When to Write Tests – How to Structure Tests

Lecture Objectives:

1) Critique the advantages of writing tests before development has commenced.
2) Critique the advantages of having a second person write tests independent from the developer.
3) Explain three methods for organizing test cases and the advantages and disadvantages of each one.

3 approaches.

Who should write tests?

- You (as the developer)

— Not paying someone else.

— Knowing what the code should do versus what someone thinks it should do.

— You can fix your own.

Who should write tests?

- Another developer (for you)

- No bias.
- different viewpoint.
- More important things to do :-)
  (lazy...)

When to write tests

- Before coding

Test Driven Development

Clear definition of
Code working.

- Not writing tests 7, pass.
- Thinking about testing when we write our code.

When to write tests

- In parallel with implementation

- Catch bugs as they are written.
- Catch complexity as it is implemented and make sure test cases fully test the complexity.
- Easier to change tests if requirements change.
- Learning as you implement.

Additional resources for a project.

When to write tests

- After implementation

- You know what is done and working
- Everything can be tested after.
- May not need mocks / simulations.
- May have cleaner regs after implementation is done.

time crunch

Where to put the test cases

- We all agree test cases are good
  - I hope ☺

- Where should we store them?
- Doesn't matter much for small projects, but can make a huge difference on big ones…
- 3-4 general methods

Same file

- Test cases are written directly in the same file as the production code
  - Works much better for c / c++ than java
- Advantages
- Disadvantages

Convenient
Easy to work on ...

#ifdef TESTING
// All test code
#endif

test & SRC are mixed.
⇒ Harder to work in parallel. ⇒ Regression
⇒ Harder to manage releases.

- Test classes placed in same directory as production code

Same directory

- Access to protected methods.
- East of finding tests.
- Not as cluttered.
- Easier to work in parallel.

Hard to separate tests from production.

Same directory advantages

- Advantages
  - Easy to find tests
  - Easy to test part of the system
  - Can have visibility of internal variables
    - Use scope other than private to achieve this
- Disadvantages
  - Protected may expose core logic
  - May yield messy directory structure
  - Problems with test cases being released in jars
  - Configuration management issues

- Test cases are placed in a "test" subdirectory underneath the production code

Test subdirectory

```
com/
└── pragprog/
    └── wibble/
        ├── Account.java
        └── test/
            └── TestAccount.java
```

Tests exist in a test directory.

- Advantages

Test subdirectory

– Separate files

– Separate folder

– Still easy to find

- Disadvantages

- still in directory
- Lost protected access.
- Lost package scope
- May have lots of test directories.

Test subdirectory

Trees

Parallel

- Test cases are placed in a separate, parallel tree
  - All of the advantages of the code being in the same directory
  - Avoids the disadvantages
  - Can cause pathing issues with tools
  - Have to watch naming…