Computational Expression

Computer and Java Basics

Janyl Jumadinova

16 September, 2019
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;
  ```
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.
Variable names correspond to locations in the computer’s memory.
Memory Concepts

- Variable names correspond to locations in the computer’s memory.
- Every variable has:
  - a name,
  - a type,
  - a size,
  - a value

```java
first = input.nextInt();
```
stores whatever the user types into the location associated with `first`.
Memory Concepts

- Variable names correspond to locations in the computer’s memory.
- Every variable has:
  - a name,
  - a type,
  - a size,
  - a value
- `first = input.nextInt();` stores whatever the user types into the location associated with `first`.
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.
Data Types

- 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></td>
<td>64 bits</td>
<td>±10^{-45} to ±10^{38}</td>
</tr>
<tr>
<td>double</td>
<td>16 bits = 2 bytes</td>
<td>0 to 2^{16} or \u0000 to \uFFFF</td>
</tr>
<tr>
<td>char</td>
<td>1 bit</td>
<td>NA</td>
</tr>
<tr>
<td>boolean</td>
<td>NA</td>
<td></td>
</tr>
</tbody>
</table>
Literal

A constant value in Java is created by using a literal representation of it.

- 100 (integer literal)
- 98.6 (float literal)
- ‘X’ (character literal)
- ‘“This is a test”’ (String literal)
Constants

- Constants hold the same value during their existence.
- Can use a keyword `final` before the type and name of the variable:
  - always contains the same value.
- `final int MAX_BUDGET = 1000`
Expression is a combination of one or more operators (+, −, %, ...) and operands (literals, constants, variables,...)
Operators are evaluated in an expression according to the rules of precedence.
Order of Precedence

- Operators are evaluated in an expression according to the rules of precedence.
- Operators within ( ) are evaluated first.
- *, /, % evaluated next (L to R).
- +, - evaluated last (L to R).
Conversion

- **Widening**
  - from *byte* to *short, int, long, float* or *double*
  - from *short* to *int, long, float, double*
  - from *char* to *int, long, float, double*
  - from *int* to *long, float, double*
  - from *long* to *float, double*
  - from *float* to *double*
Narrowing - should be avoided!
- from \texttt{byte} to \texttt{char}
- from \texttt{short} to \texttt{byte}, \texttt{char}
- from \texttt{char} to \texttt{byte}, \texttt{short}
- from \texttt{int} to \texttt{byte}, \texttt{short}, \texttt{char}
- from \texttt{long} to \texttt{byte}, \texttt{short}, \texttt{char}, \texttt{int}
- from \texttt{float} to \texttt{byte}, \texttt{short}, \texttt{char}, \texttt{int}, \texttt{long}
- from \texttt{double} to \texttt{byte}, \texttt{short}, \texttt{char}, \texttt{int}, \texttt{long}, \texttt{float}
Conversion

- **Assignment:** grade = ‘A’
- **Promotion:** total/count, where total is a floating point value and count is an integer
  - Occurs automatically, count is *promoted* to a floating point value
- **Casting:** grade = (int) total
  - Java operator: type name in parentheses
  - Casting converts floating point value total into an integer, truncating any fractional part.
Scanner

- The Scanner class in the java.util package is a simple text scanner which can parse primitive types and strings.
- We can use the Scanner class to get the input.
The `Scanner` class in the `java.util` package is a simple text scanner which can parse primitive types and strings.

We can use the `Scanner` class to get the input.

We must first create an instance of the `Scanner` as:

```java
Scanner name = new Scanner (System.in);`
```
to read from the terminal,
The Scanner class in the java.util package is a simple text scanner which can parse primitive types and strings.

We can use the Scanner class to get the input.

We must first create an instance of the Scanner as:

Scanner name = new Scanner (System.in); to read from the terminal, or
Scanner name = new Scanner (File filename); to read from the file, where

File filename = new File ("input.txt");

and name is the name you choose for your instance of the Scanner
Scanner Methods

- `next()` : get the next word (token) as a String
- `nextLine()` : get a line of input as a String
- `nextInt()` : get an integer
- `nextDouble()` : get a double value
- `nextFloat()` : get a float value