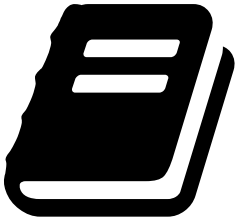
CPSC 310 – Software Engineering

Debugging

The Importance of Clean Code

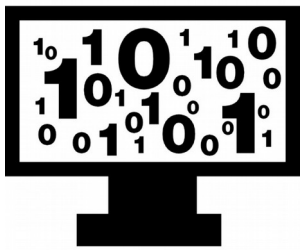
- It is easier to find the root of the problem when your code is clean
  - Good names
  - Abstraction
  - Indentation

Different expectations



What the specifications say the program should do

What you believe the program is doing



What the program actually does

Always check your assumptions

"*Finding your bug is a process of confirming the many things that you believe are true — until you find one which is not True*" N. Matloff

Debugger (direct approach)

A tool to control (pause) the execution of a program in order to observe its state (values values of the variable)

Allow you to define breakpoints to indicate where you want to stop.

Has usually the following commands:

- Step into: execute next line of code, go into method
- Step over: execute next line of code, do not go into method
- Step out: finish the execution of the current method
- Resume: go to next breakpoint
- Pause
- Stop

Where to start ?

- If you have no idea of the location of the bug you can try a binary search approach
  - Check if the program is working fine at the “halfway point”
    - If yes check at “3/4-way point”
    - If not check at “1/4-way point”
  - Every time narrow down to one half of the previous portion

Logger (indirect approach)

- Sometimes it is not possible to execute directly the code
  - eg. code running on the server in a client/server architecture
- Dedicated libraries are available depending on your software stack.
  - They make the job easier
  - Usually they define several levels of severity and have a specific one for debugging

Congratulations, you fixed a bug

Rerun the test to check if you introduced a new bug.

Depending on:

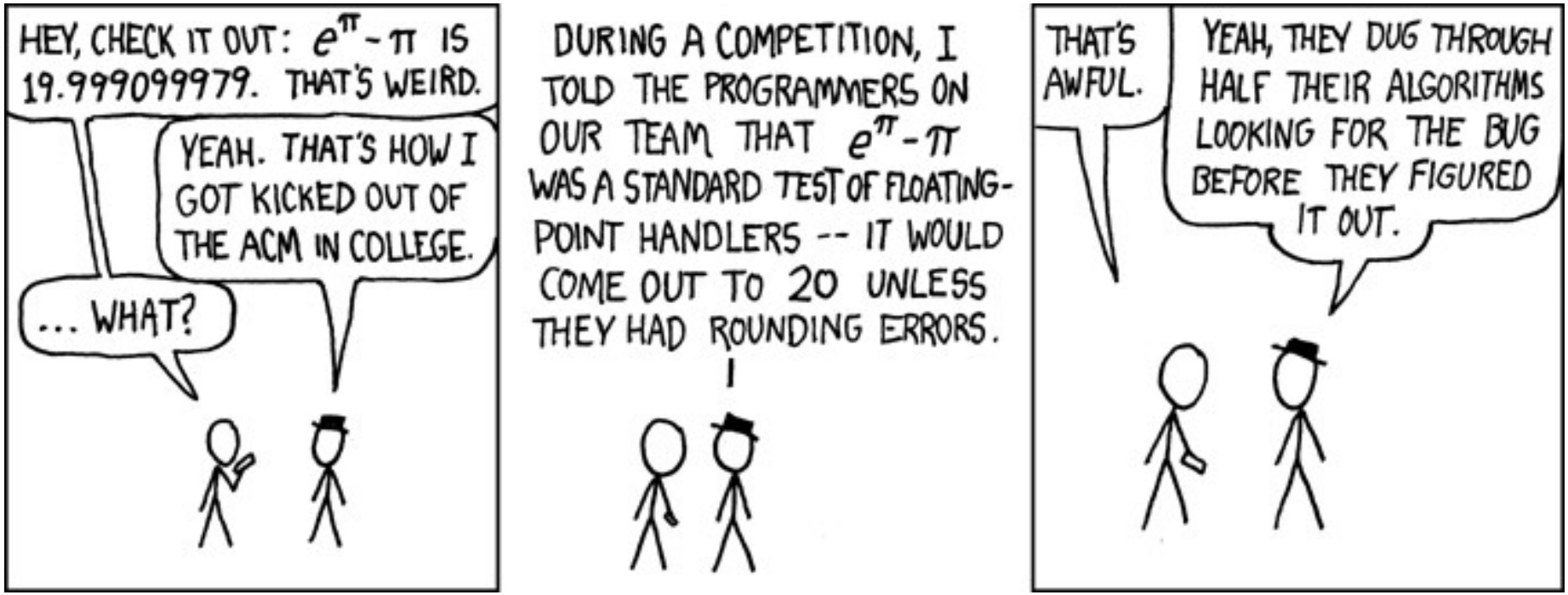
- the severity of the bug
- the likelihood that it can be reintroduced

It is a good practice to write a test that can verify that it will not reappear in the future (regression testing)

Your turn

- Launch your Eclipse installation
- Import the project available on the lecture webpage
  - HELP: http://agile.csc.ncsu.edu/SEMaterials/tutorials/import_export/
- Familiarize yourself with the Eclipse debugger
  - HELP: http://www.vogella.com/tutorials/EclipseDebugging/article.html
- Find the bugs in exercise 1 using the debugger
  - Use breakpoints

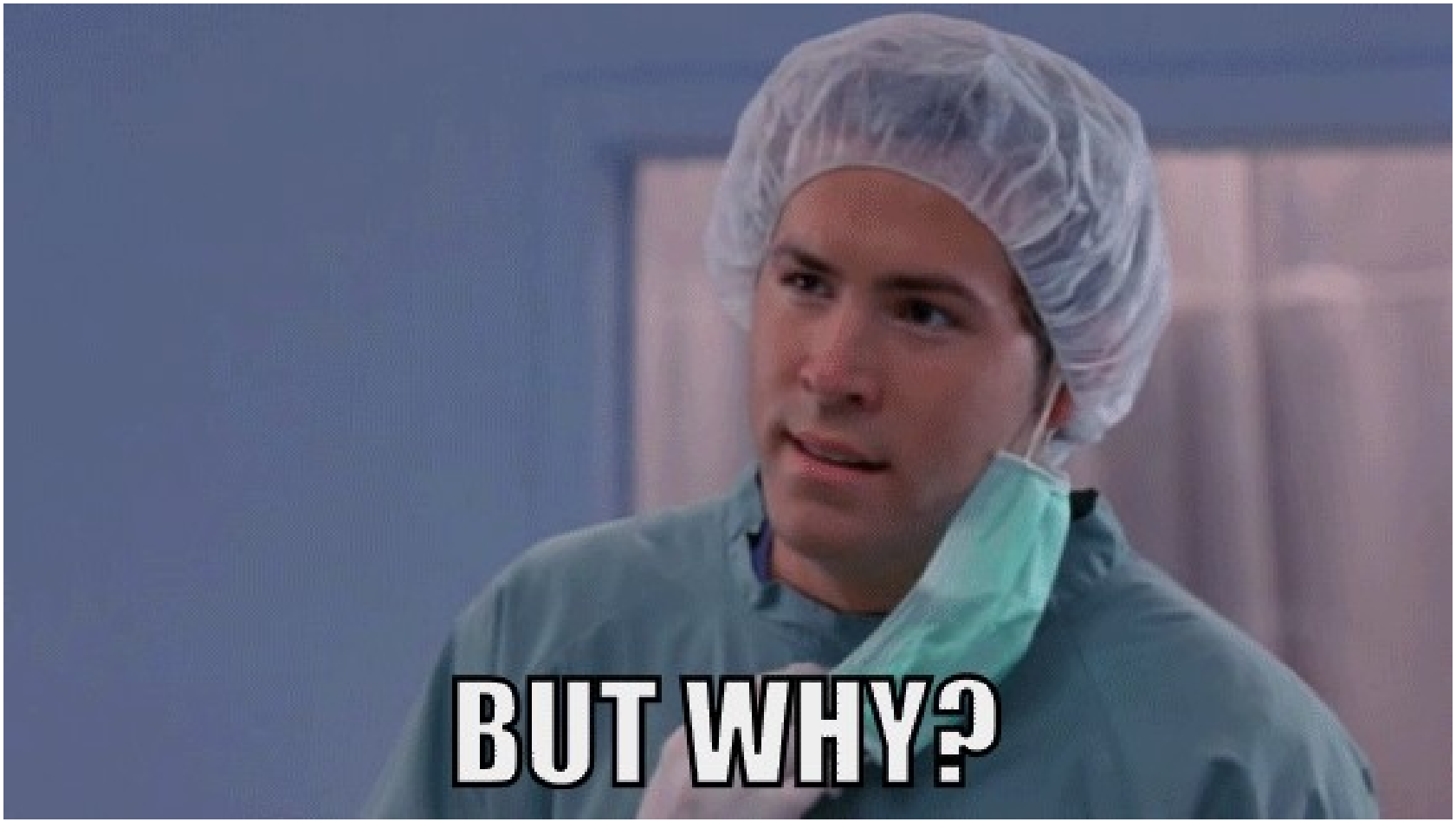
Floating-point arithmetic



http://www.xkcd.com/217/

Floating-point arithmetic

0.1+0.2 = 0.30000000000000004



Floating-point arithmetic problem

Computers use base 2

We use base 10

Problem: a round number in base 10 is not necessarily a round number in base 2

Floating-point arithmetic “solution”

- Use only integers by using the smallest unit (eg. cent)
  - It's more work/bugs since you have to redefine a lot things
  - Not flexible
- Use a decimal data type (eg BigDecimal in Java)
  - This means that the exponent of the number stored is interpreted in base 10.
  - Arbitrary-precision: mantissa/exponent size can vary