

Lecture Set #8: Debugging

1. Complete Class Summary
2. The Eclipse Debugger



CMSC 131 Spring 2008
Jan Plane

Putting the pieces together

- Constructors
 - default constructor
 - constructors with parameters
 - copy constructors
- Data
 - data members: instance/static and public/private
 - local variables
 - stack and heap
 - null references
- Methods
 - instance/static and public/private
 - overloading: toString and others
- Libraries
 - importing and using methods from the library (the API)
- JUnit Testing
- Exceptions
 - Throwing, trying, catching



CMSC 131 Spring 2008
Jan Plane

1



The problem

- Problem
 - JUnit can only tell if that passes or fails and where
 - Need a way to be able to see what is in memory (variables) at every step to be able to do complete trace [like that call stack examples we have been doing]
- Solution
 - The debugger gives the ability to go through the code – displaying additional information similar to the by-hand call stack that we have been doing



Terminology

- Break Point
 - drop a marker into the code so when it runs the execution will stop at that point
 - allows you to not have to go step by step through things you believe are correct
- Step Over
 - takes one step in the current method
 - if that step is a method call, it performs that whole method call and steps to the next line in the current method
- Step Into
 - takes one step in the current method
 - if that step is a method call, it steps into that method so that you can then step through it before getting to the next line in the method you were in



Eclipse

- Perspective
 - Debug Perspective
 - Java Perspective
- Run
 - Debug As...
 - Run As...
- Know if it is still running
 - Watch the red square – click it to kill



Corner Cases

- Those that fit between
- or are different than the normal
- examples:
 - really long
 - empty string
 - single character word