CS100M
Introduction to Computer Programming

Spring 2004
Lecture 15
Java Statements

Announcements

- Prelim 2 info on website (review, topics, rooms)
- A4 due date
- DIS office hours for today 2-3 (not 1-2)
- Dr Java and labs
- A3 comments:
  - review grading comments (don't repeat mistakes)
  - submit all required files (non-compile: 0)
  - format: indentation, text files & line wrap
  - comments: brief, relevant
  - compiling: any warning mesg means 0
  - testing: test your code! be exhaustive
  - debugging: reminder about tracing

Overview

- Programming language: chars, tokens, statements
- Java statements:
  - empty
  - block
  - expression
  - declaration
  - assignment
  - labeled
  - selection
  - repetition
  - function call
  - return
  - object creation
  - others...

Empty

- Just like MATLAB:
  semicolon for “succeed”:
  ```
  ;
  ```
- example)
  ```
  for(int i=1;i<1e6;i++);
  ```

Block

- Block Statement:
  - no "**end**" in Java
  - need to use punctuation with braces (**{ }**)
- syntax:

  { statements }
  - no semicolon after closing brace
  - can have 0, 1, or many statements
  - can include other blocks
  - variables and blocks? (coming up)
- example:

```
if (x == 10) {
   System.out.println("Hello!");
   System.out.println("x is " + x);
} else
   System.out.println("Bye!");
```


Expression

- Expression Statements:
  - cannot just say **1+1;** as a complete statement
  - expr stats: assignments, increment/decrement, method calls, object creation
- Expressions:
  - similar rules as MATLAB, but now have to deal with types and some different operators
  - ***type promotion***: mixed types in an operation give a result of the higher type
    - **char** < **int** < **double** (leaving out some)
      - **int** combined with **double** → _____
      - **int** combined with **char** → _____
      - **char** combined with **double** → _____
    - **String** addded to anything → _____
- a bit more about operators....

Basic Operators

- arithmetic
- logic
- relations
- assignment
- increment/decrement
- precendence and associativity: Appendix 2

Some Examples

- String promotion, output:

```
System.out.println(1 + 1);
System.out.println("1" + 2);
System.out.println("1" + 2 + 3);

String s = "";
s = s + 3;
System.out.println(s);
```

- Other examples:

```
System.out.println(1/2.0);
System.out.println(6/7);
System.out.println('a'/97);
```

Casting

- Use casting to "defeat" promotion or use notion that some types "include" others
- syntax of cast operation:
  (type) expression
  - returns value of expression as ***type***
  - the "non-type" portion is effectively lost/truncated
  - precedence is higher than arithmetic operators
- examples:

```
int x;
x = (int) 7.7;
System.out.println(s);
int x = (int)(Math.random()*(11);
double y = 10;
System.out.println((char)('a'+1.0));
```

9

Declaration

- Reminder about variables:
  - all variables have types (strongly typed)!
  - variables may be declared only once in a block!
    (revisit this rule after reviewing the Scope panels)
  - all variables must have values before used!
  - all variables do not have initial values in methods!
    (when declared in class, defaults are "zero")
- Declaration syntax:

```
type name;
type name = expression;
final type name = expression;
```

  - can use shortcut:
  - can appear in level of class, method param, local variable in method (as any valid statement)

10

Introduction to Scope

- _____ ?
- Life of a variable in a block (including method body):
  -
  -
  -
- Method:
  - variables as params: "created" when method is called
  - variables in decl stats in method body: "created" statement is reached
- Example:

```
public double add(int a, int b) {
   int c = 1;
   return a + b + c;
}
```

11

Example for Vars in Methods

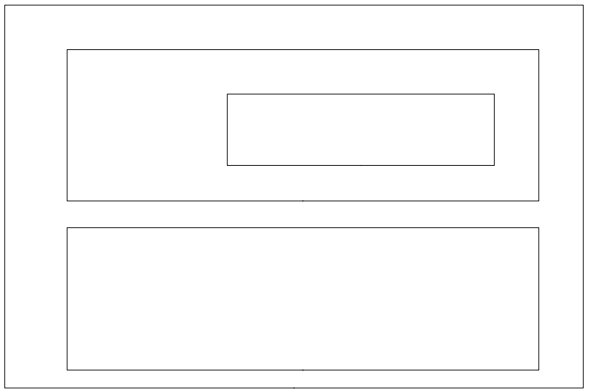
```
public class Vars {

  public static void main(String[] args) {

    int x = 2;
    System.out.println(add(x,x));

  }

  public static int add(int x,int y) {

    return x + y;

  }

}
```

12

Variables in Blocks

- Method body is effectively a block
- So, what is the more general rule for blocks?
  - variable "created" by an enclosing block are "seen" inside an enclosed block
    - visible to both blocks, assuming the varible was created before the enclosed block is encountered
    - changes to variable inside the enclosed block are OK
  - variable "created" only inside an enclosed block is not seen by an enclosing block
  - blocks can be nested (an outer block's variable will be seen by all inner blocks, assuming the variable is created before the first inner block)

Depiction of Scope



Another Example of Scope

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

     int x = 1;
     System.out.println("S1 for x: "+x);

     {
        System.out.println("S2 for x (before changing): "+x);
        x = x + 1;
        System.out.println("S2 for x (after changing):  "+x);
        int y = 3;

        {
          System.out.println("S3: "+(x + y));
        }

     }

     String y = "yes, this is legal";
     System.out.println("S1: "+y);

  }

}
```


Assignment

- Store result of expression in a typed variable
- Syntax:

```
type name = expression;
name = expression;
final type name = expression;
```

  - must declare var before using
  - expression type must match var type (OOP will modify this rule a bit)
- Example

```
public class Scope2 {
  public static void main(String[] args) {
    int x = 1;
    blah(x);
    System.out.println(x);
  }
  public static void blah(int x) {
    x = 2;
  }
}
```

Increment/Decrement

- Increment:
  - irritating to type `x = x + 1;` all the time
  - special shorcut: `x++;` or `++x;`
- Pre-increment:

```
int x = 1;
int y = ++x;
```

- Post-increment:

```
int x = 1;
int y = x++;
```

- Other kinds of increment? decrement?

Control Flow

- nice overview at
  http://java.sun.com/docs/books/tutorial/java/nutsandbolts/flowsummary.html
- Branching:
  - labeled statement, `continue`, `break` (avoid!)
  - `return`, `break` inside `switch` (good)
- Selection:
  - `if else`
  - `switch`
  - `try`, `catch`, `finally`
- Repetition:
  - `do`, `while`
  - `for`

Others?

- coming up!
  - Method call
  - Object creation

Suggested Exercises

- Download `SavitchIn.java` and learn how to prompt the user to enter integers, doubles, and strings.
- Write a program that prompts the user for their first name, last name, and then reports their full name.
- Write a program that prompts the user to enter a low value and high value and then reports a random integer between those values.
- Write a program that prompts the user for three integers and then reports their average in terms of a double.
- Write a program that outputs the integer value of the character `*`.