

System Contracts

- ❖ Define responsibilities within the system
- ❖ Operational contracts take the form of *pre-conditions* and *post-conditions*
 - Each module tests pre-conditions, then runs, then tests post-conditions before returning to the caller
 - ◆ C/C++'s `<assert.h>`
 - ◆ Proof by method of “inductive assertions”

1

System Contracts

- ❖ Operational contracts state what an operation commits to achieving.
- ❖ Emphasizes *what* will happen, not how
- ❖ Describes changes in the state of the system
 - Semantics

2

How to Make a Contract

- ❖ Identify operations from the system sequence diagram
- ❖ For each system operation, construct a contract
- ❖ Start by writing the *Responsibilities* section, informally describing the purpose of the operation
- ❖ Then complete the *Post-conditions* section, describing state changes that occur to objects in the Conceptual Model
- ❖ To describe post-conditions, use the following categories:
 - Instance creation and deletion
 - Attribute modification
 - Associations formed and broken

p. 152
3

POST Contract Process

4

Advice on Writing Contracts

- ❖ Fill in the Responsibilities section first
 - Post-conditions section next
 - Pre-conditions section last!
- ❖ Use past tense in stating post-conditions
- ❖ Use Notes to discuss any algorithm selection or details pertinent to design
- ❖ Use Exceptions section for truly exceptional events

5

Advice on Writing Contracts

- ❖ Use these categories of state changes in post-conditions
 - Instance creation and deletion
 - Attribute modification
 - Associations formed and broken
- ❖ Remember to establish a “memory” between existing objects by defining the *forming of an association*

6

