CSE 331 SOFTWARE DESIGN & IMPLEMENTATION MIDTERM REVIEW

Autumn 2011

The other kind of testing...

- Actually, it's the same as software testing (mostly)
- By picking effective subdomains, I hope to determine how likely it is that you understand the material – it's inherently sampling, not proof
- In this situation, a single test suite will be executed across 56 different processors

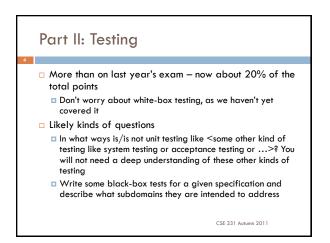
CSE 331 Autumn 2011

The form of the test: Subject to some (but limited) change

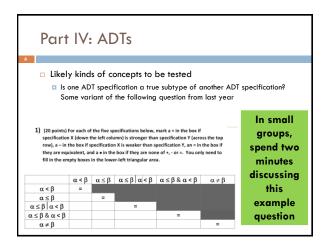
- Part I: True/false with a brief justification
 5-10 questions
- □ Two examples from last year
 - "hashCode can be determined at most once that is, only when it is first actually requested by a client and then it can be cached."
 - "If an immutable object throws an exception, it is never left in an undesirable or indeterminate state."

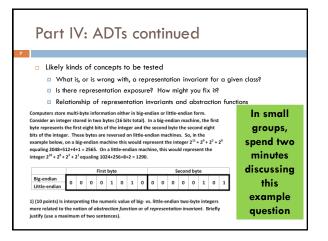
In small groups, spend two minutes discussing these

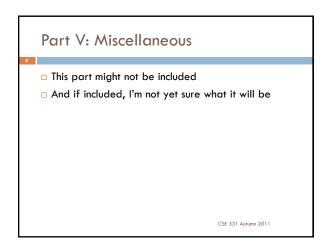
CSE 331 Autumn 2011



<section-header> Orar III: Specifications Oracle Antipication of the second sec



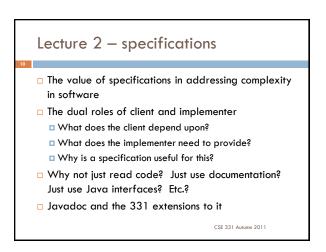




Per lecture: points to focus upon But others are fair game still Lecture 1 – introduction Programs (implementation) satisfying specifications It's tricky business It's a many-to-many mapping No notion of a "correct" specification

Some can surely admit implementations that are highly unlikely to be desired

CSE 331 Autumn 2011



Lecture 3 - testing

- $\hfill\square$ Testing is one form of quality assurance for software
- Testing terminology pass, fail, test case, test suite, ...
- □ General notion of kinds of testing
- Subdomains
- □ JUnit's role what can it help you do and not do?

CSE 331 Autumn 2011

Lecture 4 – equality

- Different notions of equality
- □ Key underlying properties of (any useful) equality
- Relationship of equals and hashCode
- Overriding vs. overloading

CSE 331 Autumn 2011

Lectures 5 and 6 – ADTs

- Motivations for the use of ADTs
- Primary focus on ADT operations rather than representations
 Different kinds of ADT operations (observers, mutators, etc.) and differences between mutable and immutable ADTs
 - Hide the implementation decisions to allow change
- Abstraction function what is it, why is it important, how is it used?
- Representation invariant what is it, why is it important, how is it used?
- The relationship between the AF and RI, the ADT and its implementation (that diagram)
- Representation exposure what is it, how to eliminate it?

CSE 331 Autumn 2011

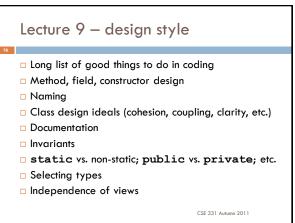
Lecture 7 – subtyping & subclassing

- A way to share behaviors and/or code
- Weaker and stronger specifications and the relationship to satisfying implementations
- True subtyping vs. Java subtyping allowing substitutability
- Subtyping is over specifications; subclassing is over implementations – both use similar mechanisms in Java
- Mutability can be useful, but can confuse the issue of true subtyping

CSE 331 Autumn 2011

Lecture 8 – modular design principles Cohesion (why together?) and coupling (how do modules interact?) Different kinds of dependences – invokes, names, extends, etc. Ways to manage dependences – e.g., Law of Demeter Module dependence diagrams (largely to identify coupling) Selection

CSE 331 Autumn 2011



Lectures 10-12

- Design patterns, basic GUI
- Not a focus of this test will be fair game on the final



