CMSC 131

Object-Oriented Programming I

Testing/Debugging

Dept of Computer Science
University of Maryland College Park

This material is based on material provided by Ben Bederson, Bonnie Dorr, Fawzi Emad, David Mount, Jan Plane

Overview

- Class Summary
- Debugging
- Eclipse Debugger

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
  - Overriding: toString
  - Overloading: constructors
- Libraries
  - Importing and using methods from the library (the API)
- JUnit Testing
- Exceptions
  - Throwing, trying, catching

Debugging

- Process of finding and fixing software errors
  - After testing detects error
- Goal
  - Determine cause of run-time & logic errors
  - Correct errors (without introducing new errors)
- Similar to detective work
  - Carefully inspect information in program
    - Code
    - Values of variables
    - Program behavior
- It is a skill you need to develop
  - After all, there is no such thing as the “Company TA” ☺

Debugging Approaches

- Classic
  - Insert debugging statements
  - Trace program control flow
  - Display value of variables
- Modern
  - IDE (integrated development environment)
  - Interactive debugger

Interactive Debugger

- Capabilities
  - Provides trace of program execution
  - Shows location in code where error encountered
- Interactive program execution
  - Single step through code
  - Run to breakpoints
- Displays values of variables
  - For current state of program

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

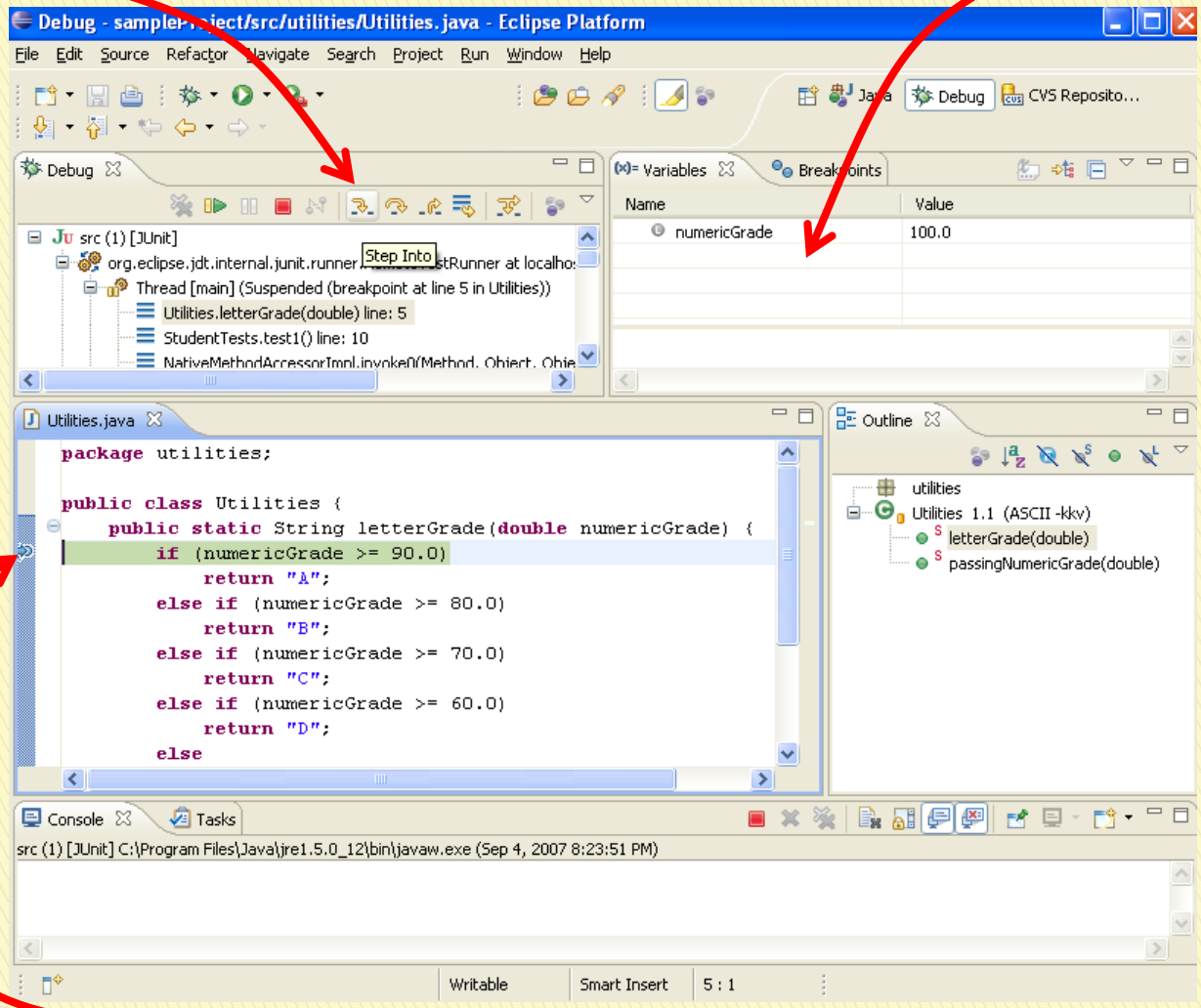
Eclipse

- Perspective
  - Debug Perspective
  - Java Perspective
- Run
  - Debug As…
  - Run As…
- Know if it is still running
  - Watch the red square – click it to kill
- Eclipse debugging information available at:
  - http://www.cs.umd.edu/eclipse/EclipseTutorial/debugging.html

Eclipse Debugger

Single Step

Data Display



Breakpoint

Features to Explore

- Right-click on left side of code pane allows you to set numbers
- Setting breakpoints
  - If you right-click on Breakpoints pane you will see additional breakpoint options
- Stepping through
  - Green right arrow (with yellow rectangle to the left) enable us to resume execution
- Stack
  - How to change to different stack entries
- Looking at parameters/local variables/instance data through the debugger
- Color scheme used to identify private (red), protected (yellow), public (green)

Features to Explore

- Looking at constants and other class members
  - To see constants and other members use the down arrow on the pane where Variables/Breakpoints appear. Right-click on the arrow and select Java.
- How when selecting an object through the debugger we see the result of calling toString() on the object
- When an exception occurs
  - Notice how execution is suspended
- Outline Pane
  - Allows you to select a method
- Debugging and JUnit tests
  - We can also run JUnit tests through the debugger

Testing

- Make sure all your methods are tested
  - Code Coverage → Submit server can provide information about % of your code tested by submitted tests
- Corner cases → while testing/debugging consider corner cases
  - Those that fit between or are different than the normal
  - Examples:
    - Really long
    - Empty string
    - Single character word
    - null