CS 112 Introduction to Programming

Java Primitive Data Types; Arithmetic Expressions

Yang (Richard) Yang
Computer Science Department
Yale University
208A Watson, Phone: 432-6400
Email: yry@cs.yale.edu
Preston, do you consider programming more of an art or a science?

Quiet! I'm trying to cut and paste 300 lines of code into 7 different places!

Never mind.
Questions on pset 1?
Recap: Java Static Methods

- Why methods?
  - Denote structure of a program
  - Allow code reuse to eliminate redundancy

- Syntax: defining methods and invoking methods

- Basic method design/impl. methodology
  - Top-down decomposition/bottom-up implementation
Recap: Static Method Example

- Write a program to print these figures.

```java
public class Figures1 {
    public static void main(String[] args) {
        System.out.println("\n\n+++++++");
        System.out.println("\n\n\n\n+++++++");
        System.out.println("\n\n\n\n+++++++");
        System.out.println("\n\n\n\n+++++++");
        System.out.println("\n\n\n\n+++++++");
        System.out.println("\n\n\n\n+++++++");
        System.out.println("\n\n\n\n+++++++");
        System.out.println("\n\n\n\n+++++++");
        System.out.println("\n\n\n\n+++++++");
    }
}
```
Recap: Decomposition Example
Foundational Programming Concepts

Any program you might want to write

- Objects
- Methods and classes
- Graphics, sound, and image I/O
- Arrays
- Conditionals and loops
- Math
- Text I/O
- Primitive data types
- Assignment statements
Outline

- Admin and recap
- Primitive data types
  - why data types
Memory

RAM is divided into many cells; each cell can be identified by a numeric address.

Each memory cell has a set number of bits (usually 8 bits, or one byte); a bit can represent 2 values of 0 or 1.

A computer can use multiple cells (e.g., 2 bytes) to store a value.

- How many possible values can a byte represent?
- How many possible values can 2 bytes represent?

Primary storage area for programs and data.

Also called RAM.

Main Memory.

Primary storage area for programs and data.

Each memory cell has a set number of bits (usually 8 bits, or one byte); a bit can represent 2 values of 0 or 1.

A computer can use multiple cells (e.g., 2 bytes) to store a value.

- How many possible values can a byte represent?
- How many possible values can 2 bytes represent?
Variable

RAM is divided into many cells; each cell can be identified by a numeric address.

In high-level programming, instead of a numerical address, you can refer to a memory location by a name, say x. This is called a variable.
Problem

- What does the number (combination) stored at a given memory location represent?

```
  00110000
```
### Problem: How can the computer tell what `00110000` stands for: a character or number?

<table>
<thead>
<tr>
<th>Dec</th>
<th>Hex</th>
<th>Char</th>
<th>Dec</th>
<th>Hex</th>
<th>Char</th>
<th>Dec</th>
<th>Hex</th>
<th>Char</th>
<th>Dec</th>
<th>Hex</th>
<th>Char</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>00</td>
<td>Null</td>
<td>32</td>
<td>20</td>
<td>Space</td>
<td>64</td>
<td>40</td>
<td></td>
<td>96</td>
<td>60</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>01</td>
<td>Start of heading</td>
<td>33</td>
<td>21</td>
<td>!</td>
<td>65</td>
<td>41</td>
<td>A</td>
<td>97</td>
<td>61</td>
<td>a</td>
</tr>
<tr>
<td>2</td>
<td>02</td>
<td>Start of text</td>
<td>34</td>
<td>22</td>
<td>&quot;</td>
<td>66</td>
<td>42</td>
<td>B</td>
<td>98</td>
<td>62</td>
<td>b</td>
</tr>
<tr>
<td>3</td>
<td>03</td>
<td>End of text</td>
<td>35</td>
<td>23</td>
<td>#</td>
<td>67</td>
<td>43</td>
<td>C</td>
<td>99</td>
<td>63</td>
<td>c</td>
</tr>
<tr>
<td>4</td>
<td>04</td>
<td>End of transmit</td>
<td>36</td>
<td>24</td>
<td>$</td>
<td>68</td>
<td>44</td>
<td>D</td>
<td>100</td>
<td>64</td>
<td>d</td>
</tr>
<tr>
<td>5</td>
<td>05</td>
<td>Enquiry</td>
<td>37</td>
<td>25</td>
<td>%</td>
<td>69</td>
<td>45</td>
<td>E</td>
<td>101</td>
<td>65</td>
<td>e</td>
</tr>
<tr>
<td>6</td>
<td>06</td>
<td>Acknowledge</td>
<td>38</td>
<td>26</td>
<td>&amp;</td>
<td>70</td>
<td>46</td>
<td>F</td>
<td>102</td>
<td>66</td>
<td>f</td>
</tr>
<tr>
<td>7</td>
<td>07</td>
<td>Audible bell</td>
<td>39</td>
<td>27</td>
<td>'</td>
<td>71</td>
<td>47</td>
<td>G</td>
<td>103</td>
<td>67</td>
<td>g</td>
</tr>
<tr>
<td>8</td>
<td>08</td>
<td>Backspace</td>
<td>40</td>
<td>28</td>
<td>(</td>
<td>72</td>
<td>48</td>
<td>H</td>
<td>104</td>
<td>68</td>
<td>h</td>
</tr>
<tr>
<td>9</td>
<td>09</td>
<td>Horizontal tab</td>
<td>41</td>
<td>29</td>
<td>)</td>
<td>73</td>
<td>49</td>
<td>I</td>
<td>105</td>
<td>69</td>
<td>i</td>
</tr>
<tr>
<td>10</td>
<td>0A</td>
<td>Line feed</td>
<td>42</td>
<td>2A</td>
<td>*</td>
<td>74</td>
<td>4A</td>
<td>J</td>
<td>106</td>
<td>6A</td>
<td>j</td>
</tr>
<tr>
<td>11</td>
<td>0B</td>
<td>Vertical tab</td>
<td>43</td>
<td>2B</td>
<td>+</td>
<td>75</td>
<td>4B</td>
<td>K</td>
<td>107</td>
<td>6B</td>
<td>k</td>
</tr>
<tr>
<td>12</td>
<td>0C</td>
<td>Form feed</td>
<td>44</td>
<td>2C</td>
<td>,</td>
<td>76</td>
<td>4C</td>
<td>L</td>
<td>108</td>
<td>6C</td>
<td>l</td>
</tr>
<tr>
<td>13</td>
<td>0D</td>
<td>Carriage return</td>
<td>45</td>
<td>2D</td>
<td>-</td>
<td>77</td>
<td>4D</td>
<td>M</td>
<td>109</td>
<td>6D</td>
<td>m</td>
</tr>
<tr>
<td>14</td>
<td>0E</td>
<td>Shift out</td>
<td>46</td>
<td>2E</td>
<td>.</td>
<td>78</td>
<td>4E</td>
<td>N</td>
<td>110</td>
<td>6E</td>
<td>n</td>
</tr>
<tr>
<td>15</td>
<td>0F</td>
<td>Shift in</td>
<td>47</td>
<td>2F</td>
<td>/</td>
<td>79</td>
<td>4F</td>
<td>O</td>
<td>111</td>
<td>6F</td>
<td>o</td>
</tr>
<tr>
<td>16</td>
<td>10</td>
<td>Data link escape</td>
<td>48</td>
<td>30</td>
<td>0</td>
<td>80</td>
<td>50</td>
<td>P</td>
<td>112</td>
<td>70</td>
<td>p</td>
</tr>
<tr>
<td>17</td>
<td>11</td>
<td>Device control 1</td>
<td>49</td>
<td>31</td>
<td>1</td>
<td>81</td>
<td>51</td>
<td>Q</td>
<td>113</td>
<td>71</td>
<td>q</td>
</tr>
<tr>
<td>18</td>
<td>12</td>
<td>Device control 2</td>
<td>50</td>
<td>32</td>
<td>2</td>
<td>82</td>
<td>52</td>
<td>R</td>
<td>114</td>
<td>72</td>
<td>r</td>
</tr>
<tr>
<td>19</td>
<td>13</td>
<td>Device control 3</td>
<td>51</td>
<td>33</td>
<td>3</td>
<td>83</td>
<td>53</td>
<td>S</td>
<td>115</td>
<td>73</td>
<td>s</td>
</tr>
<tr>
<td>20</td>
<td>14</td>
<td>Device control 4</td>
<td>52</td>
<td>34</td>
<td>4</td>
<td>84</td>
<td>54</td>
<td>T</td>
<td>116</td>
<td>74</td>
<td>t</td>
</tr>
<tr>
<td>21</td>
<td>15</td>
<td>Neg. acknowledge</td>
<td>53</td>
<td>35</td>
<td>5</td>
<td>85</td>
<td>55</td>
<td>U</td>
<td>117</td>
<td>75</td>
<td>u</td>
</tr>
<tr>
<td>22</td>
<td>16</td>
<td>Synchronous idle</td>
<td>54</td>
<td>36</td>
<td>6</td>
<td>86</td>
<td>56</td>
<td>V</td>
<td>118</td>
<td>76</td>
<td>v</td>
</tr>
<tr>
<td>23</td>
<td>17</td>
<td>End trans. block</td>
<td>55</td>
<td>37</td>
<td>7</td>
<td>87</td>
<td>57</td>
<td>W</td>
<td>119</td>
<td>77</td>
<td>w</td>
</tr>
<tr>
<td>24</td>
<td>18</td>
<td>Cancel</td>
<td>56</td>
<td>38</td>
<td>8</td>
<td>88</td>
<td>58</td>
<td>X</td>
<td>120</td>
<td>78</td>
<td>x</td>
</tr>
<tr>
<td>25</td>
<td>19</td>
<td>End of medium</td>
<td>57</td>
<td>39</td>
<td>9</td>
<td>89</td>
<td>59</td>
<td>Y</td>
<td>121</td>
<td>79</td>
<td>y</td>
</tr>
<tr>
<td>26</td>
<td>1A</td>
<td>Substitution</td>
<td>58</td>
<td>3A</td>
<td></td>
<td>90</td>
<td>5A</td>
<td>Z</td>
<td>122</td>
<td>7A</td>
<td>z</td>
</tr>
<tr>
<td>27</td>
<td>1B</td>
<td>Escape</td>
<td>59</td>
<td>3B</td>
<td></td>
<td>91</td>
<td>5B</td>
<td>[</td>
<td>123</td>
<td>7B</td>
<td>{</td>
</tr>
<tr>
<td>28</td>
<td>1C</td>
<td>File separator</td>
<td>60</td>
<td>3C</td>
<td>&lt;</td>
<td>92</td>
<td>5C</td>
<td>\</td>
<td>124</td>
<td>7C</td>
<td></td>
</tr>
<tr>
<td>29</td>
<td>1D</td>
<td>Group separator</td>
<td>61</td>
<td>3D</td>
<td>=</td>
<td>93</td>
<td>5D</td>
<td>]</td>
<td>125</td>
<td>7D</td>
<td>)</td>
</tr>
<tr>
<td>30</td>
<td>1E</td>
<td>Record separator</td>
<td>62</td>
<td>3E</td>
<td>&gt;</td>
<td>94</td>
<td>5E</td>
<td>^</td>
<td>126</td>
<td>7E</td>
<td>~</td>
</tr>
<tr>
<td>31</td>
<td>1F</td>
<td>Unit separator</td>
<td>63</td>
<td>3F</td>
<td>?</td>
<td>95</td>
<td>5F</td>
<td></td>
<td>127</td>
<td>7F</td>
<td>□</td>
</tr>
</tbody>
</table>
**Type System**

- **type**: A category or set of values and operations defined on those values.
  - By specifying the type of a memory location, we know what the values represent.
- Many languages ask the programmer to specify types.
  - Examples: integer, real number, character.
**Variable and Type**

- **Variable**: A piece of the computer's memory that is given a name and a type to store value of the type.

- **Steps for using a variable**:
  - Declare it - state its name and type
  - Assign value - initialize or update its value
  - Use it - print it or use it as part of an expression
Primitive Data Types

- There are eight (simple) primitive data types in Java
  - six numerical types (e.g., `int`, `double`)
    - for mathematical calculation
  - characters
    - for text processing
  - Boolean (logical) values
    - for decision making
Declaration

- **Variable declaration**: Sets aside memory for storing a value.
  - Variables must be declared before they can be used.

- **Syntax**:
  \[
  \text{<type> <name>;} \\
  \]

  - `int x;`
  - `double myGPA;`
Assignment

- **Assignment**: Stores a value into a variable.
  - The value can be an expression; the variable stores its result.

- **Syntax**:
  
  $\langle\text{name}\rangle = \langle\text{expression}\rangle$;

- **Examples**:
  
  - `int x;
    x = 3;`  
    - $x$ | 3
  
  - `double myGPA;
    myGPA = 1.0 + 2.25;`  
    - `myGPA` | 3.25

- A variable can only store a value of its own type.
Outline

- Admin and recap
- Primitive data types
  - why data types?
  - storage and representation
The differences among the various numeric primitive types are their storage sizes and representation format, and hence the ranges & precision of the values they can store.
**Integer Numeric Data Types**

Different integer numeric data types have different *ranges* and *precision*.

<table>
<thead>
<tr>
<th>Type</th>
<th>Storage</th>
<th>Min Value</th>
<th>Max Value</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>long</td>
<td>8 bytes</td>
<td>&lt; -9 \times 10^{18}</td>
<td>&gt; 9 \times 10^{18}</td>
</tr>
</tbody>
</table>
Real Numeric Data Types

Question: can computer store all real numbers in a range?

- Represented using the IEEE 754 format
  - with limited # of precision bits
  - See Precision.java
All Numeric Data Types

- Different integer numeric data types have different **ranges** and **precision**

<table>
<thead>
<tr>
<th>Type</th>
<th>Storage</th>
<th>Min Value</th>
<th>Max Value</th>
<th>IEEE 754 format</th>
</tr>
</thead>
<tbody>
<tr>
<td>byte</td>
<td>1 byte</td>
<td>-128</td>
<td>127</td>
<td>numbers with no fractional part</td>
</tr>
<tr>
<td>short</td>
<td>2 bytes</td>
<td>-32,768</td>
<td>32,767</td>
<td></td>
</tr>
<tr>
<td>int</td>
<td>4 bytes</td>
<td>-2,147,483,648</td>
<td>2,147,483,647</td>
<td></td>
</tr>
<tr>
<td>long</td>
<td>8 bytes</td>
<td>&lt; -9 x 10^{18}</td>
<td>&gt; 9 x 10^{18}</td>
<td></td>
</tr>
<tr>
<td>float</td>
<td>4 bytes</td>
<td>+/- 3.4 x 10^{38} with 7 significant digits</td>
<td></td>
<td></td>
</tr>
<tr>
<td>double</td>
<td>8 bytes</td>
<td>+/- 1.7 x 10^{308} with 15 significant digits</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Java Numerical Value and Type

- Java is a strongly typed language, i.e., every data item has a type
  - An integer literal is by default of type `int`
    - That is, a literal number 4 in Java is of type `int`
    - To say that the number 4 is of type `long`, write `4l` or `4L` (4L is preferred over 4l since lower case “l” is hard to distinguish from 1)
  - A real (floating point) literal (e.g., -1.23 6.12e23) is by default of type `double`
    - To say that the number 0.1 is of type `float`, write 0.1f or 0.1F
Questions

Question: to represent the number of students at Yale, which numeric data type variable do you use?

Question: to represent the world population, which numeric data type variable do you use?

Question: to represent your GPA, which numeric data type variable do you use?

Question: to represent a person’s height in meters, which numeric data type variable do you use?

Question: to represent pi as 3.14159265359, which numeric data type variable do you use?
Real Life Example: Ariane 5

- Historical example: Ariane 5 explosion in 1996
  (http://www.youtube.com/watch?v=kYUrqdUyEpI;
  http://www.ima.umn.edu/~arnold/disasters/ariane.html)
Real Life Example: Ariane 5

- Historical example: Ariane 5 explosion in 1996
  (http://www.youtube.com/watch?v=kYUrqdUyEpI;
  http://www.ima.umn.edu/~arnold/disasters/ariane.html)
Real Life Example: Ariane 5


- Reason: range error
  - trying to store a 64-bit real number (a double) to a 16-bit integer led to the crash
Real Life Example: Patriot Failure

- The Patriot Missile Failure in 1991
  - Perfect detection of a Scud missile, but the intercepting Patriot missed the target
- Reason: precision error
  - A computer cannot represent 0.1 precisely; for a 24-bit floating point number they used, it is off by 0.000000095.
  - After 100 hours in operation, it is off by 0.34 seconds (=0.000000095*100 hours * 60 min/hour * 60 sec/min * 10), leading to an error of about 600 meters ([http://www.ima.umn.edu/~arnold/disasters/patriot.html](http://www.ima.umn.edu/~arnold/disasters/patriot.html))
In the Movie

http://www.youtube.com/watch?v=G_wiXgRWrIU
Characters

- A `char` is a single character from a character set.
- A `character set` is an ordered list of characters; each character is given a unique number.
- Character literals are represented in a program by delimiting with single quotes:

  `'a'  'X'  '7'  '$'  ','  '\n'`
Java Character Set

- Java uses the Unicode character set, a superset of ASCII
  - Uses sixteen bits (2 bytes) per character, allowing for 65,536 unique characters
  - It is an international character set, containing symbols and characters from many languages
  - Code chart can be found at: http://www.unicode.org/charts/
Boolean

- A boolean value represents logical value: true or false

- The keywords true and false are the only valid values for a boolean type

- A boolean can also be used to represent any two states, such as a light bulb being on or off
Outline

- Admin and recap
- Primitive data types
  - why data types?
  - storage and representation
  - operations
Data Type and Operations

A type defines not only the storage/representation but also the allowed and meaning (semantics) of operations

- Discussions: reasonable operations that can be performed on two operands
  - Integers: \( i_1 \ ? \ i_2 \)
  - Strings: \( s_1 \ ? \ s_2 \)
  - Characters: \( c_1 \ ? \ c_2 \)
# Data Type and Operations

<table>
<thead>
<tr>
<th>type</th>
<th>set of values</th>
<th>literal values</th>
<th>operations</th>
</tr>
</thead>
<tbody>
<tr>
<td>char</td>
<td>characters</td>
<td>'A', '@'</td>
<td>compare (more details later on +-), equal, not</td>
</tr>
<tr>
<td>String</td>
<td>sequences of characters</td>
<td>&quot;Hello&quot;, &quot;112 is fun&quot;</td>
<td>concatenate +, equal, not</td>
</tr>
<tr>
<td>int</td>
<td>integers</td>
<td>17, 12345</td>
<td>compare add +, sub -, multiply *, divide /, modulus %</td>
</tr>
<tr>
<td>double</td>
<td>floating-point numbers</td>
<td>3.1415, 6.022e23</td>
<td>compare add +, sub -, multiply *, divide /, modulus %</td>
</tr>
<tr>
<td>boolean</td>
<td>truth values</td>
<td>true, false</td>
<td>equal, not, and, or, not</td>
</tr>
</tbody>
</table>
Data Type and Operations

- Most operations (+, -, *, /) are intuitive and similar to our daily-life use.
- Perhaps a first major surprise in learning programming is that the result of an operation depends on the data type.

```
3 + 5       v.s.       “3” + “5”
3 / 5       v.s.       3.0 / 5.0
```

See TypeDep.java
Interpretation

You should think that there are multiple versions of the same operator, each for a type, e.g.,

• +int +string …
• /int /double …
**Integer Division with /**

- When we divide integers, the result is an integer (the fractional part is discarded)
  - $14 \div 4$ is $3$, not $3.5$

```
  3
4 ) 14
  10
  4
```

- $32 \div 5$ is $6$
- $8 \div 10$ is $0$
- $156 \div 100$ is $1$

- Dividing by 0 causes an error when your program runs.
The % operator computes the remainder from integer division.

- \( 14 \div 4 \) is 2
- \( 218 \div 5 \) is 3

Practice (offline):

- \( 45 \div 6 \) is 7
- \( 2 \div 2 \) is 0
- \( 8 \div 20 \) is 4
- \( 11 \div 0 \)

Obtain last digit of a number: \( 230857 \div 10 \) is 7

Obtain last 4 digits: \( 230857 \div 10000 \) is 857

See whether a number is odd: \( 7 \div 2 \) is 1, \( 42 \div 2 \) is 0
Outline

- Admin and recap
- Primitive data types
  - why data types?
  - storage and representation
  - operations
  - expressions
Evaluating Arithmetic Expression

- Arithmetic operators can be combined into complex arithmetic expressions
  - $(7 + 2) \times 6 / 3$

- The evaluation order of the operators in an arithmetic expression is determined by a well-defined precedence order
  - Remember?
    - *Pretty Please My Dear Aunt Sally*
Operator Precedence Rules

- Generally operators evaluate left-to-right.
  \[ 1 - 2 - 3 \] is \((1 - 2) - 3\) which is \(-4\)

- But \(*\ / \%\) have a higher level of precedence than \(+\ -\)
  \[ 1 - 3 * 4 \] is \(-11\)

- Parentheses can force a certain order of evaluation:
  \[ (1 + 3) * 4 \] is \(16\)

- Spacing does not affect order of evaluation
  \[ 1+3 * 4-2 \] is \(11\)
What is the order of evaluation in the following expressions?

1. \( a + b + c + d + e \)
   - 1 2 3 4

2. \( a + b \times c - d \div e \)
   - 3 1 4 2

3. \( a \div (b + c) - d \mod e \)
   - 2 1 4 3

4. \( a \div (b \times (c + (d - e))) \)
   - 4 3 2 1
Precedence: Examples (Offline)

\[
1 \times 2 + 3 \times 5 \% 4
\]

\[
\frac{2}{2} + \frac{15}{3} \% 4
\]

\[
\frac{2}{2} + 3
\]

\[
\frac{5}{5}
\]

\[
1 + 8 \div 3 \times 2 - 9
\]

\[
1 + \frac{2}{2} \times 2 - 9
\]

\[
1 + \frac{4}{4} - 9
\]

\[
\frac{5}{5} - 9
\]

\[
-4
\]
Precedence Questions (Offline)

What values result from the following expressions?

- 9 / 5
- 695 % 20
- 7 + 6 * 5
- 7 * 6 + 5
- 248 % 100 / 5
- 6 * 3 - 9 / 4
- (5 - 7) * 4
- 6 + (18 % (17 - 12))
Real Number Example (Offline)

\[ 2.0 \times 2.4 + 2.25 \times 4.0 / 2.0 \]

\[ 4.8 + 2.25 \times 4.0 / 2.0 \]

\[ 4.8 + 9.0 / 2.0 \]

\[ 4.8 + 4.5 \]

\[ 9.3 \]
Problem

- Sometimes it is more efficient and natural to represent data as one type, but during a computation, we may want to get desired result in a different type.
  - e.g., raw grade points and # of grades as integers, but GPA as double (see GPA.java)
Backup Slides
Why Data Conversion

- Sometimes it is more efficient and natural to represent data as one type, but during a computation, we may want to get *desired* result in a different type.
  - e.g., raw grade points and # of grades as integers, but GPA as double (see GPA.java)
Why Data Conversion

- Sometimes it is more efficient and natural to represent data as one type, but during a computation, we may want to get desired result in a different type
  - e.g., raw grade points and # of grades as integers, but GPA as double (see GPA.java)

- Sometimes we just write mixed-type expressions
  - 4.0 / 8 (Q: is the value 0 or 0.5?)
Outline

- Admin and recap
- Data conversion
  - why?
  - what is data conversion
Data Conversion

- Data conversion is the conversion of data from one type to another, e.g.,
  - an int → a double,
  - a double → an int,
  - an int → a string

- Java data conversion is per-operator, occurring when the operator is evaluated according to the precedence rule

- Java has two types of data conversion
  - Implicit (automatic/predefined) data conversion
  - Explicit (cast) data conversion
(Implicit) Predefined Data Conversion

- Seeing a mixed operation, Java tries a set of **predefined data conversion rules**
  - If successful, you get the results
  - If not, you get a compiler error
Predefined Data Conversion Rule:

Arithmetic (numeric) Promotion

- **Occurs automatically** when the operands of a binary arithmetic operator are of different types
  - if either operand is `double`, the other is converted to `double`
  - otherwise, if either operand is `float`, the other is converted to `float`
  - otherwise, if either operand is `long`, the other is converted to `long`
  - otherwise, both operands are converted to `int`

Examples:
- `4.0 / 8` (which `/` is it: `/double`, `/float`, `/int`)
- `4 / 8.0` (which `/` is it: `/double`, `/float`, `/int`)
- `4 / 8` (which `/` is it: `/double`, `/float`, `/int`)
Example: Mixed Arithmetic Expression

\[
2.5 + \frac{10}{3} \times 2.5 - \frac{6}{4}
\]

\[
2.5 + \frac{3}{\underline{2.5}} - \frac{6}{4}
\]

\[
2.5 + \frac{7.5}{2.5} - \frac{6}{4}
\]

\[
2.5 + \frac{7.5}{\underline{1}}
\]

\[
\frac{10.0}{\underline{1}}
\]

\[
\underline{9.0} \text{ (not 9!)}
\]
Practice: Mixed Arithmetic Expression (Offline)

\[
\frac{7}{3} \times 1.2 + \frac{3}{2}
\]

\[
2 \times 1.2 + \frac{3}{2}
\]

\[
2.4 + \frac{3}{2}
\]

\[
2.4 + 1
\]

\[
3.4
\]
Example: Mixed Assignment

- Automatic mixed type assignment allowed only if allowed by automatic numeric promotion

```c
int x;
x = 2.5; // ERROR: incompatible types
```

```c
double myGPA = 4;
double avg = 11 / 2;
```

<table>
<thead>
<tr>
<th></th>
<th>myGPA</th>
<th>4.0</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>avg</td>
<td>5.0</td>
</tr>
</tbody>
</table>
Predefined Data Conversion Rule: Numeric to Java String

- Occurs *automatically* when one operand is a number and the other a string in the “+” operator

- The conversion is per-operator, affecting only its operands.

- This produces the convenient string concatenation operation.
Java String Concatenation
Conversion: Examples

1 + "abc" + 2 is "1abc2"
"abc" + 1 + 2 is "abc12"
1 + 2 + "abc" is "3abc"
"abc" + 9 * 3 + 1 is "abc271"
4 - 1 + "abc" is "3abc"
Examples

- See IntOps.java
- Fix the GPA.java program
User Forced (Explicit) Conversion: Type Casting

- **type cast**: An explicit, **FORCED** conversion from one type to another.

- **Syntax**:
  
  \[(\text{type}) \text{ expression}\]

- Type casting has **high precedence** and casts only the item immediately next to it.

- You can cast either up (promotion) or down (truncate)
Type Casting Examples

double result = (double) 19 / 5;  // 3.8

int result2 = (int) result;  // 3

double x = (double) 1 + 1 / 2;  // 1.0

double y = 1 + (double) 1 / 2;  // 1.5