Introduction to Java

Sizes of Primitive Types

- byte: 8 bits
- short: 8 bits
- int: 8 bits, 8 bits
- long: 8 bits, 8 bits, 8 bits, 8 bits, 8 bits, 8 bits, 8 bits
- float: 8 bits, 8 bits, 8 bits
- double: 8 bits, 8 bits, 8 bits, 8 bits, 8 bits, 8 bits, 8 bits, 8 bits
- char: 8 bits
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.
  - \( 2^8 = 256 \).

Another way to think of this is that a byte can represent 256 different patterns of 0s and 1s.
- We can assign different meanings to each pattern.
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?
  - $8 \times 10^0 = 8$
  - $2 \times 10^1 = 20$
  - $5 \times 10^2 = 500$
  - $1 \times 10^3 = 1000$
  - $8 + 20 + 500 + 1000 = 1528$
Binary

- Binary numbers work the same way
- Piece of cake!
- There are 1101 ducks in a row
- Base-2: What does this mean?
  \[
  1 \times 2^0 = 1 \\
  0 \times 2^1 = 0 \\
  1 \times 2^2 = 4 \\
  + 1 \times 2^3 = 8 \\
  \]
  \[13\]

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>
</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>
</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>
</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>
</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>
</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>
</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>
</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>
</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>
</tr>
<tr>
<td>(bs)</td>
<td>8</td>
<td>0008</td>
<td>0x08</td>
<td>(</td>
<td>40</td>
<td>0048</td>
<td>0x28</td>
<td>H</td>
<td>72</td>
<td>0108</td>
<td>0x48</td>
</tr>
<tr>
<td>(ht)</td>
<td>9</td>
<td>0009</td>
<td>0x09</td>
<td>)</td>
<td>41</td>
<td>0049</td>
<td>0x29</td>
<td>I</td>
<td>73</td>
<td>0109</td>
<td>0x49</td>
</tr>
<tr>
<td>(nl)</td>
<td>10</td>
<td>0010</td>
<td>0x0a</td>
<td>*</td>
<td>42</td>
<td>004A</td>
<td>0x2A</td>
<td>J</td>
<td>74</td>
<td>0110</td>
<td>0x4A</td>
</tr>
<tr>
<td>(vt)</td>
<td>11</td>
<td>0011</td>
<td>0x0B</td>
<td>+</td>
<td>43</td>
<td>004B</td>
<td>0x2B</td>
<td>K</td>
<td>75</td>
<td>0111</td>
<td>0x4B</td>
</tr>
<tr>
<td>(np)</td>
<td>12</td>
<td>0012</td>
<td>0x0C</td>
<td>,</td>
<td>44</td>
<td>004C</td>
<td>0x2C</td>
<td>L</td>
<td>76</td>
<td>0112</td>
<td>0x4C</td>
</tr>
<tr>
<td>(cr)</td>
<td>13</td>
<td>0013</td>
<td>0x0D</td>
<td>-</td>
<td>45</td>
<td>004D</td>
<td>0x2D</td>
<td>M</td>
<td>77</td>
<td>0113</td>
<td>0x4D</td>
</tr>
<tr>
<td>(enq)</td>
<td>14</td>
<td>0014</td>
<td>0x0E</td>
<td>.</td>
<td>46</td>
<td>004E</td>
<td>0x2E</td>
<td>N</td>
<td>78</td>
<td>0114</td>
<td>0x4E</td>
</tr>
<tr>
<td>(esc)</td>
<td>15</td>
<td>0015</td>
<td>0x0F</td>
<td>/</td>
<td>47</td>
<td>004F</td>
<td>0x2F</td>
<td>O</td>
<td>79</td>
<td>0115</td>
<td>0x4F</td>
</tr>
<tr>
<td>(ins)</td>
<td>16</td>
<td>0016</td>
<td></td>
<td>0</td>
<td>48</td>
<td>0050</td>
<td>0x30</td>
<td>P</td>
<td>80</td>
<td>0116</td>
<td>0x50</td>
</tr>
<tr>
<td>(del)</td>
<td>17</td>
<td>0017</td>
<td></td>
<td>1</td>
<td>49</td>
<td>0065</td>
<td>0x35</td>
<td>Q</td>
<td>81</td>
<td>0117</td>
<td>0x65</td>
</tr>
<tr>
<td>(clm)</td>
<td>18</td>
<td>0018</td>
<td></td>
<td>2</td>
<td>50</td>
<td>0070</td>
<td>0x3A</td>
<td>R</td>
<td>82</td>
<td>0118</td>
<td>0x70</td>
</tr>
<tr>
<td>(cmd)</td>
<td>19</td>
<td>0019</td>
<td></td>
<td>3</td>
<td>51</td>
<td>0071</td>
<td>0x3B</td>
<td>S</td>
<td>83</td>
<td>0119</td>
<td>0x71</td>
</tr>
<tr>
<td>(tab)</td>
<td>20</td>
<td>0020</td>
<td></td>
<td>4</td>
<td>52</td>
<td>0072</td>
<td>0x3C</td>
<td>T</td>
<td>84</td>
<td>0120</td>
<td>0x72</td>
</tr>
<tr>
<td>(app)</td>
<td>21</td>
<td>0021</td>
<td></td>
<td>5</td>
<td>53</td>
<td>0073</td>
<td>0x3D</td>
<td>U</td>
<td>85</td>
<td>0121</td>
<td>0x73</td>
</tr>
<tr>
<td>(crm)</td>
<td>22</td>
<td>0022</td>
<td></td>
<td>6</td>
<td>54</td>
<td>0074</td>
<td>0x3E</td>
<td>V</td>
<td>86</td>
<td>0122</td>
<td>0x74</td>
</tr>
<tr>
<td>(entr)</td>
<td>23</td>
<td>0023</td>
<td></td>
<td>7</td>
<td>55</td>
<td>0075</td>
<td>0x3F</td>
<td>W</td>
<td>87</td>
<td>0123</td>
<td>0x75</td>
</tr>
<tr>
<td>(can)</td>
<td>24</td>
<td>0024</td>
<td></td>
<td>8</td>
<td>56</td>
<td>0076</td>
<td>0x40</td>
<td>X</td>
<td>88</td>
<td>0124</td>
<td>0x76</td>
</tr>
<tr>
<td>(em)</td>
<td>25</td>
<td>0025</td>
<td></td>
<td>9</td>
<td>57</td>
<td>0077</td>
<td>0x41</td>
<td>Y</td>
<td>89</td>
<td>0125</td>
<td>0x77</td>
</tr>
<tr>
<td>(sub)</td>
<td>26</td>
<td>0026</td>
<td></td>
<td>10</td>
<td>58</td>
<td>0078</td>
<td>0x42</td>
<td>Z</td>
<td>90</td>
<td>0126</td>
<td>0x78</td>
</tr>
<tr>
<td>(esc)</td>
<td>27</td>
<td>0027</td>
<td></td>
<td>11</td>
<td>59</td>
<td>0079</td>
<td>0x43</td>
<td>a</td>
<td>91</td>
<td>0127</td>
<td>0x79</td>
</tr>
<tr>
<td>(del)</td>
<td>28</td>
<td>0028</td>
<td></td>
<td>12</td>
<td>60</td>
<td>007A</td>
<td>0x44</td>
<td>b</td>
<td>92</td>
<td>0128</td>
<td>0x7A</td>
</tr>
<tr>
<td>(ins)</td>
<td>29</td>
<td>0029</td>
<td></td>
<td>13</td>
<td>61</td>
<td>007B</td>
<td>0x45</td>
<td>c</td>
<td>93</td>
<td>0129</td>
<td>0x7B</td>
</tr>
<tr>
<td>(cmd)</td>
<td>30</td>
<td>0030</td>
<td></td>
<td>14</td>
<td>62</td>
<td>007C</td>
<td>0x46</td>
<td>d</td>
<td>94</td>
<td>0130</td>
<td>0x7C</td>
</tr>
<tr>
<td>(app)</td>
<td>31</td>
<td>0031</td>
<td></td>
<td>15</td>
<td>63</td>
<td>007D</td>
<td>0x47</td>
<td>e</td>
<td>95</td>
<td>0131</td>
<td>0x7D</td>
</tr>
</tbody>
</table>

- The letter “a” can be represented in memory as 0110 0001.
  - This is an example of a bit string.
  - Note that it uses eight bits - a byte.
  - 0110 0001 in decimal?
  - 97
Binary Codes

- The number 7 can be represented as 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
  - 6170.20389
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 (significand)
  - 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.

- The default evaluation order is
  - - (Negation)
  - *, /, %
  - +, - (Subtraction)
- So,
  \(2 + 4 * 3\)
  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?
Comparison (Relational) Operators

• One thing that 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".

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

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

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`
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

Variables

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

\[
\text{testScoreSum} = 0;
\]

\[
\text{doTheValuesMatch} = \text{false};
\]

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

Variables

• Assignment statements can use calculations.

\[
\text{testScoreSum} = 2 + 3 + 7;
\]
Variables

• 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);
  ```

Variables

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

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

Variables

```java
int tomsScore = 4;
int susScore = 3;
int patsScore = 14;
int testScoreSum; // The total of the scores
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);
```