Title

Primitive Types, String, Console I/O

Cheng, Wei  COMP110-001  May 16, 2014
Today’s contents

• Primitive Data Types
  – Integer
  – Boolean
  – Float/Double
  – Character

• String

• Console I/O
Memory & Data & Variable

- Memory is just a long list of binary digits (bits) that you can use to store information. It’s divided into units of 1 byte (8 bits):

```
0001 1001 1001 0101 0011 0001 1010 1111 1111
```

Unit Addr:

<table>
<thead>
<tr>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>6</td>
<td>....</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Primitive Types

- Integer (**byte**, **short**, **int**, **long**)
  - 0, -3, 5, 43
- Floating-point number (**float**, **double**)
  - 0.5, 12.4863, -4.3
- Characters (**char**)
  - A, r, %, T
- Boolean (**boolean**)
  - true, false
Primitive Types: small to big

+/- $1.79 \times 10^{308}$ to +/− $4.94 \times 10^{-324}$

We have talked about this before

-128 to 127
When declaring a variable, a certain amount of memory is assigned based on the declared primitive type.

The cells reserved will not be used by other variables later.
Assignment compatibilities of numerical values

- Usually, we need to put values of a certain type into variables of the same type.
- However, in some cases, the value will automatically be converted when types are different.

```java
int age;
age = 10;
double length;
length = age;
```
Type Casting for numerical values

- You can also cast a type into another type. But note the order of numerical types.

- $3 / 2 = 1$

- \((\text{double})\ 3 / (\text{double})\ 2 = 1.5\)

- Try it yourself:
  ```java
  System.out.println( 3/ 2);
  System.out.println( (float) 3/ (float) 2);
  ```

- What happens if you cast a double into int?
Basic arithmetic operators for numerical values

+, -, *, /, %,

- rate*rate + delta
- 1/(time + 3*mass)
- (a - 7)/(t + 9*v)

- Modular Arithmetic - %
- Remainder
- 7 % 3 = 1 (7 / 3 = 2, remainder 1)
- 8 % 3 = 2 (8 / 3 = 2, remainder 2)
- 9 % 3 = 0 (9 / 3 = 3, remainder 0)
Basic arithmetic operators for numerical values

Resulting types:
- `int / int = int`
- `float * float = float`
- `double + double = double`

What about:
- `float * int = ?`
- `double + float = ?`
Basic arithmetic operators for numerical values

Each type has its range and precision. Computer does not have infinite precision.

Do not go beyond the range or precision.

E.g.:

2147483647 + 1 = ?
1 + 0.00000000000000001 = ?
Character

char letter = 'A';

2 bytes in Java

- All characters you can type with keyboard
- Special control characters (new line, end of file...)
- Characters in many other languages

<table>
<thead>
<tr>
<th>Dec 16</th>
<th>Hex</th>
<th>Oct</th>
<th>Chr</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>000</td>
<td>DEL (bell)</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>001</td>
<td>BS (backspace)</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>002</td>
<td>HT (start of heading)</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>003</td>
<td>LF (start of text)</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>004</td>
<td>VT (end of text)</td>
</tr>
<tr>
<td>5</td>
<td>5</td>
<td>005</td>
<td>FF (end of transmission)</td>
</tr>
<tr>
<td>6</td>
<td>6</td>
<td>006</td>
<td>CR (enquiry)</td>
</tr>
<tr>
<td>7</td>
<td>7</td>
<td>007</td>
<td>AK (acknowledge)</td>
</tr>
<tr>
<td>8</td>
<td>8</td>
<td>010</td>
<td>DS (acknowledgment)</td>
</tr>
<tr>
<td>9</td>
<td>9</td>
<td>011</td>
<td>DC1 (horizontal tab)</td>
</tr>
<tr>
<td>10</td>
<td>A</td>
<td>012</td>
<td>DC2 (vertical tab)</td>
</tr>
<tr>
<td>11</td>
<td>B</td>
<td>013</td>
<td>DC3 (form feed, new page)</td>
</tr>
<tr>
<td>12</td>
<td>C</td>
<td>014</td>
<td>DC4 (form feed, new page)</td>
</tr>
<tr>
<td>13</td>
<td>D</td>
<td>015</td>
<td>GR (carriage return)</td>
</tr>
<tr>
<td>14</td>
<td>E</td>
<td>016</td>
<td>SO (shift out)</td>
</tr>
<tr>
<td>15</td>
<td>F</td>
<td>017</td>
<td>SI (shift in)</td>
</tr>
<tr>
<td>16</td>
<td>10</td>
<td>020</td>
<td>DL (data link escape)</td>
</tr>
<tr>
<td>17</td>
<td>11</td>
<td>021</td>
<td>DC1 (device control 1)</td>
</tr>
<tr>
<td>18</td>
<td>12</td>
<td>022</td>
<td>DC2 (device control 2)</td>
</tr>
<tr>
<td>19</td>
<td>13</td>
<td>023</td>
<td>DC3 (device control 3)</td>
</tr>
<tr>
<td>20</td>
<td>14</td>
<td>024</td>
<td>DC4 (device control 4)</td>
</tr>
<tr>
<td>21</td>
<td>15</td>
<td>025</td>
<td>NAK (negative acknowledge)</td>
</tr>
<tr>
<td>22</td>
<td>16</td>
<td>026</td>
<td>SYN (synchronous idle)</td>
</tr>
<tr>
<td>23</td>
<td>17</td>
<td>027</td>
<td>ETX (end of text block)</td>
</tr>
<tr>
<td>24</td>
<td>18</td>
<td>030</td>
<td>CAN (cancel)</td>
</tr>
<tr>
<td>25</td>
<td>19</td>
<td>031</td>
<td>EM (end of medium)</td>
</tr>
<tr>
<td>26</td>
<td>20</td>
<td>032</td>
<td>SUB (substitute)</td>
</tr>
<tr>
<td>27</td>
<td>21</td>
<td>033</td>
<td>ESC (escape)</td>
</tr>
<tr>
<td>28</td>
<td>22</td>
<td>034</td>
<td>FS (file separator)</td>
</tr>
<tr>
<td>29</td>
<td>23</td>
<td>035</td>
<td>GS (group separator)</td>
</tr>
<tr>
<td>30</td>
<td>24</td>
<td>036</td>
<td>RS (record separator)</td>
</tr>
<tr>
<td>31</td>
<td>25</td>
<td>037</td>
<td>US (unit separator)</td>
</tr>
</tbody>
</table>

Source: www.LeadingTables.com
Boolean

- Boolean (boolean): True or False. 1 bit in theory.
- Boolean operators:
  - `&&` ---- and
  - `||` ---- or
  - `!` ---- negation

- Write down the truth table if you are not sure

<table>
<thead>
<tr>
<th>and</th>
<th>True</th>
<th>False</th>
</tr>
</thead>
<tbody>
<tr>
<td>True</td>
<td></td>
<td></td>
</tr>
<tr>
<td>False</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>or</th>
<th>True</th>
<th>False</th>
</tr>
</thead>
<tbody>
<tr>
<td>True</td>
<td></td>
<td></td>
</tr>
<tr>
<td>False</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Order of Operations

- Arithmetic operations & parenthesis – check book if you are not sure

- Boolean: negation -> and -> or

- E.g., !true && (false || true) || true

- Mixed operations:
  - “The sum is ” + 5 + 6  -> “The sum is 56”
  - “The sum is ” + (5 + 6)  -> “The sum is 11”
String

• A special class. Object of String class can be defined using shortcut:

   String str = “UNC is Great”;

• Each String object consists of:
  – A sequence of characters (char)
    | U | N | C | i | s | G | r | e | a | t |
    | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
  – A set of methods that can process the sequence of characters
String

String str = “UNC is Great”;

<table>
<thead>
<tr>
<th>U</th>
<th>N</th>
<th>C</th>
<th>i</th>
<th>s</th>
<th>G</th>
<th>r</th>
<th>e</th>
<th>a</th>
<th>t</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>9</td>
</tr>
</tbody>
</table>

int len = str.length();
System.out.println(len);

char firstLetter = str.charAt(0);
System.out.println(firstLetter);

String lowerCaseStr = str.toLowerCase();

System.out.println( str.substring(0, 3) ); // this prints out UNC
String

• Check Java API for the whole list of methods in String
• You do not have to memorize them.
• But you should know how to use existing classes if you have documentation (just like Lab1)
How do you put quotes in a string?

System.out.println("How do I put \"quotes\" in my string?");
But what about backslashes?

System.out.println("How do I put a \ in my string?");
## Escape Characters

<table>
<thead>
<tr>
<th>Escape Character</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;</td>
<td>Double quote</td>
</tr>
<tr>
<td>'</td>
<td>Single quote</td>
</tr>
<tr>
<td>\</td>
<td>Backslash</td>
</tr>
<tr>
<td>\n</td>
<td>New line</td>
</tr>
<tr>
<td>\t</td>
<td>Tab</td>
</tr>
</tbody>
</table>
Screen Output

• We've seen several examples of screen output already.

• `System.out` is an object that is part of Java.

• `println()` is one of the methods available in the `System.out` object.
Printing to Screen

System.out.print(“this is a string”);
System.out.println(“this is a string”);

• What is the difference?

System.out.print(“This prints ”);
System.out.print(“multiple parts ”);
System.out.println(“in one line”);
Keyboard Input

- Near the beginning of your program, insert
  ```java
  import java.util.Scanner;
  ```

- Create an object of the **Scanner** class
  ```java
  Scanner Scanner_object_name = new Scanner(System.in);
  ```

- Read data using methods of the object
  ```java
  Scanner_object_name.next(); // → String
  Scanner_object_name.nextLine(); // → String
  Scanner_object_name.nextInt(); // → int
  Scanner_object_name.nextDouble(); // → double
  Scanner_object_name.nextBoolean(); // → boolean
  ```