Recap: White Space

- White space
  - Blanks between identifiers and other symbols
  - Tab and newline characters are included

- White space does not affect how program runs

- Use white space to format programs we create so they're easier for people to understand

Recap: Floating Point Numbers

- significant digits
- \( 42 = 4.2 \times 10 = 4.2 \times 10^1 \)
- \( 4.2 = 4.2 \times 1 = 4.2 \times 10^0 \)
- \( 42000000 = 4.2 \times 10000000 = 4.2 \times 10^7 \)
- only need to remember nonzero digits
- where to put the decimal point
- floats around when multiply/divide by 10

Recap: Errors

- Compile-time errors
  - syntax/structure
- Run-time errors
- Logical errors
  - semantics/meaning

Assignment

- 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:
  - \( \text{typeName variableName;} \)

Assignment Statements

- Java first computes value on right side
- Then assigns value to variable given on left side
- Old value will be overwritten if variable was assigned before

Here's an occasional point of confusion:

\[
\begin{align*}
    &a = 7; \quad // \text{what's in } a \\
    &b = a; \quad // \text{what's in } b \\
    &\text{weekly_pay} = \text{pay_rate} * \text{hours_worked}; \quad // \text{what's in } \text{weekly_pay now?}
\end{align*}
\]
Assignment Statements

Here's an occasional point of confusion:

a = 7; // what's in a?
b = a; // what's in b?
System.out.println("a is " + a + " b is " + b);

Variable values on left of = are clobbered

Variable values on right of = are unchanged

copy of value assigned to a also assigned to b
but that doesn't change value assigned to a

Assignment Statements

Here's an occasional point of confusion:

a = 8; // what's in a?
b = a; // what's in b?
System.out.println("a is " + a + " b is " + b);

Memory locations a and b are distinct

copy of value assigned to a also assigned to b
changing a later does not affect previous copy

more later

Variable Declaration and Assignment

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

tax:  

typeName variableName;

typeName variableName = value;

can declare and assign in one step

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

3

7 + 2

7 + 2 * 5

(7 + 2) * 5

Operator Precedence

What does this expression evaluate to?
7 + 2 * 5

Multiplication has higher operator precedence than
addition (just like in algebra)

preference operand operation
1 higher +, % unary plus and minus
2 * / % multiply, divide, remainder
3 lower +, - add, subtract

Use parentheses to change precedence order or just
clarify intent

(7 + 2) * 5
7 + (2 * 5)

Converting Between Types

Which of these are legal?

int shoes = 2;
double socks = 1.75;
double socks = 1;
int shoes = 1.5;

Converting Between Types

Which of these are legal?

int shoes = 2;
double socks = 1.75;
double socks = 1;
int shoes = 1.5;

Casting

Casting: convert from one type to another 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);

Data Types Sizes

Type Size Min Max
int 4 bytes approx -2,147,483,648 approx 2,147,483,647
float 4 bytes approx -3.4E38 (7 sig digits) approx 3.4E38 (7 sig digits)
double 8 bytes approx -1.7E308 (15 sig digits) approx 1.7E308 (15 sig digits)

is there more to life than 4-byte ints or 8-byte doubles?

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>approx -2,147,483,648</td>
<td>approx 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

range of choices for storage capacity

Using Long Integers

// Fest2.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);
        long toes = Math.round(1.99);
        System.out.println("toes = " + toes);
    }
}

Or Cast To Int

// Fest3.java Author: Tamara
// What type of things can be put on feet?
//+++++++++++++++++++++++++++++++++++++++++++++++
public class Feet3 {
    public static void main (String[] args) {
        int shoes = 2;
        int socks = (int) 1.75;
        System.out.println("shoes = " + shoes + " socks = " + socks);
        long toes = Math.round(1.99);
        System.out.println("toes = " + toes);
    }
}

Primitive Data Types: Non-numeric

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

more generally represents any data that has two states

yes/no, on/off
### What Changes, What Doesn’t?

//*****************************************
// Vroom.java Author: Tamara
// Playing with constants
//*****************************************

```java
public class Vroom {
    public static void main (String[] args) {
        double lightYears, milesAway;
        lightYears = 4.35; // to Alpha Centauri
        System.out.println("lightYears: "+ lightYears + " milesAway "+ milesAway);
        lightYears = 68; // to Aldebaran
        System.out.println("lightYears: " + lightYears + " milesAway "+ milesAway);
    }
}
```

### Constants

- **Things that do not vary**
  - unlike variables
  - will never change

**Syntax:**
- `final typeName variableName;`
- `final typeName variableName = value;`
- Constant names in all upper case
  - Java convention, not compiler/syntax requirement

### Programming With Constants

```java
public static void main (String[] args) {
    double lightYears, milesAway;
    final int LIGHTSPEED = 186000;
    final int SECONDS_PER_YEAR = 60*60*24*365;
    final double ALPHACENT_DIST = 4.35; // to Alpha Centauri
    final double ALDEBARAN_DIST = 68; // to Aldebaran
    lightYears = ALPHACENT_DIST;
    milesAway = lightYears * LIGHTSPEED * SECONDS_PER_YEAR;
    System.out.println("lightYears: " + lightYears + " miles " + milesAway);
    lightYears = ALDEBARAN_DIST;
    milesAway = lightYears * LIGHTSPEED * SECONDS_PER_YEAR;
    System.out.println("lightYears: " + lightYears + " miles " + milesAway);
}
```

### Avoiding Magic Numbers

- **magic numbers:** numeric constants directly in code
  - almost always bad idea!
  - hard to understand code
  - hard to make changes
  - typos possible
  - use constants instead

### Reading Next Week

- Rest of Chap 2
  - 2.3-4, 2.6-2.10
- Rest of Chap 4
  - 4.3-4.7