Java Primitive Data Types
Lecture 5

Borrowing from slides by Alan Hu, Kurt Eiselt, Paul Carter, and Tamara Munzner

Administrative News
- Demco Learning Center
  - ICICS/CS x150
  - Learning center is ramping up
  - Staffed during normal business hours
    - 10 am - 6 pm M-Th, 10 am - 4 pm F
  - TAs from all 1st year courses
  - Drop by if you have any questions
- WebCT bulletin board
  - Monitored by TAs, use to ask questions

Reading Assignments
- For this week, read
  - Edition 2 and 3: Ch 2.1-2.5, Ch 4.1-4.2
- For next week read
  - Ch 4.7 (both editions)
  - Ch 3 (both editions)
  - Note: updated reading!

Recap: Variables
- **Variable**: name for location in memory where data is stored
  - A little like variables in algebra class
- **pay_rate, hours_worked, a, b, and c** are all variables
- Variable names begin with lower case letters
  - Java convention, not compiler/syntax requirement
- Variable may be name of single byte in memory or may refer to a group of contiguous bytes
  - More about this later…

Recap: Variable Declaration
- variable declaration is instruction to compiler
  - reserve block of main memory large enough to store data type specified in declaration
- variable name is specified by identifier
- syntax:
  - `typeName variableName;`

Recap: Variables & Assignment
```java
// Test3.java  Author: Kurt
// Our third use of variables!
public class Test3
{
    public static void main (String[] args)
    {
        int a; // these
        int b; // are
        int c; // variable declarations
        b = 3; // these
        c = 5; // are
        a = b + c; // assignment statements
        System.out.println ("The answer is " + a);
    }
}
```
Recap: Assignment Statements
- Assignment statement assigns value to variable
- sometimes say binds value to variable
- Assignment statement is
  - identifier
  - followed by assignment operator (=)
  - followed by expression
  - followed by semicolon (;)
- Note that = is not a test for equality!

Recap: Assignment Statements
- Here’s an occasional point of confusion:
  ```java
  a = 7;       // what’s in a?
  b = a;       // what’s in b?
  System.out.println("a is "+a+"and b is "+b);
  a = 8;
  System.out.println("a is "+a+"and b is "+b);
  ```
- Memory locations a and b are distinct
  - value of a at that point assigned to b
  - changing a later does not affect previous copy

Recap: Data Types
- Java requires that we tell it what kind of data it is working with
- For every variable, we have to declare a data type
- Java language provides eight primitive data types
  - i.e. simple, fundamental
- For more complicated things, can use data types
  - created by others provided to us through the Java libraries
  - that we invent
  - More soon - for now, let’s stay with the primitives
- We want a, b, and c to be integers. Here’s how we do it...

Recap: Primitive Data Types: Numbers
<table>
<thead>
<tr>
<th>Type</th>
<th>Size</th>
<th>Min</th>
<th>Max</th>
</tr>
</thead>
<tbody>
<tr>
<td>byte</td>
<td>1 byte</td>
<td>-128</td>
<td>127</td>
</tr>
<tr>
<td>short</td>
<td>2 bytes</td>
<td>-32,768</td>
<td>32,767</td>
</tr>
<tr>
<td>int</td>
<td>4 bytes</td>
<td>-2,147,483,648</td>
<td>2,147,483,647</td>
</tr>
<tr>
<td>float</td>
<td>4 bytes</td>
<td>approx. 3.4E38 (7 sig. digits)</td>
<td>approx. 3.4E38 (7 sig. digits)</td>
</tr>
<tr>
<td>double</td>
<td>8 bytes</td>
<td>approx. 1.7E308 (15 sig. digits)</td>
<td>approx. 1.7E308 (15 sig. digits)</td>
</tr>
</tbody>
</table>

- Six primitives for numbers
  - fixed size, so finite capacity
  - integer vs. floating point

Recap: Non-numeric Primitive Types
- Character Type
  - named char
  - Java uses the Unicode character set so each char occupies 2 bytes of memory.
- Boolean Type
  - named boolean
  - Variables of type boolean have only two valid values
  - true and false
  - Often represents whether particular condition is true
  - Ignore these for now. We’ll see them again later.

Today’s Objectives
- Learn the Java primitive types
- Get comfortable with assignments
- Learn the most common Java operators for the primitive types
- Combine operators into simple expressions
- Understand some basic type conversions on Java numeric types.
**Data Type Sizes**

<table>
<thead>
<tr>
<th>Type</th>
<th>Size</th>
<th>Min</th>
<th>Max</th>
</tr>
</thead>
<tbody>
<tr>
<td>int</td>
<td>4 bytes</td>
<td>-2,147,483,648</td>
<td>2,147,483,647</td>
</tr>
<tr>
<td>float</td>
<td>8 bytes</td>
<td>approx -1.7E308</td>
<td>approx 1.7E308</td>
</tr>
</tbody>
</table>

- fixed size, so finite capacity

**Floating Point Numbers**

- significant digits
  - 42 = 4.2 x 10 = 4.2 x 10^1
  - 4.2 = 4.2 x 1 = 4.2 x 10^0
  - 42000000 = 4.2 x 10000000 = 4.2 x 10^7
  - 0.000042 = 4.2 x .00001 = 4.2 x 10^-5

**Primitive Data Types: Numbers**

<table>
<thead>
<tr>
<th>Type</th>
<th>Size</th>
<th>Min</th>
<th>Max</th>
</tr>
</thead>
<tbody>
<tr>
<td>byte</td>
<td>1 byte</td>
<td>-128</td>
<td>127</td>
</tr>
<tr>
<td>short</td>
<td>2 bytes</td>
<td>-32,768</td>
<td>32,767</td>
</tr>
<tr>
<td>int</td>
<td>4 bytes</td>
<td>-2,147,483,648</td>
<td>2,147,483,647</td>
</tr>
<tr>
<td>float</td>
<td>4 bytes</td>
<td>approx -3.4E38 (7 sig digits)</td>
<td>approx 3.4E38 (7 sig digits)</td>
</tr>
<tr>
<td>double</td>
<td>8 bytes</td>
<td>approx -1.7E308 (15 sig digits)</td>
<td>approx 1.7E308 (15 sig digits)</td>
</tr>
</tbody>
</table>

- Primary primitives are **int** and **double**
- Just worry about those for now
- Don’t need to memorize exact limits, but know roughly what the limits are.

**Data Types: Int and Double**

- int
  - integer
  - 4 bytes, about -2 billion to 2 billion
- double
  - real number
  - (double-precision floating point)
  - 8 bytes, 15 sig figs, humongous range
  - (Number systems briefly explained in Appendix L)
Variable Declaration Examples

- person's age in years
- height of mountain to nearest meter
- length of bacterium in centimeters
- number of pets at home

Expressions

- expression is combination of
  - one or more operators and operands
  - operator examples: +, *, /, ...
  - operand examples: numbers, variables, ...
  - usually performs a calculation
    - don't have to be arithmetic but often are
    - examples

Arithmetic Operators

- + addition
  - Works on int, double, byte, short, long, ...
- - subtraction
  - Works on all numeric types, too
- * multiplication
  - Didn't have times sign on keyboard
  - Works on all numeric types, too

More Arithmetic Operators

- / division
  - Integer division on integer types!
  - Example: 13 / 5 results in 2
  - Just like before you learned fractions
  - Normal division on double and float
  - % remainder (aka "mod")
    - Only works on integer types
    - Example: 13 % 5 results in 3

(Aside: Operator Overloading)

- Hmm... the same symbol / can do different things for ints and doubles:
  - 13/5 results in 2 (the type is int)
  - 13.0/5.0 results in 2.6 (the type is double)
- Similar for other operators, e.g., +
  - 13+5 is 18 (18 is an int)
  - 13.0+5.0 is 18.0 (18.0 is a double)
  - “13”+“5” is “135” ("135" is a String)!

Operator Precedence

- What does this expression evaluate to?
  - $7 + 2 \times 5$
Operator Precedence

- What does this expression evaluate to?
  \[ 7 + 2 \times 5 \]
- Multiplication has higher operator precedence than addition (just like in algebra)

<table>
<thead>
<tr>
<th>precedence</th>
<th>operator</th>
<th>operation</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 higher</td>
<td>+, -</td>
<td>unary plus and minus</td>
</tr>
<tr>
<td>2</td>
<td>* / %</td>
<td>multiply, divide, remainder</td>
</tr>
<tr>
<td>3 lower</td>
<td>+, -</td>
<td>add, subtract</td>
</tr>
</tbody>
</table>

Operator Precedence

- What does this expression evaluate to?
  \[ 7 + 2 \times 5 \]
- Multiplication has higher operator precedence than addition (just like in algebra)

Use parentheses to change precedence order or just clarify intent

\[(7 + 2) \times 5 \quad 7 + (2 \times 5)\]

Associativity

- What about this?
  \[ 7 - 5 - 3 \]

The result is -1

- (7 – 5) – 3, not 7 – (5 – 3)
- Arithmetic operators of same precedence are left associative
- Matters for some operators; doesn’t for others
- Use parentheses to be clear!
- (Operators and precedence in Appendix F)

Converting Between Types

- Which of these are legal?
  - `int shoes = 2;`
  - `double socks = 1.75;`
  - `double socks = 1;`
  - `int shoes = 1.5;`

Integers are subset of reals
- but reals are not subset of integers
- Java automatically converts `int` to `double` when needed (or smaller size to larger size)
Casting

- **Casting**: force Java to convert from one type to another, even with information loss
- Converting from real to integer
  - `int shoes = (int) 1.5;`
- Truncation: fractional part thrown away
  - `int shoes = (int) 1.75;`
  - `int shoes = (int) 1.25;`
- Rounding: must be done explicitly
  - `shoes = Math.round(1.99);`

Converting Between Types

```java
//*****************************************
// Feet.java   Author: Tamara
// What type of things can be put on feet?
//*****************************************
public class Feet
{
    public static void main (String[] args)
    {
        int shoes = 2;
        int socks = (int) 1.75;
        System.out.println("shoes = " + shoes + " socks = " +
        socks);
        int toes = Math.round(1.99);
        System.out.println("toes = " + toes);
    }
}
```

- What's wrong?

Primitive Data Types: Numbers

<table>
<thead>
<tr>
<th>Type</th>
<th>Size</th>
<th>Min</th>
<th>Max</th>
</tr>
</thead>
<tbody>
<tr>
<td>byte</td>
<td>1 byte</td>
<td>-128</td>
<td>127</td>
</tr>
<tr>
<td>short</td>
<td>2 bytes</td>
<td>-32,768</td>
<td>32,767</td>
</tr>
<tr>
<td>int</td>
<td>4 bytes</td>
<td>-2,147,483,648</td>
<td>2,147,483,647</td>
</tr>
<tr>
<td>float</td>
<td>4 bytes</td>
<td>approx -3.4E38 (7 sig digits)</td>
<td>approx 3.4E38 (7 sig digits)</td>
</tr>
<tr>
<td>double</td>
<td>8 bytes</td>
<td>approx -1.7E308 (15 sig digits)</td>
<td>approx 1.7E308 (15 sig digits)</td>
</tr>
</tbody>
</table>

- Primary primitives are `int` and `double`
- three other integer types
- one other real type

Or Tell Java with a Type Cast

```java
//*****************************************
// Feet2.java   Author: Tamara
// What type of things can be put on feet?
//*****************************************
public class Feet2
{
    public static void main (String[] args)
    {
        int shoes = 2;
        int socks = (int) 1.75;
        System.out.println("shoes = " + shoes + " socks = " +
        socks);
        int toes = (int) Math.round(1.99);
        System.out.println("toes = " + toes);
    }
}
```