages (C, Java, etc.) and algorithmic or mathematical notation
 - NOTE: this definition is just for reference
 - we will return to ADTs at length very shortly

CIS*2520 - Summer 2006

D. McCaughan