## Sizes of Primitive Types

| Type   | Size | Size
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>byte</td>
<td>8 bits</td>
<td>8 bits</td>
</tr>
<tr>
<td>short</td>
<td>8 bits</td>
<td>8 bits</td>
</tr>
<tr>
<td>int</td>
<td>8 bits</td>
<td>8 bits</td>
</tr>
<tr>
<td>long</td>
<td>8 bits</td>
<td>8 bits</td>
</tr>
<tr>
<td>float</td>
<td>8 bits</td>
<td>8 bits</td>
</tr>
<tr>
<td>double</td>
<td>8 bits</td>
<td>8 bits</td>
</tr>
<tr>
<td>char</td>
<td>8 bits</td>
<td>8 bits</td>
</tr>
</tbody>
</table>
BINARY CODES

Inside the computer, every piece of information is either a 0 or a 1.
These can be represented electrically or magnetically.

A single 0/1 value is a bit.
- It represents a single, simple piece of information
  - Yes/No
  - True/False
  - On/Off

A byte in a computer is a group of eight bits.
- The collection of eight bits can represent significant information.
- If one bit can represent two things, two bits can represent four things.
  - 0 and 0, 0 and 1, 1 and 0, or 1 and 1.
- Some number $n$ of bits can represent $2^n$ different things.
- So, a byte can represent $2^8$ different things.
  - $2^8 = 256$. 
NUMBER SYSTEMS

• Binary is just a number system with a base of 2.
• What base number system do we commonly use?
  • Decimal (10)
• What other base number systems are common?
  • Octal (8)
    • UNIX Permissions
  • Hexadecimal (16)
    • 0 1 2 3 4 5 6 7 8 9 A B C D E F
    • RGB Colors
    • WiFi Passwords

DECIMAL

• There are 1528 pigeons on a ledge.
  • Base-10: What does this mean?
    • 8 is in the one’s place
    • 2 is in the ten’s place
    • 5 is in the hundred’s place
    • 1 is in the thousand’s place

• There are 1528 pigeons on a ledge.
  • Base-10: What does this mean?
    • $8 \times 1$
    • $2 \times 10$
    • $5 \times 100$
    • $1 \times 1000$

$\begin{align*}
8 \times 1 &= 8 \\
2 \times 10 &= 20 \\
5 \times 100 &= 500 \\
1 \times 1000 &= 1000 \\
\hline
\text{Total} &= 1528
\end{align*}$
BINARY

- Binary numbers work the same way
- Piece of cake!
- There are $1101$ ducks in a row
  - Base-2: What does this mean?
    - $1 * 2^3 = 8$
    - $0 * 2^1 = 0$
    - $1 * 2^0 = 1$

$13$

BINARY CODES

- Everything in the computer is encoded using collections of bits.
  - Usually grouped into one or more bytes.

ASCII CODES

<table>
<thead>
<tr>
<th>Char</th>
<th>Dec</th>
<th>Oct</th>
<th>Hex</th>
<th>Char</th>
<th>Dec</th>
<th>Oct</th>
<th>Hex</th>
<th>Char</th>
<th>Dec</th>
<th>Oct</th>
<th>Hex</th>
<th>Char</th>
<th>Dec</th>
<th>Oct</th>
<th>Hex</th>
</tr>
</thead>
<tbody>
<tr>
<td>(nul)</td>
<td>0</td>
<td>0000</td>
<td>0x00</td>
<td>(sp)</td>
<td>32</td>
<td>0040</td>
<td>0x20</td>
<td>@</td>
<td>64</td>
<td>0100</td>
<td>0x40</td>
<td>`</td>
<td>96</td>
<td>0140</td>
<td>0x60</td>
</tr>
<tr>
<td>(soh)</td>
<td>1</td>
<td>0001</td>
<td>0x01</td>
<td>!</td>
<td>33</td>
<td>0041</td>
<td>0x21</td>
<td>A</td>
<td>65</td>
<td>0101</td>
<td>0x41</td>
<td>a</td>
<td>97</td>
<td>0141</td>
<td>0x61</td>
</tr>
<tr>
<td>(stx)</td>
<td>2</td>
<td>0002</td>
<td>0x02</td>
<td>&quot;</td>
<td>34</td>
<td>0042</td>
<td>0x22</td>
<td>B</td>
<td>66</td>
<td>0102</td>
<td>0x42</td>
<td>b</td>
<td>98</td>
<td>0142</td>
<td>0x62</td>
</tr>
<tr>
<td>(etx)</td>
<td>3</td>
<td>0003</td>
<td>0x03</td>
<td>#</td>
<td>35</td>
<td>0043</td>
<td>0x23</td>
<td>C</td>
<td>67</td>
<td>0103</td>
<td>0x43</td>
<td>c</td>
<td>99</td>
<td>0143</td>
<td>0x63</td>
</tr>
<tr>
<td>(eot)</td>
<td>4</td>
<td>0004</td>
<td>0x04</td>
<td>$</td>
<td>36</td>
<td>0044</td>
<td>0x24</td>
<td>D</td>
<td>68</td>
<td>0104</td>
<td>0x44</td>
<td>d</td>
<td>100</td>
<td>0144</td>
<td>0x64</td>
</tr>
<tr>
<td>(enq)</td>
<td>5</td>
<td>0005</td>
<td>0x05</td>
<td>%</td>
<td>37</td>
<td>0045</td>
<td>0x25</td>
<td>E</td>
<td>69</td>
<td>0105</td>
<td>0x45</td>
<td>e</td>
<td>101</td>
<td>0145</td>
<td>0x65</td>
</tr>
<tr>
<td>(ack)</td>
<td>6</td>
<td>0006</td>
<td>0x06</td>
<td>&amp;</td>
<td>38</td>
<td>0046</td>
<td>0x26</td>
<td>F</td>
<td>70</td>
<td>0106</td>
<td>0x46</td>
<td>f</td>
<td>102</td>
<td>0146</td>
<td>0x66</td>
</tr>
<tr>
<td>(bel)</td>
<td>7</td>
<td>0007</td>
<td>0x07</td>
<td>'</td>
<td>39</td>
<td>0047</td>
<td>0x27</td>
<td>G</td>
<td>71</td>
<td>0107</td>
<td>0x47</td>
<td>g</td>
<td>103</td>
<td>0147</td>
<td>0x67</td>
</tr>
<tr>
<td>(bs)</td>
<td>8</td>
<td>0010</td>
<td>0x08</td>
<td>(</td>
<td>40</td>
<td>0050</td>
<td>0x28</td>
<td>H</td>
<td>72</td>
<td>0110</td>
<td>0x48</td>
<td>h</td>
<td>104</td>
<td>0148</td>
<td>0x68</td>
</tr>
<tr>
<td>(ht)</td>
<td>9</td>
<td>0011</td>
<td>0x09</td>
<td>)</td>
<td>41</td>
<td>0051</td>
<td>0x29</td>
<td>I</td>
<td>73</td>
<td>0111</td>
<td>0x49</td>
<td>i</td>
<td>105</td>
<td>0149</td>
<td>0x69</td>
</tr>
<tr>
<td>(nl)</td>
<td>10</td>
<td>0012</td>
<td>0x0a</td>
<td>*</td>
<td>42</td>
<td>0052</td>
<td>0x2a</td>
<td>J</td>
<td>74</td>
<td>0112</td>
<td>0x4a</td>
<td>j</td>
<td>106</td>
<td>0150</td>
<td>0x6a</td>
</tr>
<tr>
<td>(vt)</td>
<td>11</td>
<td>0013</td>
<td>0x0b</td>
<td>+</td>
<td>43</td>
<td>0053</td>
<td>0x2b</td>
<td>K</td>
<td>75</td>
<td>0113</td>
<td>0x4b</td>
<td>k</td>
<td>107</td>
<td>0151</td>
<td>0x6b</td>
</tr>
<tr>
<td>(np)</td>
<td>12</td>
<td>0014</td>
<td>0x0c</td>
<td>,</td>
<td>44</td>
<td>0054</td>
<td>0x2c</td>
<td>L</td>
<td>76</td>
<td>0114</td>
<td>0x4c</td>
<td>l</td>
<td>108</td>
<td>0152</td>
<td>0x6c</td>
</tr>
<tr>
<td>(cr)</td>
<td>13</td>
<td>0015</td>
<td>0x0d</td>
<td>-</td>
<td>45</td>
<td>0055</td>
<td>0x2d</td>
<td>M</td>
<td>77</td>
<td>0115</td>
<td>0x4d</td>
<td>m</td>
<td>109</td>
<td>0153</td>
<td>0x6d</td>
</tr>
<tr>
<td>(so)</td>
<td>14</td>
<td>0016</td>
<td>0x0e</td>
<td>.</td>
<td>46</td>
<td>0056</td>
<td>0x2e</td>
<td>N</td>
<td>78</td>
<td>0116</td>
<td>0x4e</td>
<td>n</td>
<td>110</td>
<td>0154</td>
<td>0x6e</td>
</tr>
<tr>
<td>(si)</td>
<td>15</td>
<td>0017</td>
<td>0x0f</td>
<td>/</td>
<td>47</td>
<td>0057</td>
<td>0x2f</td>
<td>O</td>
<td>79</td>
<td>0117</td>
<td>0x4f</td>
<td>o</td>
<td>111</td>
<td>0155</td>
<td>0x6f</td>
</tr>
<tr>
<td>(dle)</td>
<td>16</td>
<td>0020</td>
<td>0x10</td>
<td>0</td>
<td>48</td>
<td>0060</td>
<td>0x30</td>
<td>P</td>
<td>80</td>
<td>0120</td>
<td>0x50</td>
<td>p</td>
<td>112</td>
<td>0160</td>
<td>0x70</td>
</tr>
<tr>
<td>(dc1)</td>
<td>17</td>
<td>0021</td>
<td>0x11</td>
<td>1</td>
<td>49</td>
<td>0061</td>
<td>0x31</td>
<td>Q</td>
<td>81</td>
<td>0121</td>
<td>0x51</td>
<td>q</td>
<td>113</td>
<td>0161</td>
<td>0x71</td>
</tr>
<tr>
<td>(dc2)</td>
<td>18</td>
<td>0022</td>
<td>0x12</td>
<td>2</td>
<td>50</td>
<td>0062</td>
<td>0x32</td>
<td>R</td>
<td>82</td>
<td>0122</td>
<td>0x52</td>
<td>r</td>
<td>114</td>
<td>0162</td>
<td>0x72</td>
</tr>
<tr>
<td>(dc3)</td>
<td>19</td>
<td>0023</td>
<td>0x13</td>
<td>3</td>
<td>51</td>
<td>0063</td>
<td>0x33</td>
<td>S</td>
<td>83</td>
<td>0123</td>
<td>0x53</td>
<td>s</td>
<td>115</td>
<td>0163</td>
<td>0x73</td>
</tr>
<tr>
<td>(dc4)</td>
<td>20</td>
<td>0024</td>
<td>0x14</td>
<td>4</td>
<td>52</td>
<td>0064</td>
<td>0x34</td>
<td>T</td>
<td>84</td>
<td>0124</td>
<td>0x54</td>
<td>t</td>
<td>116</td>
<td>0164</td>
<td>0x74</td>
</tr>
<tr>
<td>(nak)</td>
<td>21</td>
<td>0025</td>
<td>0x15</td>
<td>5</td>
<td>53</td>
<td>0065</td>
<td>0x35</td>
<td>U</td>
<td>85</td>
<td>0125</td>
<td>0x55</td>
<td>u</td>
<td>117</td>
<td>0165</td>
<td>0x75</td>
</tr>
<tr>
<td>(syn)</td>
<td>22</td>
<td>0026</td>
<td>0x16</td>
<td>6</td>
<td>54</td>
<td>0066</td>
<td>0x36</td>
<td>V</td>
<td>86</td>
<td>0126</td>
<td>0x56</td>
<td>v</td>
<td>118</td>
<td>0166</td>
<td>0x76</td>
</tr>
<tr>
<td>(etb)</td>
<td>23</td>
<td>0027</td>
<td>0x17</td>
<td>7</td>
<td>55</td>
<td>0067</td>
<td>0x37</td>
<td>W</td>
<td>87</td>
<td>0127</td>
<td>0x57</td>
<td>w</td>
<td>119</td>
<td>0167</td>
<td>0x77</td>
</tr>
<tr>
<td>(can)</td>
<td>24</td>
<td>0030</td>
<td>0x18</td>
<td>8</td>
<td>56</td>
<td>0070</td>
<td>0x38</td>
<td>X</td>
<td>88</td>
<td>0130</td>
<td>0x58</td>
<td>x</td>
<td>120</td>
<td>0170</td>
<td>0x78</td>
</tr>
<tr>
<td>(em)</td>
<td>25</td>
<td>0031</td>
<td>0x19</td>
<td>9</td>
<td>57</td>
<td>0071</td>
<td>0x39</td>
<td>Y</td>
<td>89</td>
<td>0131</td>
<td>0x59</td>
<td>y</td>
<td>121</td>
<td>0171</td>
<td>0x79</td>
</tr>
<tr>
<td>(sub)</td>
<td>26</td>
<td>0032</td>
<td>0x1a</td>
<td>:</td>
<td>58</td>
<td>0072</td>
<td>0x3a</td>
<td>Z</td>
<td>90</td>
<td>0132</td>
<td>0x5a</td>
<td>z</td>
<td>122</td>
<td>0172</td>
<td>0x7a</td>
</tr>
<tr>
<td>(esc)</td>
<td>27</td>
<td>0033</td>
<td>0x1b</td>
<td>;</td>
<td>59</td>
<td>0073</td>
<td>0x3b</td>
<td>[</td>
<td>91</td>
<td>0133</td>
<td>0x5b</td>
<td>{</td>
<td>123</td>
<td>0173</td>
<td>0x7b</td>
</tr>
<tr>
<td>(fs)</td>
<td>28</td>
<td>0034</td>
<td>0x1c</td>
<td>&lt;</td>
<td>60</td>
<td>0074</td>
<td>0x3c</td>
<td>\</td>
<td>92</td>
<td>0134</td>
<td>0x5c</td>
<td></td>
<td></td>
<td>124</td>
<td>0174</td>
</tr>
<tr>
<td>(gs)</td>
<td>29</td>
<td>0035</td>
<td>0x1d</td>
<td>=</td>
<td>61</td>
<td>0075</td>
<td>0x3d</td>
<td>]</td>
<td>93</td>
<td>0135</td>
<td>0x5d</td>
<td>}</td>
<td>125</td>
<td>0175</td>
<td>0x7d</td>
</tr>
<tr>
<td>(rs)</td>
<td>30</td>
<td>0036</td>
<td>0x1e</td>
<td>&gt;</td>
<td>62</td>
<td>0076</td>
<td>0x3e</td>
<td>^</td>
<td>94</td>
<td>0136</td>
<td>0x5e</td>
<td>~</td>
<td>126</td>
<td>0176</td>
<td>0x7e</td>
</tr>
<tr>
<td>(us)</td>
<td>31</td>
<td>0037</td>
<td>0x1f</td>
<td>?</td>
<td>63</td>
<td>0077</td>
<td>0x3f</td>
<td>(del)</td>
<td>127</td>
<td>0177</td>
<td>0x7f</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
BINARY CODES

• The number 7 can be represented as 0000 0111.

Even the instructions the computer uses are bit strings.
  – The instruction to add two numbers for a particular machine might be 1011 1011.

REPRESENTING INFORMATION

• **Two’s complement** is a representation that allows both positive and negative integers.
  – For instance, a byte integer represents numbers from -128 to +127.
• **IEEE 754** is a way to represent floating point numbers.
• **Unicode** is a way to represent characters.
  – It is better than earlier representations because it can be used for additional alphabets, etc.

FLOATING POINT NUMBERS

• These are numbers with a fractional part
  – 6.17020389
FLOATING POINT NUMBERS

- They are stored as binary numbers in scientific notation.
- For example, the value -52.202 is stored as -0.52202 \times 10^2.
- It has three parts:
  - The sign, that is + or - (1 bit)
  - The digits in the number (mantissa)
  - The exponent (8 bits)

FLOATING POINT NUMBERS

- There are two types of floating point numbers:
  - Float
    - with 6-7 significant digits accuracy
  - Double
    - with 14-15 significant digits accuracy

OPERATOR ORDER

- The default evaluation order is
  - Negation -
  - Multiplication *, Division /, Modulo (remainder) %
  - Addition +, Subtraction -
- Negation is done before multiplication, which is done before addition.
- If you have operators of the same precedence (e.g. *, /) evaluate them from left to right.

OPERATOR ORDER

- The default evaluation order is
  - - (Negation)
  - *, /, %
  - +, - (Subtraction)
- So,
  \[ 2 + 4 \times 3 \]
  is 14.
- As opposed to 18, which it would be if evaluation was simply left to right.
OPERATOR ORDER

• The default evaluation order is
  - - (Negation)
  - *, /, %
  - +, - (Subtraction)

• The default order can be changed
  - By using parenthesis
    - (2 + 4) * 3 versus 2 + 4 * 3

MATH OPERATOR ORDER EXERCISE

• Try 2 + 3 * 4 + 5
• Add parentheses to make it clear what is happening first. 2 + (3 * 4) + 5
• How do you change it so that 2 + 3 happens first? (2 + 3) * 4 + 5
• How do you change it so that it multiplies the result of 2 + 3 and the result of 4 + 5? (2 + 3) * (4 + 5)

PRINTING OUTPUT TO THE CONSOLE

• In a program, you will often want to output the value of something.
• In Java the way to print to the console (screen) is to use
  - System.out.println();
    • Prints out the value of the thing in the parentheses and a new line
  - System.out.print();
    • To print just the thing in the parentheses without a new line

CONSOLE OUTPUT EXERCISE

• Use System.out.println() to print the results of expression to the console
  - System.out.println(3 * 28);
  - System.out.println(14 – 7);
  - System.out.println(10 / 2);
  - System.out.println(128 + 234);
  - System.out.println("Hi" + "There");
  - System.out.println("128 + 234");
• Try using System.out.print() instead
  - What is the difference?
• One thing that a computer can do is compare numbers.
• Java does this with **relational operators**.
• They test a particular relation between numbers.
  – Return true if that relationship holds.
  – Return false if it does not.

**Greater than**
• Represented by `>`
  - `4 > 3` is true
  - `3 > 3` is false
  - `3 > 4` is false

**Less than**
• Represented by `<`
  - `2 < 3` is true
  - `3 < 2` is false

**Greater than or equal**
• Represented by `>=`
  - `3 >= 4` is false
  - `3 >= 3` is true
  - `2 >= 4` is false

**Less than or equal**
• Represented by `<=`
  - `2 <= 3` is true
  - `2 <= 2` is true
  - `4 <= 2` is false

**Equal**
• Represented by `==`
  - `3 == 3` is true
  - `3 == 4` is false

**Not equal**
• Represented by `!=`
  - `3 != 4` is true
  - `3 != 3` is false
COMPARISON (RELATIONAL) OPERATORS

- Equal ==
  3 == 3 is true
  3 == 4 is false

- Not equal !=
  3 != 4 is true
  3 != 3 is false

Like many programming languages, Java uses "==" to test for equality. This may strike you as odd, since mathematics notation trains us to think of "=" for this. But, as we will see, Java uses "=" to assign values.

It's confusing, but you get used to it.

COMPARISON OPERATORS EXERCISE

- In DrJava
  – Try out the comparison operators in the interactions pane
    - with numbers
      3 < 4
      4 <= 4
      5 < 4
      6 == 6.0

COMPARISON OPERATORS EXERCISE

- Try comparing
  'A' == 'a'

- What do you get?
- Why?
STRINGS

- Java has a type called String
- String is not a simple *(primitive)* type like the ones we’ve seen so far.
- A string is an object that holds a sequence of characters.
  - It can have many characters
  - “This is one long string with spaces in it.”
  - It can have no characters (the null string “”)
  - Everything in a string will be printed out as it was entered:
    - Even what would otherwise be interpreted as math operations: “128 + 234”.

VARIABLES

- Using the primitive types, we can define places where the computer can store and access data of these types.
- These are called variables.

STRINGS

- Java knows how to “add” strings
  - It returns a new string with the characters of the second string after the characters of the first.
    - With no added space
  - This is called *concatenation*.
- Some examples:
  - “Foot” + “ball” is “Football”
  - ”Java” + ”doesn’t” + ”add” + ”spaces” is ”Javadoesn’taddspaces”
  - ”Add ” + ”your ” + ”own ” + ”spaces” is ”Add your own spaces”

VARIABLES

- To declare a variable in Java, you must specify two things:
  - The variable’s type
  - The variable’s name
- Some examples:
  - int testScoreSum
  - boolean doTheValuesMatch
• To declare a variable in Java, you must specify two things:
  – The variable's type
  – The variable's name

• Some examples:
  – int testScoreSum
  – boolean doTheValuesMatch

• Once you have declared a variable, your Java program can store information there.

  testScoreSum = 0;
  doTheValuesMatch = false;

• This is known as assignment
• This is what the “=” operator is used for; instead of checking for equality.

• Assignment statements can use calculations.

  testScoreSum = 2 + 3 + 7;
• You can also retrieve the values stored in variables.

• For instance, this statement will get and print out the current value in the variable `testScoreSum`:

```java
System.out.println(testScoreSum);
```

```java
int tomsScore;
int susScore;
int patsScore;
int testScoreSum;
tomsScore = 4;
susScore = 3;
patsScore = 14;
testScoreSum = tomsScore + susScore + patsScore;
System.out.println(testScoreSum);
```

• Variables can also be used on the right hand side of assignment statements.

• You can also assign values to variables when you declare them.
int tomsScore = 4;
int susScore = 3;
int patsScore = 14;
int testScoreSum;
testScoreSum = tomsScore + susScore + patsScore;
System.out.println(testScoreSum);

double testScoreMean; // The total of the scores
int testScoreSum; // The sum is 
double testScoreMean; /* The average */
testScoreSum = tomsScore + susScore + patsScore;
testScoreMean = testScoreSum / 3.0;
System.out.println("The sum is " + testScoreSum);
System.out.println("The mean is " + testScoreMean);