4156 11/5/15

Testing Overview

Head First ch. 7-8 & appendix i.4

start w/ appendix — short

system tests vs. unit test

what is the difference?

unit tests test code

does the code do what it should?
class & method level

build process should re-run every unit test after every commit (or nightly)
— continuous integration

system tests test full application

does it fulfill all requirements?
functional & non-functional

need to do both

4156 11/5/15

acceptance tests are customer's version of system tests

system & acceptance testing often require human tester, or simulation of human tester

unit tests can be totally automated

acceptance tests & some systems tests are totally black box from end-user's point of view

black box - focuses on input & output

- functionality
  does the system do what the user story/use case says it should do?

- user input validation
  need to check all inputs from user
  reject or sanitize

will return to later

4156

11/5/15

- output results
  even if inputs are valid, outputs might not be
  check results

- state transitions
  - if application has concept of states, are transitions correct?
  - can user jump to an inappropriate state
  - web pages often correspond to states in web application
    - url hacking

- boundary cases + off by one errors
  values a little too small or too big
  zero-based arrays

grey box - often done by professional testers who aren't developers
similar to black box but looks below surface

4156

11/5/15

- verify auditing & logging
  if its supposed to happen,
  check that it does
  ordinarily, not visible to users

- data destined for other systems
  check output format & data details

- system-added info
  checksums, hashes for checking
  secured correct storage
  hand-check values
  timestamps accurate,
  right time zone
  stored w/ right data

- scraps left lying around
  security risk & resource leak
  delete what is supposed
  to be deleted
  scraps in files, registries,
  uninstall should leave
  system clean

think about everything that
could go wrong

4156

11/5/15

special concerns for distributed systems
- client/server
- peer/peer
- multi-process on same host
- web browser
- mobile/cloud app

what could go wrong?

white box testing
- inside knowledge of code
- try to ~~break~~ break!!

- testing all the branches
  - what input needed to force a branch or a combination of branches

- proper error handling
  - does invalid data generate appropriate error (not just any error)

- is code cleaning up after itself
  - release resources - file handles, mutexes, memory

4156 11/5/15

working as documented
is method thread-safe
can ~~you~~ you pass null argument
security role needed

handle resource constraints
memory, disk, network connection
-what does code do
if can't get?
try to force situation

white box tested usually automated
whether unit or system level
code-on-code not human user

testing frameworks
run tests
they do not write tests
ensure consistency
maybe- "flaky" tests

build up a suite of tests
run all tests w/ one command
regression testing for free
regression - adding new code introduces
bugs in old code or breaks old tests

4156 11/5/15

but the longer a test suite takes to run, the less often it will be run

the book ch. 7 discusses how to use JUnit, but there are better tutorials on line

get JUnit working assignment
- yes it is due during the project demo period
(because the following assignment will run with thanksgiving)

continuous integration =
run JUnit (or other testing framework) as part of build
EVERY build

code is not finished until it compiles & all test pass
- even tests for OTHER code

code coverage - what percent of your code is actually exercised by your test cases later on this
BRANCHES not just lines