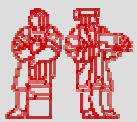
Coding Methodology

How to Design Code

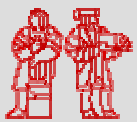
Pay Attention to Detail

- When implementing or using APIs details are everything.
- Spelling and capitalization.
- Names.
- Argument types.
- Return type.



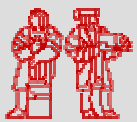
Create a Skeleton

- Type in method signatures with empty bodies:
  - public static void foo() { }
- For methods with primitive return types, declare a dummy variable or return 0:
  - private int bar(int x) { int i; return i; }
  - double deuce() { return 0.0; }
- For Object return types, return null:
  - public String toString() { return null; }



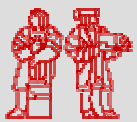
Write Test Code

- Write test code that makes calls to your skeleton.
- You'll expect null or zero values and can't call anything on the returned objects.
- Start out with really basic tests, like instantiating an object.
- Add new tests as you fill in your skeleton.



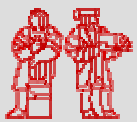
Types of Bugs

- Compile time bugs: typos and syntax.
- Logic or control bugs: Correct syntax, but incorrect design. Compiles, but code does not work as expected.
- Runtime bugs: Bugs that arise from data provided at runtime.
  - Bad input, divide by zero, null pointers.
  - Can be handled with Exceptions.
  - Or can cause program to crash.



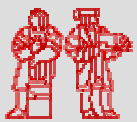
Add Debugging Output

- Put in a lot of println() statements that output values of variables for yourself.
- Can add messages like *"Entering method foo"* or *"Exiting Method NNN"*.
- Can also add debugging messages that help you trace program flow through control structures.
- Java 1.4 has java.util.logging package that helps with debugging output.



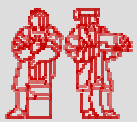
Code, Compile, Repeat

- Add some code to a skeleton method.
- Write test code to check the new code.
- Compile your code.
- Run it.
- Check for correct debugging output.
- Repeat.



Philosophies

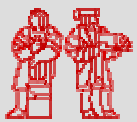
- Extreme Programming (XP):
  - Design test cases first, always test.
  - Implement incrementally.
  - Design organically (hack).
  - Expect to write the same code twice.
  - Code in pairs: Typist and shoulder-surfer.
- Old School:
  - Design everything on paper.
  - Rigid implementation plan.
  - Testing and QA is last step.



"Debug a Blank Sheet of Paper"

- Dr. Brian Harvey

UC Berkeley



EC.S01 Internet Technology in Local and Global Communities
Spring 2005-Summer 2005