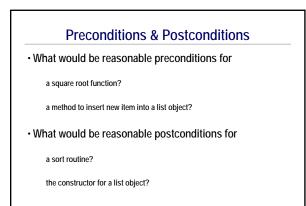


(c) 2001-5, University of Washington

10-7

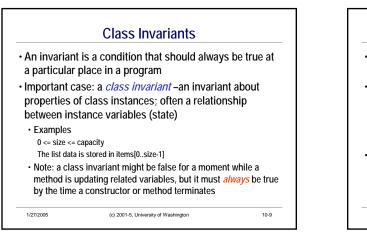
1/27/2005

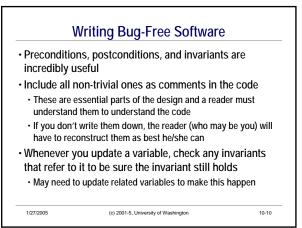


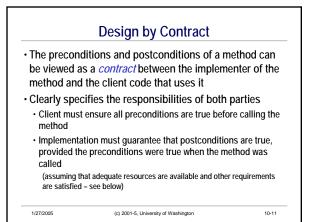
(c) 2001-5, University of Washington

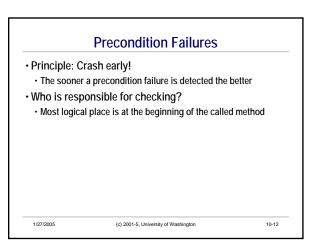
10-8

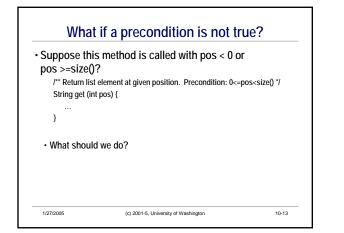
1/27/2005

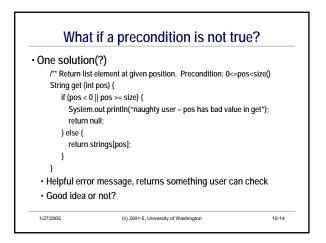


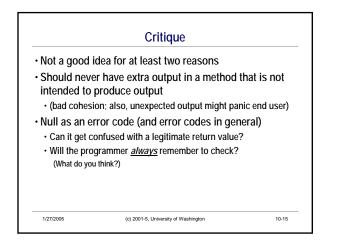


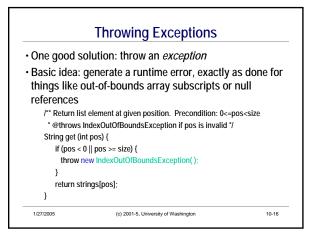


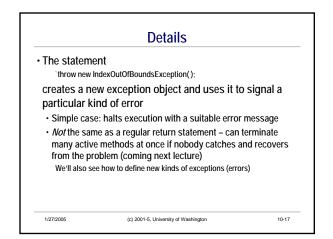


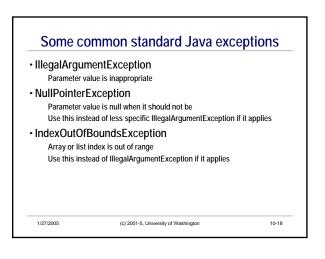












How much checking should we do?

· Can overdo it

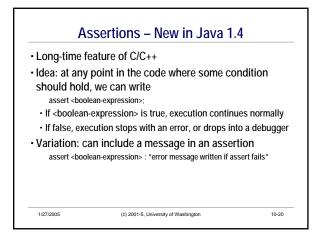
1/27/2005

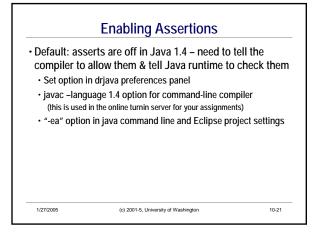
- Error checking code can overwhelm normal code Harder to read, understand, modify
- Checking takes time; can have unacceptable performance penalty

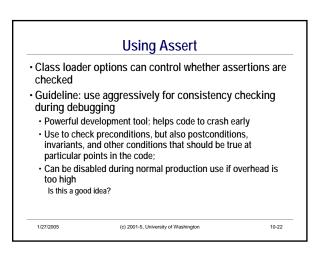
Distinguish two cases

- Public methods: can't trust the caller Need to check parameters and signal errors whenever possible
 Non-public methods: programmer controls circumstances
- under which method is called Programmer has no one else to blame if something is wrong
- Still, worth some sort of check during development to catch bugs
 - (c) 2001-5, University of Washington

10-19







Assert vs Exceptions Main guideline Use assert to check for programming errors (bugs) Use exceptions to signal failures or problems during execution (network connection fails, all object memory used up, ...) Use about checking preconditions? These are programming bugs, so use asserts, except that... ... if asserts are disabled these will be missed with disastrous results during execution Best practice: use asserts for internal checking, throw an exception to signal precondition errors due to external client code

	Cummony	
	Summary	
 Use assertion handling 	ons and exceptions for disciplined	error
 Assert to catch bugs in your code; exceptions for dealing with the outside world 		
•	ciple: it is <i>much</i> better to fail early execution in a buggy state	instead of
5	action: exception handling – reacti rom problems	ing to and
Then: on to a	streams and files	
1/27/2005	(c) 2001-5, University of Washington	10-24