Introduction to testing, erros and debugging

* Compiler time rrors
* Link time erros
* Run time error
* Logic errors

Compiler errors

Options for error recording

* System.err
* Exit with a non zero error code

I System.err.println(“Invalid Length”)

System.exit(return 0)

Exeptions

* + An object created to represent an error or other unusual occurrence and then propagated via special mechanisms until caught by special handling code.
* You can(and often shoukld)use a pre-definde exception
  + RuntimeException(“Bad Dates”)
  + IllegalArguementException(Nextative size)
* Throw an exception when an error occurs
  + Many library methods already throw exceptions
* Define a scope in which to watch for an exception(try{})
  + The code inside the bracese is monitored for exceptions
* Immediately after the try scope define one or more exception
  + (catch(Exceptions))
* You may catch multuiple types of exceptions with one try

Basics Test Options

* Interactive testing
  + Run the program like a nightmare user- try to break it
  + Some testing is likely to require a written test procedure
  + Labor intensive, so avoid this as much as possible
* Regression testing
  + Write a test program
  + Systematically try expected inputs for correct operation
  + Try forbidden inputs for proper exception handling
  + When you find a bug, write a new test to ensudre it doesn’t pop up again
* Test-driven development
  + Write tests for requirements first, then prove the tests fail
  + Repeat for each erequirement
  + Refactor to create a clean and maintainable solution

Debugger

* Option one
  + Mental Execution
    - Pretend that you are the computer running your program
    - Does it output match your expectations
* Option 2
  + Add output
    - Send intermediate data to out
    - Or use a logging framework
* Option 3
  + Invoke the debugger
    - Rebuild your program with additional hooks
    - Run it inside a program specifically designed to help
      * Examine variables while it runs
      * Stop when something interesting happens
  + Execute and stop at specific statement
  + Single step through program statements
  + List all methods calls as they occure
  + Examine attributes and local variables
  + Each IDE has its own debugger
  + WE will ocver jdb, the official command line debugger
    - But learn and use your ides debugger
  + Commands
    - Stop
    - Run
    - Clear
    - Ignore
    - Catch
    - Step
      * Step
      * Step up
      * Stepi
      * Next
      * cont
    - Locals