Lecture 8: unit testing

This Set

- Unit testing and JUnit

The problem

- Problems:
  - need to be able to make sure all parts are tested
  - need to know in testing exactly which part was not as expected
  - need to be able to keep the tests for modifications made later
- Unit testing helps overcome this problems of making sure everything is tested
  - Unit testing: test each class and each part of the class (unit) individually
  - Goal is to eliminate inconsistencies between the API and the actual working of the code

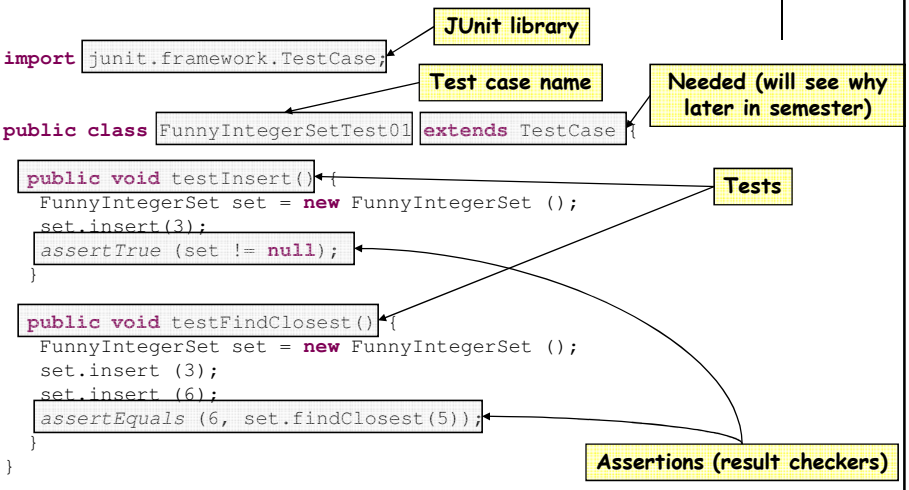
Unit Testing

- Unit testing helps overcome this problems of making sure everything is tested in a structured way
  - Unit testing: test each unit individually (micro level – each method or specifically each interaction described in the API)
  - Goal is to eliminate errors within classes
- Needs for unit testing
  - Method for defining tests = inputs, expected outputs
  - Method for running tests
  - Method for reporting results
- One possibility: write a driver for each class
  - Driver class contains main method
  - main method creates objects in class to be tested, calls methods, prints outputs
  - User checks outputs, determines correctness
  - Good: easy, no special tools needed
  - Bad: tedious, relies on human inspection of outputs
- Another approach: JUnit

JUnit

- A unit-testing tool for Java
- Includes capabilities for:
  - Test definition, including output checking
  - Test running (execution)
  - Result reporting
- Seamless integration with Eclipse
- Note
  - In this class we will use JUnit 3.8.1
  - So, when given a choice select **JUnit 3**

Structure of a JUnit 3.8.1 Test Case



A Test Case Is … A Class!

- assertion checkers
  - **`assertTrue(`***`expression`***`);`**
    - If `statement` is true, keep running test; otherwise, halt test, report "fail"
  - **`assertFalse(`***`expression`***`);`**
    - If `statement` is false, keep running test; otherwise, halt test, report "fail"
  - **`assertEquals(`***`expression1, expression2`***`);`**
    - If `expression1`, `expression2` equal, keep running test; otherwise, halt test, report "fail"
- If test terminates without failing an assertion and without throwing an uncaught exception, then it passes that test
- It continues with all subsequent tests regardless of passing or failing the current test

Hints on Testing

- Give names to tests that relate to class being tested
- Develop some tests before you code
  - Helps you to clarify what you are supposed to be doing
  - Gives you some ready-made tests to run while you code
- Use tests to debug
- How many tests?
  - Statement coverage: develop tests to make sure each statement in class is executed at least once (including constructors)
  - Decision coverage: develop tests to make each condition (if statement) in program both true and false
  - You should at least reach statement coverage in your own testing