CS100M/CIS121/EAS121

Introduction to Computer Programming

Spring 2004

Lecture 14

Introduction to Java

Announcements

- Prelim 2 on March 18
  - everything up to and including subfunctions
  - review material, topics on Prelims
  - rooms TBA
- A4 coming up (see schedule on Assignments)
- CIS/EAS gone, but not forgotten...

Background

- Problem solving in a new language
- Concepts same (languages, design) but implemented differently by each language
- Languages sometimes have themes (overriding philosophies):
  - MATLAB: everything is an array
  - Java: everything is an object
- OOP:
  - object oriented programming
  - verbs: now called methods, operators very similar
  - nouns: constants, variables, fields, objects, classes

Overview

- Java Language:
  - software, creating, compiling, running
  - whitespace and comments
  - alphabet
  - tokens
  - statements
- Statements:
  - declaration, assignment, empty, I/O, object creation
  - control: selection, repetition
  - methods (think functions)
- OOP (our focus, but first...the language....)

Accessing Java

- Java info on website:
  - **More Java!**
  - has most links I'm showing in these notes
- Download from Sun (already in labs, free!):
  http://java.sun.com/docs/books/tutorial/getStarted/cupojava/index.html
  - use version **1.4.2** (at least 1.4 but not 1.5!)
  - Puts language and command-line tools "in" your computer
  - See **Java Applications** in lecture notes
- DrJava:
  - friendly (and free) environment (also in labs)
  - See **DrJava** link on course website (download most recent version)

Basic Java Application

- Everything goes in a ***class***
  - collection of data and methods
  - basic version:

```
public class name {
   data
   methods
}
```

- Ideally, one public class per file
  - filename is name of public class
  - program consists of collection of these classes
  - one class must have a method called main:

```
public static void main(String[] args)
```

  - name of program is essentially class that contains `main`

Example

```
// Simple program that greets user

public class HelloWorld {

   public static void main(String[] args) {

      System.out.println("Hello, world!");
   }

}
```

Running in DrJava?
Running from DOS prompt?

Alphabet

- UNICODE
  - use `\uxxxx` to get specific character
  - we focus on ASCII characters (pg 77, Appendix 3)
- use single quotes:

```
System.out.println('a');
System.out.println('A' == 'a');
```

- Escape characters:
  - special characters that perform actions
  - see page 77...example of new line:

```
System.out.println("Hi\nthere!");
```

Whitespace and Comments

- Single line comments:

```
// I am a comments
/* I am also a comment */
```

- Multiline comments:

```
/* This is ignored
   This is also ignored
   Me, too */
```

- Whitespace:
  - use as much as you want: it's ignored
  - but, do not split a token...eg, what's wrong below?

```
Sys tem.out. println ( " He ll o! " ) ;
```


Punctuation

- Statements: end with semicolon! (`;`)
- Classes and methods use braces (`{ }`):

```
signature { code }
```

- Blocks of statements also use braces (no "end"):

```
if (x == true) {
  count++;
  System.out.println(x);
}
```

- Methods and expressions use parens (`()`)
- See `HelloWorld.java`

Reserved Words/Keywords

- See pg 45, 892
- Common for us (there are others):
  - values: `true false null this`
  - types: `boolean int double char void`
  - control: `if else for do while return switch`
  - modifiers: `public private static final protected`
  - classes: `new class extends implements super interface`
- Amusing: `goto`

Identifiers

- Reminder:
  - names of things (variables, methods, class names)
  - case sensitive
  - start with underscore (_), letter
  - can contain any UNICODE letters, number, and currency symbols (but avoid!)
  - no reserved words!

Identifiers: Variables

- Variables:
  - must have type before using (declaration statement):
    ```
    int x;
    ```
  - must have value before using (no assumed values for methods and method parameters):
    ```
    int x;
    x = 1;
    int y = 2;
    ```
  - for fields (a class's data), variables get defaults of "zero" see `Variables.java`

13

Variables: Types

- Java is ***strongly typed***:
  - every variable must have a declared type before using
  - types must match expected types (see method headers)
- Primitive types:
  - solid/whole, specific value
  - typical for us: `int`, `boolean`, `char`, `double`
- Reference (class) types:
  - composite types, uses notion of class
  - for now, we'll use `String`

14

Example Of Lots of Stuff

```
public class PersonTester {
  public static void main(String[] args) {
    String first = "Dimmu";
    String last = "Borgir";
    Person p;
    p = new Person(first,last,123456);
    System.out.println(p);
  }
}

class Person {
  private String first;
  private String last;
  private int id;
  public Person(String fn, String ln, int i) {
    first = fn;
    last = ln;
    id = i;
  }
  public String toString() {
    return "Person: " + first + " " + last + ", id: " + id;
  }

}
```

15

Constants

- Primitive:
  - `boolean`: `true`, `false`; no 0s and 1s!
  - `char`: single quotes around a character (see **Alphabet**)
  - `int`: -2147483648 to 2147483647
  - `double`:
    - decimal point: `0.1`, `.1`, `1.`, `1.0`
    - scientific notation: `1e-6`, `1.23E2`
- Reference: `null` (no object)
- making a variable `final`:
  ```
  final double PI = 3.14;
  ```
  - variable cannot be changed after set until code finished

16

Strings

- Rem: collection of characters
  - Object in Java! Not the same as characters!
  - Use double quotes `"characters"`
- Creating:
  - Long way to create:

```
String s = new String("hello")
```

  - Short cut:

```
String s = "hello";
```

  - Empty: `""`, different than `null`!
  - Concatenating:

```
String s = "hi" + " there!";
```

17

Operators

- See Appendix 2
- arithmetic: `+`, `-`, `*`, `/` , `%` (`-` also for negation)
- logic: `&`,`&&`,`|`,`||`,`!`,`^`
- comparison: `==`,`!=`,`<`,`<=`,`>`,`>=`
  - note: compare 2 *strings* with `.equals`

```
s1.equals(s2)
```

  - `==` does not compare contents of Java strings!
  - will be used for comparing objects later on
- assignment: `=`,`+=`,`*=`,`-=`,...
- increment: `--`,`++`

18

Operators Example

```
public class Operators {
  public static void main(String[] args) {

   boolean x = true || false;

   char c = 2 + 'a';

   double d = (1 + 2)/2.0;

   System.out.println(5 <= 6);

   String s1="A"; String s2="a";
   System.out.println(s1.equals(s2));

   System.out.println(2.123 % 5.343);

  }
}
```

19

More Operators

- casting: `(type)`

```
double x = 7.7;
int y = (int)x;
System.out.println(y);

double x = Math.random()*2;
int y = (int)x;
System.out.println(y);
```

- others?
  - objects: `.`, `new`, `instanceof`
  - arrays: `[]`
  - conditional: `?`

20