Administrative Details

Programming assignments (30%)
- Weekly
  - Due Monday night (midnight)
- Lowest assignment dropped

Quizzes (10% of final grade)
- On Thursdays
- Will be brief. But you’ll need to know the material well. Little chance of figuring it out on the test.
- Lowest quiz dropped

Midterm (25%)
- One: Feb. 14

Final (35%)

Miscellaneous

See website:
- <www.cse.ucsd.edu/classes/wi06/cse8b>

Reading

Textbook
- Savitch, *Absolute Java, second edition*

Read readings before class
Print slides before class (available by 10 p.m. the previous day)
Take notes on slides during class
Read slides (including notes) after class
Read slides online with my inked markup
- Password to read inked slides is:
Read readings again, after class
You should do all self-text exercises in the reading
- Don’t turn them in
Review

CSE 8A:

Chapter 1
- Bytecode, Compiler, Interpreter
- Primitive types, casting (implicit, explicit), constants, expressions, precedence
- ++, other special operators, %
- String class (know methods used in book), Unicode

Chapter 2
- System.out.println(s)
- Scanner

Chapter 3
- If/else/else if
- Blocks/scoping
- switch
- Boolean expressions, &&, ||, short-circuiting
- Precedence, compound Boolean expressions
- Loops (do, while, for)
- Infinite loops, sentinel values, break, continue

Chapter 4
- Using new
- Calling methods
- Scoping, local variables, blocks
- Parameter passing
- this
- Return types
- Public and private
- Accessors and mutators
- Overloading, signature, type conversion and overloading
- Constructors
  - default (no parameters)
  - copy
  - calling methods from inside methods

Chapter 5
- Static methods, calling, creating, how differ from instance methods
- Main in any class
- Static variables
  - When to use in a class
- Math class
- Wrapper classes
  - Auto-boxing and unboxing
- Reference objects, parameter passing
- Class type parameters, null pointer exceptions
- Privacy leaks and deep copy constructors
- Immutable classes
Review

CSE 8A:
- Chapter 6
  - Array basics, looping over, using .length, index out of bounds
  - Arrays of objects
  - Arrays are reference variables, == and =
  - Passing arrays as parameters
  - Two-dimensional arrays

What we'll be covering

Java particulars
- Swing
- Applets
- Packages
- JavaDoc

General programming
- Frameworks/Object-Oriented Programming (OOP)
- UI programming
- Exception handling
- File I/O
- Generics
- Collections

Software Development
- Loop invariants
- Unit tests/Test-Driven Development
- Pair programming

Testing

How to test code?
Two different types of tests
- Unit test
  - Tests the Application Programming Interface (API) of your class
- Acceptance test
  - Tests that the program as a whole does what it should
- These types of tests should be separated

Acceptance Test
- Type in input
- Look at output
- Problem: ______________________
- Automate
  - Run the program with input from a file and output to a file
    - java Program < testInput.txt > testOutput.txt
  - Check the output it creates against known-good output
    - diff testOutput.txt knownGoodOutput.txt

Unit Testing

Tests at a lower level than acceptance testing
- Each test tests one specific test-case

We'll be using JUnit
- Written by Kent Beck
- Available at JUnit.org
- Similar testing frameworks many languages
  - See wikipedia entry for xUnit
- Already installed on ieng6 machines
Our Example

Temperature.java (from 8A)

- scale must be ‘C’ or ‘F’ (anything else uses ‘C’)

```java
public Temperature ()
public Temperature (double mValue, char mScale)
public double getTempInCelsius()
public double getTempInFahrenheit()
public void setTemperature(double mValue, char mScale)
public boolean equals(Temperature otherTemp)
public boolean equals(double mValue)
public boolean lessThan(Temperature otherTemp)
public void AminusBIntoA(Temperature otherTemp)
public void BminusAIntoB(Temperature otherTemp)
public Temperature AminusBIntoC(Temperature otherTemp)
public String toString()
public void swap(Temperature other)
```

Testing the Temperature Class

We want to write lots of small tests:

- Test each aspect of the class against the specification

```java
public void testTemperatureWithoutParamsIsCelsiusZero()
public void testCreateC()
public void testCreateF()
public void testConstructorWithBogusScaleUsesCelsiusInstead()
public void testCanRetriveCelsiusFromFahrenheit()
public void testCanRetriveFahrenheitFromCelsius()
public void testSwapCallingObjectAndArgument()
public void testToString()
public void testAMinusBIntoC()
public void testBMinusAIntoB()
public void testAMinusBIntoA()
public void testEquals()
public void testNotEquals()
public void testLessThan()
public void testNotLessThan()
public void testEqualsValue()
public void testNotEqualsValue()
public void testSetTemperatureF()
public void testSetTemperatureC()
public void testSetTemperatureX()
```

Getting Started

Create a new file: TestTemperature.java

Compile and run with:

```
java TestTemperature
```

```
import junit.framework.TestCase;

public class TestTemperature extends TestCase
{
    public static void main(String[] args) {
        junit.textui.TestRunner.run(TestTemperature.class);
    }
}
```

```
Time: 0.009
There was 1 failure:
1) warning(junit.framework.TestSuite$1)
    junit.framework.AssertionFailedError: No tests found in TestTemperature
    at TestTemperature.main(TestTemperature.java:6)
FAILURES!!!
Tests run: 1, Failures: 1, Errors: 0
```

Add a test

Tests are public void methods with no parameters that begin with test

Compile and execute:

```
public void testTemperatureWithoutParamsIsCelsiusZero()
{
    Temperature t = new Temperature();
    assertEquals(t.getTempInCelsius(), 0, 0.0001);
}
```

```
Time: 0.011
OK (1 test)
```
Add more tests

Tests are public void methods with no parameters that begin with test

public void testCreateC()
{
    Temperature t = new Temperature(31.5, 'C');
    assertEquals(t.getTempInCelsius(), 31.5, 0.0001);
}
public void testCreateF()
{
    Temperature t = new Temperature(212, 'F');
    assertEquals(t.getTempInFahrenheit(), 212, 0.0001);
}

Compile and execute:
...
Time: 0.012
OK (3 tests)

Get rid of redundancy

DRY: Don’t Repeat Yourself

protected void assertTempEquals(double temp1, double temp2)
{
    final double allowableDifference = 0.0001;
    assertEquals(temp1, temp2, allowableDifference);
}
protected void assertCelsiusEquals(Temperature t, double temperature)
{
    assertTempEquals(t.getTempInCelsius(), temperature);
}
protected void assertFahrenheitEquals(Temperature t, double temperature)
{
    assertTempEquals(t.getTempInFahrenheit(), temperature);
}

Get rid of redundancy

Change:

public void testTemperatureWithoutParamsIsCelsiusZero()
{
    Temperature t = new Temperature();
    assertEquals(t.getTempInCelsius(), 0, 0.0001);
}
public void testCreateC()
{
    Temperature t = new Temperature(31.5, 'C');
    assertEquals(t.getTempInCelsius(), 31.5, 0.0001);
}
public void testCreateF()
{
    Temperature t = new Temperature(212, 'F');
    assertEquals(t.getTempInFahrenheit(), 212, 0.0001);
}

Run test again!

When a test fails

....F.............
Time: 0.06
There was 1 failure:
1) testConstructorWithBogusScaleUsesCelsiusInstead(TestTemperature)
junit.framework.AssertionFailedError: expected:<212.0> but was:<213.0>
at TestTemperature.assertTempEquals(TestTemperature.java:22)
at TestTemperature.assertCelsiusEquals(TestTemperature.java:27)
at TestTemperature.testConstructorWithBogusScaleUsesCelsiusInstead(TestTemperature.java:55)
at sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method)
at sun.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAccessorImpl.java:39)
at sun.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.java:25)
at TestTemperature.main(TestTemperature.java:16)
FAILURES!!!
Tests run: 20, Failures: 1, Errors: 0
Another test

Add another test.

```java
public void testCanRetriveCelsiusFromFahrenheit()
{
    Temperature t = new Temperature(212, 'F');
    assertCelsiusEquals(t, 100);
}
```

Remove more redundancy

We keep creating celsius 31.5 and fahrenheit 212 temperatures.
Instead, declare as instance variables

Initialize in setUp() method

- Runs before every test

```java
private Temperature c31;  
private Temperature f212;

protected void setUp()
{
    c31 = new Temperature(31.5, 'C');
    f212 = new Temperature(212, 'F');
}
```

Use in test methods:

```java
public void testCreateC()
{
    assertCelsiusEquals(c31, 31.5);
}
```

tearDown() is run after every test

Makefile

Copy the makefile from ~/../public/makefile

To build and run tests (all classes beginning with Test)

- make

To cleanup (removes all .class files)

- make clean

To build without running tests

- make