

Writing Bug-Free Software

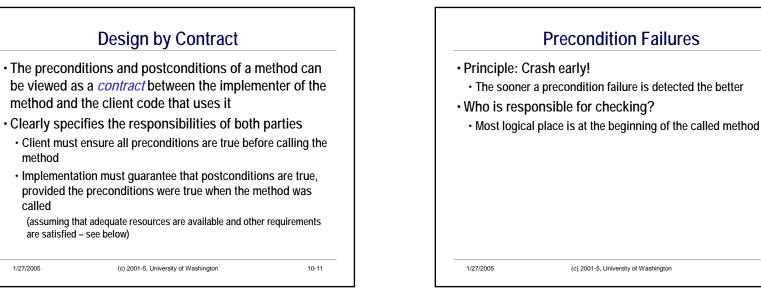
- Preconditions, postconditions, and invariants are incredibly useful
- Include all non-trivial ones as comments in the code
 - These are essential parts of the design and a reader must understand them to understand the code
 - If you don't write them down, the reader (who may be you) will have to reconstruct them as best he/she can
- Whenever you update a variable, check any invariants that refer to it to be sure the invariant still holds
- May need to update related variables to make this happen

1/27/2005

(c) 2001-5, University of Washington

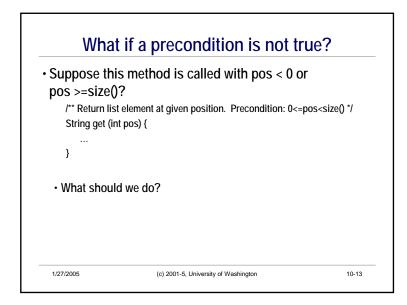
10-10

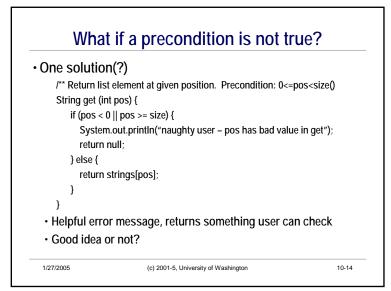
10-12

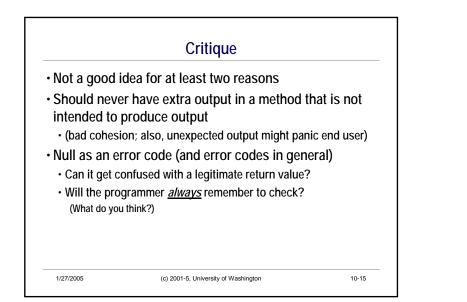


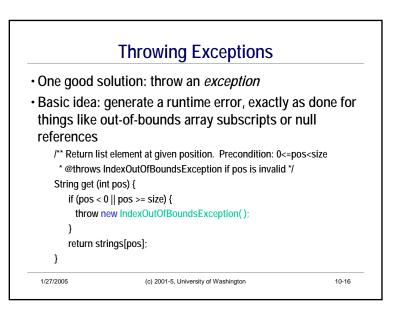
- method and the client code that uses it Clearly specifies the responsibilities of both parties

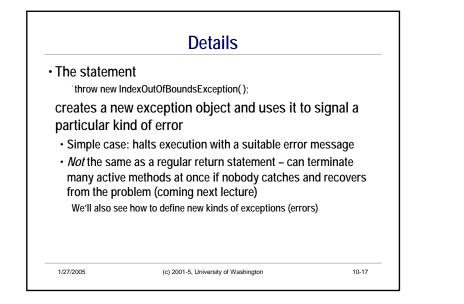
CSE143 Wi05

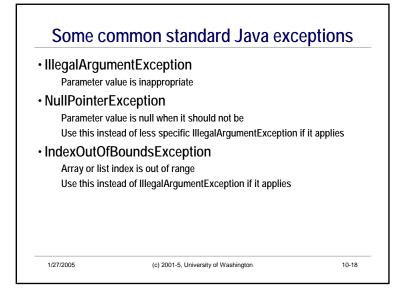


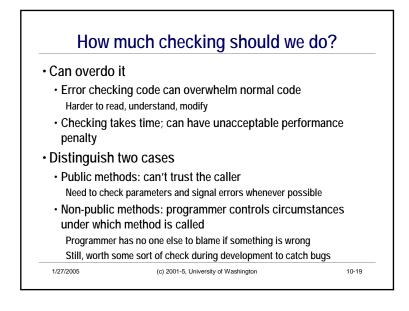


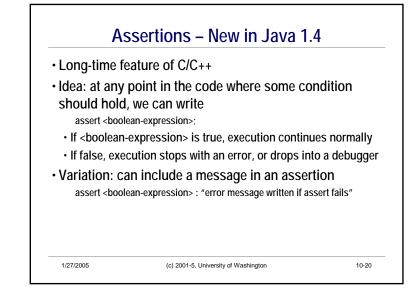


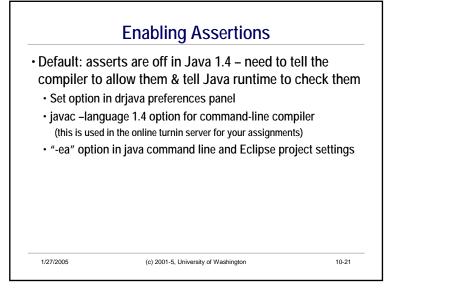


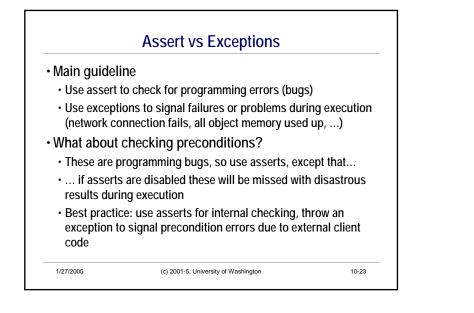












<section-header><list-item><list-item><list-item><list-item><list-item><list-item><list-item><list-item><list-item>

	Summary	
 Use assertion handling 	ons and exceptions for discipline	d error
Assert to cather the outside	atch bugs in your code; exceptions for world	r dealing with
	nciple: it is <i>much</i> better to fail ear execution in a buggy state	ly instead of
•	action: exception handling – read rom problems	cting to and
• Then: on to	streams and files	
1/27/2005	(c) 2001-5, University of Washington	10-24