CS 211: Potpourri
 Enums, Packages, Unit Tests,
 Multi-dimensional Arrays
 Command Line Args

Chris Kauffman

Week 9-1
This Week

- Mon: Potpourri
- Wed: Exceptions
- Thu: Lab 09 Task

Today

- P4
- Packages
- Unit Testing
- Command Line Arguments

BJP Reading

- Chapters 4, 7, 9
- Topics somewhat spread out

Lab Manual Reading

- 09. Enumerations
- 11. Command-line Arguments
- 12. Junit
- 10. Exceptions
P4: Modern Gems

- Larger project: start early
- Test cases available soon
- Examine class structure to get a high level feel
- Questions?
Exercise: Entities Assemble!

Java has 4 top-level entities

From Lecture:
- Concrete classes
- Abstract classes
- Interfaces

From Lab 08:
- Enumerations

Tell Me

1. How to declare something of each kind
2. Restrictions on each kind
3. A standard use of each kind
A namespace where classes live

- Examples in today’s code pack
- Two Fu classes: kung.Fu and sna.Fu
- Distinct classes can both be utilized by PackageFu due to packaging and directories
- Examine
Real World

- Your company X produce library with class Fu
- Company Y licenses library but also has a class Fu
- NP: refer to yours in their code as com.company.utils.Fu
- Must be buried in a class path directory com/company/utils/Fu
- For purposes of your class projects, we keep things simple as possible: no packages. May change in the future
It’s a packaged world

- Every variable and function is part of a class or interface
- Every class/interface is part of a package
- The java library is divided into packages containing classes
- There is a default package that unspecified classes live in
  - It’s screwy: other packages can’t look inside
  - Command line and DrJava don’t care
  - Eclipse and NetBeans probably do
  - Pay attention to project specs
Unit Testing

Typically don’t want programs throwing exceptions at customers
▶ Tend to not buy your product anymore
▶ Want to uncover problems early
▶ **Unit testing** means to test code to see if it produces correct results
▶ Tests *units*
  ▶ Single class
  ▶ Single function

Alternative: **top-down testing**
▶ Test whole module or program or at once

**Compare** these two
Commonly used framework to test java code: http://junit.org/

- Write class
- Each method can be a @Test
- Does something (new another class, call methods)
- Check if answer is correct, use an assert-type methods
  - assertTrue(..), assertFalse(..), assertEquals(x,y)
@Test
public void test_VigenereCipher_4(){
    Cipher c = new VigenereCipher("abcdefg", lowers)
    assertEquals(lowers, c.getAlphabet());
}

@Test public void test_VigenereCipher_6(){
    Cipher c = new VigenereCipher("SLIME");
    assertEquals("YcMQR", c.encrypt("GREEN"));
}

@Test(timeout=2000) public void test_Vigenere_28(){
    try {
        Cipher c =
            new VigenereCipher("password", lowers);
        String s = c.encrypt("HELLO");
    } catch (NotInAlphabetException e){
        return;
    }
    fail("shouldn’t encrypt any characters not in the alphabet.");
}
Import Magic

Some tricks are used to make it more convenient to test

Normal

```java
import org.junit.Assert;
...
@Test public void test_VigenereCipher_6(){
    Cipher c = new VigenereCipher("SLIME");
    Assert.assertEquals("YcMQR", c.encrypt("GREEN"));
}
```

static import magic

```java
import static org.junit.Assert.*;
...
@Test public void test_VigenereCipher_6(){
    Cipher c = new VigenereCipher("SLIME");
    assertEquals("YcMQR", c.encrypt("GREEN"));
}
```
Java Annotations

- @Information for the compiler
- Like comments but the compiler may not completely ignore
- Metadata that summarizes the intent of code

Examples

@Test  This code tests other code (compiler may just ignore)
@Deprecated  This code is old, unsupported, may disappear
@Override  Error if not overriding parent method
That’s what testing is

Unit tests are just more code

- Intended to test other code
- Can have bugs (what’s testing the tests?)
- Can miss critical stuff

junit provides convenience functions for testing

- `Assert.assertEquals()`, `Assert.assertTrue()`, `Assert.assertNotSame()`, etc

Like saying ’here is what a loop is’.

- To really learn loops, solve some loopy problems
- To really learn testing, write some tests
- May have to write tests for a project (P5???)
- Very common job in industry

Practice

Example.java and ExampleTests.java
Running Tests

- IDEs usually provide niceness (Test button in DrJava)
- Command line - like running any other class
- Need access to functions in junit
- Live inside junit-cs211.jar

```
> javac -cp .:junit-cs211.jar *.java
> java -cp .:junit-cs211.jar P3Tests
```

What do the above lines do?
How about these this line?

```
> jar tf junit-cs211.jar
```
Multidimensional Arrays: Useful for P4

Examples: MultiDimensionalArrays.java

Most common: 1D

```java
int [] ia = new int[n];
ia[0] = 1;
```

Still common: 2D (Board)

```java
int m=3, n=7;
...
double da[][] = new double[m][n];
da[i][j] = 1.23;
```

Less common: higher dimensions

```java
String sa[][][] = new String[5][10][20];
sa[4][9][19] = "last";
```

Iterating/printing are interesting with arrays, demos in MultiDimensionalArrays.java
Demo: Connect 4 Board

public static ??? readBoard(String filename)

Read text files representing board, return 2D array of something

c4_1.txt:
1 2 3 4 5 6 7
. . . . . . . 1
. . . . . . . 2
. . . . . . . 3
. . . . . . . 4
. . . . . . . 5
. . . . . . . 6
. . . . . . . 7

c4_2.txt:
1 2 3 4 5 6 7
. . . . . . . 1
. . . . . . . 2
. . . . . . . 3
. . . . . . . 4
. . . . . . . 5
. . . . . . . 6
. . . . . . . 7

c4_3.txt
1 2 3 4 5 6 7
. . . . . . . 1
. . . . . . . 2
. . . . . . . 3
. . . . . . . 4
. . . . . . . 5
. . . . . . . 6
. . . . . . . 7

c4_4.txt
1 2 3 4 5 6 7
. . . . . . . 1
. . . . . . . 2
. . . . . . . 3
. . . . . . . 4
. . . . . . . 5
. . . . . . . 6
. . . . . . . 7

Source: Wikihow
Command Line Arguments

Pass command line arguments to a program

```
java CmdArgs here are 4 args
```

Use the arguments: stored in the args array of main

```
public class CmdArgs {
    public static void main(String args[]){
        if(args.length >= 4 &&
            args[0].equals("here") &&
            args[1].equals("are") &&
            args[2].equals("4") &&
            args[3].equals("args")){
                System.out.println("Hurray!");
            }
    }
}
```

See also CommandLine.java
Exercise: Check for repeated first

Write a `main()` method which checks if the 0th arg appears as any other argument.

> javac RepeatedFirst.java
> java RepeatedFirst
No words given
> java RepeatedFirst ding
ding not repeated
> java RepeatedFirst ding a ling
ding not repeated
> java RepeatedFirst ding a ling a ling ding a ding repeated at position 5
> java RepeatedFirst ling ding a ling a ling a ling ling repeated at position 3
Exercise: Word Count

Write a `main()` method which

- Treats each command line arg as file name
- Opens each file using a `Scanner`
- Counts how many space-separated words appear in the file
- Closes the scanner
- Reports the results on screen
- Repeats for the next file in the args list

> javac WordCount.java
> java WordCount
> java WordCount WordCount.java
WordCount.java has 41 words
> java WordCount WordCount.java RepeatedFirst.java Example.java
WordCount.java has 41 words
RepeatedFirst.java has 39 words
Example.java has 137 words