Abstraction Functions and Rep Invariants

The Most Important Slide

- Refer to Ernst's handout: http://www.cs.washington.edu/education/courses/cse331/10sp/conceptual-info/abstraction-functions-and-rep-invariants.html

These slides are basically a subset of what's there

Abstract Data Types

- You're given an abstract data type (ADT)
  - Describes a high-level object / data structure
  - Doesn't describe how it's implemented
- You're writing one concrete implementation
- Implementation tied to ADT but not vice versa
- Example: Dictionary/Map
  - ADT: stores and retrieves key-value pairs
  - Implementation: BinaryTree, HashTable, …

Abstraction Function

- Need to demonstrate that your class satisfies all specifications of ADT
- Abstraction function maps from concrete representation to abstract value
- Shows how an instance of your class represents an instance of the ADT

Step 1: Describe the ADT

- See Ernst's handout: Line example
  1. Explain what the ADT represents
  2. List and explain all the ADT's fields
- May not be necessary
  - See Ernst's handout: Complex, Cons

Step 2: Describe the AF

1. In Javadoc: state what ADT you're implementing
2. In non-Javadoc: describe an instance of the ADT in terms of your class's instance fields

- Formally: AF(r) = <description>

  where r is an object of your class

  → refer to r's fields

- Informally: can drop the "AF(r)" - see Ernst's handout for examples

Step 3: Describe the RI

- Representation invariant: maps concrete representation to a boolean
- If true, object is well-formed
  - AF guaranteed to hold
  - Object guaranteed to behave correctly
- If false, object is broken
  - AF may not hold
  - Behavior undefined
- See Ernst's handout