CS 211: Existing Classes in the Java Library

Chris Kauffman

Week 3-2
Logistics

- P1 Due tonight: Questions?
- Late policy?
- Lab 3 Exercises Thu/Fri
- Play with Scanner
- Introduce it today

Goals

- Class Library and import
- Math, Array, String, Scanner
- Simple input from users/files
- Next week: Make Classes
Classes

```java
public class C {
    public static int f;       // static field
    public static void m(){   // static method
        int i;               // local variable
        ...
    }
}
```

- All methods and **fields** in java live within a class (or interface)
- Classes are partly a namespace: a place for names to exist
- **Static** methods and fields belong to the class: There is one for the whole class
- Consider `CallCounts.java` to see a demo of static methods and fields
- Draw some pictures to see how this looks in memory
- Class definitions live in global memory
Java Namespaces

- No such thing as a "global" variable
- No such thing as a "global" function
  - Every variable in a scope
  - Every scope in a class
  - Every class in a package
- `static` means class-level
  - There is one of it (method/field)
  - It is not associated with any particular object
  - It can be accessed through the class using dot
- Related concept: `namespace`, library system, package
- Access to stuff in classes is controlled
  - `public` - everybody (use this for the first project)
  - Soon `private`, `protected`, default
It’s a packaged world

- Every method/field is part of a class or interface
- Every class/interface is part of a package
- The Java library is divided into packages of related classes
  - java.lang: Essentials, automatically imported (Math, String, Integer, Object)
  - java.util: Mostly essential stuff (Scanner, Arrays)
  - java.util.concurrent: For multiple processors
  - javax.swing: GUI stuff (windows, buttons)
- There is a default package that unspecified classes live in
  - Default package is screwy: other packages can’t look inside
  - Command line and DrJava don’t care
  - Eclipse and NetBeans probably do
  - Pay attention to project specs
Mostly Static Classes

java.lang.Math

- Javadoc for Math class
- For mathy operations and a few useful constants
- Calculate the square root of pi?
- Calculate E to the 2.75 power?

java.util.Arrays

- Javadoc for Arrays class
- Useful ops for arrays
- Sort an array of doubles? of booleans?
- Nicely stringify an array of integers for printing?

Answers are in the provided MathAndArrays.java
Math and Arrays are oddities

- Most classes in java don’t consist of all static members.
- Most classes you do stuff like

  ```java
  SomeClass s = new SomeClass(arg1, arg2);
  s.doSomething(arg3);
  ```

- But you’ll never do

  ```java
  Math m = new Math();
  Arrays a = new Arrays(stuff);
  ```

  because these don’t have a constructor
String s;
s = new String("hello");
System.out.println(s);
String t = "sweet stuff";

Keeps track of length

int n = s.length();
int m = t.length() - 1;
Wait, that’s really confusing

- A **field** length (for arrays)
- Versus a **method** length() (function for String)

```java
char ca[] = new char[10];
String s = "0123456789";
if(ca.length == s.length()){
    System.out.println("Same size");
}
```
Methods of Strings

Start reading Java Docs:
http://docs.oracle.com/javase/8/docs/api/java/lang/String.html

Look for the following methods

▶ Dynamically construct a string
▶ How to retrieve a specific character from a string
▶ How to concatenate two strings
▶ Determine if a string starts with another string
▶ How to pull a substring out of a string
▶ Compare two strings for equality?
▶ How to add characters onto the end of a string
Which brings me to my next point

Concatenation

Diagram 1
What does

String a = "hello";
String b = " world";
String c = a+b;

actually do in memory?

Diagram 2
How about

String s = "";
for(int i=0; i<10; i++){
    s = s + i;
}

String Equality

Show a memory Diagram

String a = new String("hello");
String b = a;
String c = new String("hello");
String d = a + "";

What is printed

System.out.println(a == b);
System.out.println(a.equals(b));
System.out.println(a == c);
System.out.println(a.equals(c));
Scanner

Sometimes you need input. Scanner is good for this.

// Short demo of the scanner class for input
import java.util.Scanner;
public class ScannerDemo{
    public static void main(String args[]){
        Scanner input = new Scanner(System.in);
        int i = input.nextInt();
        double d = input.nextDouble();
        String s = input.next();
        System.out.println("" + i + " " + d + " " +s);
    }
}

See the larger ScannerDemoBig.java for more info

▶ How do you know about all these methods for Scanner?
▶ What about String? or System?
▶ Where do you suppose mathematical functions are stored?
Constructors for Scanner

Read from the Terminal

// Constructs a new Scanner that produces values
// scanned from the specified input stream.
Scanner(InputStream source)

Scanner in = new Scanner(System.in);

Read from a String

// Constructs a new Scanner that produces values
// scanned from the specified string.
Scanner(String source)

Scanner in = new Scanner("Give me a 1 Give me a 2");
Scanner Basics

In java.util

- Several constructors, for System.in and File sources
- `nextInt()`, `nextDouble()`, `next()`, etc.
- `nextLine()`: whole line
- `hasNext()`: true if more to read
- `close()`: when reading files
File class

- Lives in the package `java.io`
- `import java.io.File;`
- "An abstract representation of file and directory pathnames."
- How do we get at it?

```java
File f = new File("some-file.txt");
String s = "stuff.dat";
f = new File(s);
```

Useful for many things but our primary interest at the moment is for reading from files using a `Scanner`
Reading from files with Scanner

// Constructs a new Scanner that produces values
// scanned from the specified file.
Scanner(File source)

open Scanner fin = new Scanner(new File("myfile.txt"));

read int i = fin.nextInt();
  ▶ Frequently done in a loop

close fin.close();

See ScanAFile.java
A Closing Problem: The Longest Word

public static String longestWordWord(Scanner in)

- Takes an open Scanner
- Reads to the end of input
- Returns the longest word in the stream
- Ties go to earlier word
- Return "" if no words in the stream

> import java.util.Scanner;
> LongestWord.longestWord(new Scanner("word1 word123"))
"word123"
> String s = "some gargantuan words and tiny words";
> Scanner in = new Scanner(s);
> LongestWord.longestWord(in)
"gargantuan"
> LongestWord.longestWord(new Scanner(" "))
""