Java Basics
CS 99 – Summer 2000
Michael Clarkson
Lecture 2

Administration
- Lab 1 in progress
  - Due tomorrow, at beginning of lab
  - Submit on a floppy
- Lab 2 posted today
- Put a check by your name on the attendance sheet – add your name if it isn’t there
- This is the last day to enroll in the class!
- Still need questionnaire from at least 8 people

Agenda
- Variables
- Assignment
- Expressions
- Methods

Declaring Variables
- Syntax:
  type name [= init] [, name = init];
- Examples:
  int y;
  int x = y;
  double pi = 3.14;
  String hi = “Hi!”;
  int a = 1, b, c = 2;

Types
- Every value in a program has a type
- Every variable, since it holds values, also has a type
- Java has some built-in types called intrinsic types, a.k.a. primitive types
- Programmers can also create their own types using classes
Intrinsic Types

- Integers
  - Numbers that are whole valued and signed
  - e.g., 5, -1000, 42, 0
  - Java types byte, short, int, long
- Floating point numbers
  - Numbers that have a decimal component
  - e.g., 3.14, 1.78, .9944, -1.69, 1.0, 0.0
  - Java types float, double
- We'll usually use int and double

Intrinsic Types [2]

- Characters
  - The symbols in a character set, such as letters, numerals, punctuation, etc.
  - e.g., 'a', 'b', 'c', 'X', 'Y', 'Z', '!', '"'
  - Java type char
- Booleans
  - Values that are either true or false
  - true, false
  - Java type boolean

The String type

- String is an example of a user-defined type
- Strings are sequences of characters
- e.g., "Hello, world!", "1 + 1 = 2"

Naming Variables

- Follow Style Guide
- First character in name must be a letter
- Remaining characters can be letters, numbers, or the underscore "_" (e.g., cs99_2000su)
- Can be (practically) as long a name as you want

Literals

- Variables are placeholders for values in a program
- Literals are actual values written directly in a program:
  int x = 5;
  double y = x + 2;
  String s = "5 + 2 = " + y;
- Literals above: 5, 2, "5 + 2 = "

Scope

- Scope is the lifetime of a variable
- Variables are live from the statement where they are declared to the end of the block that statement is in
- You cannot use a variable if it is not live (in scope)
- Implication: variables must be declared before using them
Assignment Statement

- Syntax: 
  \( \text{variable} = \text{value} \)
- Examples:
  
  \[
  x = 5; \\
y = x; \\
z = x + y; \\
d = \text{round}(b) - 1;
  \]

A Special Assignment

- What does this mean:
  \( x = x + 1 \)
- Take the value of \( x+1 \) and store it in \( x \)
- So if \( x \) equaled 1 before executing the statement, it would equal 2 afterwards:
  
  
  // \( x = 1 \) \\
  \( x = x + 1 \) \\
  // \( x = 2 \)

Default Values

- What would this output?

  ```java
  int x;
  System.out.println(x);
  ```
- Answer: every variable is initialized to a default value, if you don’t provide one
  - Numeric types (int, double): 0
  - Boolean: false
  - Strings: “” (the empty string, or the null string)

Expressions

- Values combined by operators
- Has a value, and therefore a type

Operators

- Operators allow values to be combined
- Categories of operators
  - Arithmetic
  - Relational
  - Logical
  - (Bitwise)
- Unary, binary, ternary

Arithmetic Operators

<table>
<thead>
<tr>
<th>Operator</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>+</td>
<td>Addition</td>
</tr>
<tr>
<td>-</td>
<td>Subtraction</td>
</tr>
<tr>
<td>*</td>
<td>Multiplication</td>
</tr>
<tr>
<td>/</td>
<td>Division</td>
</tr>
<tr>
<td>%</td>
<td>Modulus</td>
</tr>
<tr>
<td>++</td>
<td>Increment</td>
</tr>
<tr>
<td>--</td>
<td>Decrement</td>
</tr>
<tr>
<td>+=</td>
<td>Addition assignment (also -=, *=, /=, %=)</td>
</tr>
</tbody>
</table>
Arithmetic Operators [2]
- Operands are numeric types
- Resulting value is a numeric type
- Unary minus
- Division and Modulus
- Assignment Operators
- Increment and Decrement

Relational Operators [2]
- For equality operators:
  - Operands must be of the same type
- For ordering operators:
  - Operands must be numeric type
- For both:
  - Resulting value is a boolean
- Common errors:
  - Equality operators don’t work on strings
  - Using the assignment operator instead of the equality operator

Logical Operators
- AND assignment, OR assignment
- & =, | =
- NOT
- !
- OR
- ||
- AND
- &&

Assignment Operator
- Assignment statement acts as an operator
- Resulting value is the value from the RHS of the assignment
  - e.g., value of x = 2 is 2
- So we can write:
  - x = y = 2;
- Final values of x and y?

Order of Operations
- Precedence
- Associativity
Methods

- Transfer of control
- Return types and values
- Parameters and arguments
- Walkthrough of a method call

Transfer of Control

```java
class MethodExample {
    static void method1() {
        System.out.println("1");
    }
    static void method2() {
        System.out.println("2");
        method3();
    }
    public static void main(String[] args) {
        method3();
        method2();
        method1();
    }
    static void method3() {
        System.out.println("3");
    }
}
```

What is the output from this code?

**return statement**

- The `return` statement can occur anywhere in the body of a method
- Its syntax is: `return [expression];`
- Examples:
  - `return;`
  - `return 5;`
  - `return x;`
  - `return x * y / 2;`

**return statement [2]**

- `return` means “stop executing this method and return to where it was called”
- If the method has a return type, the expression after return is evaluated, and the value is substituted for the method call

Return Types

- Methods can have return types and values
- So method calls can be used as values in expressions
- `void` means “this method has no return type”
- The type of the method and the type of the return must match!

Return Type Example

```java
public static void main(String[] args) {
    int x = 1;
    x = x + foo();
    System.out.println(x + bar());
}
static int foo() {
    return 5;
}
static int bar() {
    return 4/2;
}
```

What is the output from this code?
Parameters & Arguments

- Methods can be declared with parameters:
  
  ```java
  static double average(double n1, double n2, double n3)
  ```

- When you write a call to such a method, you provide arguments:
  
  ```java
  double avg;
  ...
  avg = average(x, 10/7, Math.sqrt(2));
  ```

Matching Params. & Args.

- Parameters in a method are assigned the values of the arguments in the method call in the order they occur

  - So in the previous slide:
    - n1 = x
    - n2 = 10/7
    - n3 = √2

Walkthrough of Call

- See online slideset with walkthrough of what happens during a method call