CS239

Nathan Sprague

February 7, 2012

Clicker Question

Evaluate this code in the context of PA#1:

```
1  public static int fiveScore(Dice aDice)
2      {
3          int result = 0;
4          for (int i = 0; i < 5; i++)
5          {
6              if (aDice.getFace(i) == 5)
7              {
8                  result += 5;
9              }
10         }
11         return result;
12     }
```

- A) No problems worth mentioning.
- B) Correct, with minor design problems.
- C) Correct, with serious design problems.
- D) Syntax or logic error.

Testing Happens at Multiple Levels

- Unit Testing - Test individual classes in isolation.
  - Focus is on making sure that each method works according to specification.
- Integration Testing - Test the interaction between classes.
- System Testing - Test the entire system in context.

Different Perspectives

- Black-box testing
  - Develop tests on the basis of class specifications and documentation.
- White-box testing
  - Develop tests on the basis of implementation.
  - Aim for high “code coverage”.

Test-Driven Development

- Write tests first.
  - Helps clarify specifications.
  - Helps avoid mistakes in development.

Developing Test Cases

- To *guarantee* correctness, test every possible sequence of method calls, with every possible input value.
  - Usually not possible.
- Instead, look for boundary conditions
  - Test at the boundary and on either side.
- Also test erroneous inputs
- Let's think about an example...

Regression Testing

- Testing is not a one-time process.
- Ideally, unit tests are maintained along with the code.
- This makes it easier to change the code:
  - All tests can be run after every change.

Unit Testing Frameworks

- To be useful, tests must be automated.
- JUnit...

Guidelines for Unit Tests

- Tests should:
  - **be small** - so that failures are easy to interpret.
  - **be independent** - so that failures are easy to interpret.
  - **be fast** - so they can be run frequently.
  - **not require interaction** - so they can be run frequently.
  - **have informative names** - so that failures are easy to interpret.
  - **use the correct assertion type**
    - e.g. assertEquals(a, b) not assertTrue(a.equals(b))