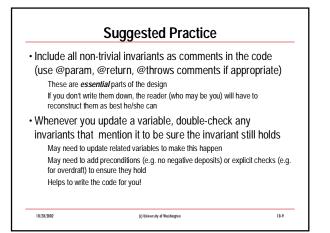
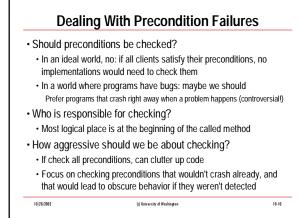


Class Invariants	Writing Bug-Free Software
 Special case: a class invariant – something that is always true for each <i>instance</i> of the class, at least as seen from the outside Class invariants express requirements on the values or relationships of instance variables It employee.jobcde 'Programmer', then employee.salary > \$50,000 0 <= this.size <= this.capacity The list data is stored in this.elements[0this.size-1] A class invariant might not hold while a method is in the middle of updating related variables, but it must <i>always</i> be true by the time a constructor or method terminates Any class invariant is automatically: A postcondition of every constructor and method of that class A precondition of every method 	 Invariants, including pre- and post- conditions, incredibly useful in design and understanding Program bugs can often be seen as unforeseed invariants being violated In principle: If you could write down all invariants, and have them the program runs, bugs would practically disappear In reality: Writing down all invariants is tedious to impossible Java gives little direct support for documenting and invariants The situation is similar in most common languages
10/28/2002 (c) University of Washington 10-7	10/28/2002 (c) University of Washington

- conditions, are erstanding
- s unforeseen cases of
- nd have them checked as Ily disappear
- menting and checking
 - jes

10-8





	a Precondition is not Tr	
 Goal: to force in 	nmediate termination	• Long-tim
 Reason: the contract has been broken 		Idea: at a
Throw a RuntimeException		can write
Since these ex	ceptions shouldn't ever be thrown, and	
	ct to handle them, they shouldn't be liste	ed in throws • If false,
clauses		Asserts of
 Not possible to handle the exception to produce some different 		
output or clean-up operation		Guideline
Write error messages to System.out or System.err?		
 Might help you 	during debug	• Use to
 Of marginal hel 	p in a production environment	Unfortun condition
Neither you nor t	he client may have access to the console windo	ow Condition
10/28/2002	(c) University of Washington	10-11 10/28/2002

Assertions – New Feature of Java 1.4	
ong-time feature of C/C++	
lea: at any point in the code where some condition should hold, we an write this type of statement:	
assert <boolean expression="">;</boolean>	
If < boolean expression> is true, execution continues normally	
If false, execution stops with an error, or drops into a debugger,	

- Asserts can be disabled without removing them from the source code
 Means there is no performance penalty for production code
- Guideline: use aggressively for consistency checking
 - Powerful development tool; helps code to crash early
 - Use to check all types of invariants, not just preconditions
- Unfortunately, not all invariants can be expressed by simple Boolean conditions.

(c) University of Washington

10-3

10-12

Invariants and Inheritance

• When methods are overriden

10/28/2002

- preconditions can be weakened in overriding methods
- postconditions can be strengthened
- Class invariants can be strengthened (since the class itself is ensuring they are respected), or even changed arbitrarily, as long as inherited methods still have proper preconditions met when they're called and inheriting code only assumes inherited postconditions are true

(c) University of Washington

10-13