Computational Expression

Computer and Java Basics

Janyl Jumadinova

9–11 September, 2019
Character Strings

string literal in class \texttt{String}

“ABC”

“This is interesting”

“ ”

“91”
string literal in class String
“ABC”
“This is interesting”
“ ”
“91”

- Use `print` or `println` methods to print a character string to the terminal
- `System.out.println(“CMPSC 100”);`
- the string “CMPSC 100” is a parameter: data sent to a method
String Concatenation

Appending one string to the end of another: use + operator
“This is ” + “interesting”
“Your grade is ” + “91”
String Concatenation

Appending one string to the end of another: use + operator
“This is ” + “interesting”
“Your grade is ” + “91”

+ is also used for arithmetic addition

System.out.println(”Adding ” + 12 + 23); is not the same as
System.out.println(”Adding ” + (12 + 23));
Escape Sequences

- Escape sequences, or escape characters, begin with a slash and are immediately followed by another character.
- This two-character sequence, inside “ ” allows you to control your output (\n, \t, \b) or output characters you wouldn’t otherwise be able to (\\, \”) inside a string.
## Escape Sequences

<table>
<thead>
<tr>
<th>Seq</th>
<th>Meaning</th>
<th>Example Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>\n</td>
<td>New line</td>
<td>System.out.println(&quot;Hi\nThere&quot;);</td>
</tr>
<tr>
<td>\t</td>
<td>Horizontal tab</td>
<td>System.out.println(&quot;What’s\tup?&quot;);</td>
</tr>
<tr>
<td>\b</td>
<td>Backspace</td>
<td>System.out.println(&quot;Hi\b Hey&quot;);</td>
</tr>
<tr>
<td>\</td>
<td>Backslash</td>
<td>System.out.println(&quot;Back\Slash&quot;);</td>
</tr>
<tr>
<td>”</td>
<td>Double quote</td>
<td>System.out.println(&quot;Dbl\Quote&quot;);</td>
</tr>
</tbody>
</table>
Variables

- **Variable** is a name for a memory’s location where a data value is stored.
Variables

- **Variable** is a name for a memory's location where a data value is stored.
- **Variable Declaration** allows the compiler to reserve space in the main memory that is large enough for the specified type.

```java
int count;
```

```java
int count = 0;
```

Must give a value to the variable before using it in the main method.
Variables

- **Variable** is a name for a memory’s location where a data value is stored.

- **Variable Declaration** allows the compiler to reserve space in the main memory that is large enough for the specified type
  ```java
  int count;
  ```

- **Variable Assignment** assigns a value to the variable
  ```java
  count = 0;
  ```
Variables

- **Variable** is a name for a memory’s location where a data value is stored.
- **Variable Declaration** allows the compiler to reserve space in the main memory that is large enough for the specified type
  ```java
  int count;
  ```
- **Variable Assignment** assigns a value to the variable
  ```java
  count = 0;
  ```
- Must give a value to the variable before using it in the main method.
Java Identifiers

- reserved keywords (class, public, static, void)
- Java classes, methods, variables: words we chose or make up when writing a program
  
  System, println, main, args
Java Identifiers

- reserved keywords (class, public, static, void)
- Java classes, methods, variables: words we chose or make up when writing a program
  
  System, println, main, args

Identifier

a letter followed by zero or more letters (including $ and _) and digits
Identifier Rules

- Identifiers must start with a letter, a currency character ($), or a connecting character such as the underscore (_).
- Identifiers cannot start with a number.
- After the first character, identifiers can contain any combination of letters, currency characters, connecting characters, or numbers.
- There is no limit to the number of characters an identifier can contain.
- You can't use a Java keyword as an identifier.
- Identifiers in Java are case-sensitive; foo and FOO are two different identifiers.
Data Types

- Data stored in memory is a string of bits (0 or 1)
Data Types

- Data stored in memory is a string of bits (0 or 1)
- How the computer interprets the string of bits depends on the context.
- In Java, we must make the context explicit by specifying the type of the data.
Java has two categories of data:
primitive data (e.g., number, character)
object data (programmer created types)

There are 8 primitive data types: byte, short, int, long, float, double, char, boolean

Primitive data are only single values; they have no special capabilities.
Primitive Data Types

- integers: byte, short, int, long
- floating point: float, double
- characters: char
- booleans: boolean
# Common Primitive Data Types

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
<th>Example of Literals</th>
</tr>
</thead>
<tbody>
<tr>
<td>int</td>
<td>integers (whole numbers)</td>
<td>42, 60634, -8</td>
</tr>
<tr>
<td>double</td>
<td>real numbers</td>
<td>0.039, -10.2</td>
</tr>
<tr>
<td>char</td>
<td>single characters</td>
<td>'a', 'B', '&amp;', '6'</td>
</tr>
<tr>
<td>boolean</td>
<td>logical values</td>
<td>true, false</td>
</tr>
</tbody>
</table>
## Range of Values

<table>
<thead>
<tr>
<th>Type</th>
<th>Storage</th>
<th>Range of Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>int</td>
<td>32 bits</td>
<td>-2,147,483,648 to 2,147,483,647</td>
</tr>
<tr>
<td>double</td>
<td>64 bits</td>
<td>$\pm 10^{-45}$ to $\pm 10^{38}$</td>
</tr>
<tr>
<td>char</td>
<td>16 bits = 2 bytes</td>
<td>0 to $2^{16}$ or \u0000 to \uFFFF</td>
</tr>
<tr>
<td>boolean</td>
<td>1 bit</td>
<td>NA</td>
</tr>
</tbody>
</table>