Lecture 2: Variables and Operators

AITI Nigeria Summer 2012
University of Lagos.
Agenda

- Variables
  - Types
  - Naming
  - Assignment
- Data Types
- Type casting
- Operators
Declaring Variables in Java

Variables are created by declaring their type and their name as follows:

• Declaring an integer named “x”:
  – int x;

• Declaring a string named “greeting”:
  – String greeting;

• Note that we have not assigned values to these variables
Java Types: Integer Types

- **int**: Most numbers you will deal with.
- **long**: Big integers; science, finance, computing.
- **short**: Smaller integers. Not as useful.
- **byte**: Very small integers, useful for small data.
Java Types: Other Types

- **Floating Point (Decimal) Types:**
  - `float`: Single-precision decimal numbers
  - `double`: Double-precision decimal numbers.
  - Some phone platforms do not support FP.
- **String**: Letters, words, or sentences.
- **boolean**: True or false.
- **char**: Single Latin Alphanumeric characters
Variable Name Rules

- Variable names (or identifiers) may be any length, but must start with:
  - A letter (a – z, A-Z),
  - A dollar sign ($),
  - Or, an underscore (_).

- Identifiers cannot contain special operation symbols like +, -, *, /, &, %, ^, etc.

- Certain reserved keywords in the Java language are illegal.
  - int, double, String, etc.
Naming Variables

- Java is case sensitive
- A rose is not a Rose is not a ROSE
- Choose variable names that are informative
  - Good: `int studentExamGrade;`
  - Bad: `int tempvar3931;`
- Camel Case": Start variable names with lower case and capitalize each word:
  - “camelsHaveHumps”.
Review

- Which of the following are valid variable names?
  - $amount
  - 6tally
  - my*Name
  - salary
  - _score
  - first Name
  - short
Integer Types

• There are 4 primitive integer types: `byte`, `short`, `int`, `long`.

• Each type has a maximum value, based on its underlying binary representation:

  – Bytes: $\pm 128$ (8 bits)
  – Short: $\pm 2^{15} \approx 32,000$ (16 bits)
  – Int: $\pm 2^{31} \approx 2$ billion (32 bits)
  – Long: $\pm 2^{63} \approx$ really big (64 bits)
Overflow

• What happens when if we store Bill Gates’s net worth in an int?
  
  – Int: $\pm 2^{31} \approx 2$ billion (32 bits)
  – Bill’s net worth: > $40$ billion USD

• Undefined!
Floating Point Types

- Initialize doubles as you would write a decimal number:
  - `double y = 1.23;`
  - `double w = -3.21e-10; // -3.21x10^{-10}

- Doubles are more precise than Floats, but may take longer to perform operations.
Floating Point Types

- We must be careful with integer division:
  ```
  double z = 1/3;  // z = 0.0 ... Why?
  ```
When we want to convert one type to another, we use type casting.

The syntax is as follows:

Example code:
- `double decimalNumber = 1.234;`
- `int integerPart = (int)decimalNumber;`

Results:
- `decimalNumber == 1.234;`
- `integerPart == 1;`
Boolean Type

• Boolean is a data type that can be used in situations where there are two options, either true or false.
• The values true or false are case-sensitive keywords. Not True or TRUE.
• Booleans will be used later for testing properties of data.
• Example:
  - boolean monsterHungry = true;
  - boolean fileOpen = false;
Character Type

- Character is a data type that can be used to store a single character such as a letter, number, punctuation mark, or other symbol.

- Characters are a single letter enclosed in single quotes.

- Example:
  - `char` firstLetterOfName = 'e' ;
  - `char` myQuestion = '?' ;
String Type

- Strings are not a primitive. They are what's called an Object, which we will discuss later.

- Strings are sequences of characters surrounded by double quotations.

- Strings have a special append operator + that creates a new String:
  - String greeting = "Jam" + "bo";
  - String bigGreeting = greeting + "!";
Review

What data types would you use to store the following types of information?:

- Population of Kenya: int
- World Population: long
- Approximation of π: double
- Open/closed status of a file: boolean
- Your name: String
- First letter of your name: char
- $237.66: double
A Note on Statements

- A statement is a command that causes something to happen.
- All statements are terminated by semicolons ;
- Declaring a variable is a statement.
- Method (or function) calls are statements:
  - `System.out.println("Hello, World");`
- In lecture 4, we’ll learn how to control the execution flow of statements.
What are Operators?

- **Expressions** can be combinations of variables, primitives and operators that result in a value

- Operators are special symbols used for:
  - mathematical functions
  - assignment statements
  - logical comparisons

- Examples with operators:
  
  3 + 5 // uses + operator
  
  14 + 5 – 4 * (5 – 3) // uses +, -, * operators
The Operator Groups

• There are 5 different groups of operators:
  - Arithmetic Operators
  - Assignment Operator
  - Increment / Decrement Operators
  - Relational Operators
  - Conditional Operators

• The following slides will explain the different groups in more detail.
Arithmetic Operators

• Java has the usual 5 arithmetic operators:
  – +, -, ×, /, %

• Order of operations (or precedence):
  1. Parentheses (Brackets)
  2. Exponents (Order)
  3. Multiplication and Division from left to right
  4. Addition and Subtraction from left to right
Order of Operations (Cont’d)

• Example: 10 + 15 / 5;

• The result is different depending on whether the addition or division is performed first

\[(10 + 15) / 5 = 5\]
\[10 + (15 / 5) = 13\]

Without parentheses, Java will choose the second case

• You should be explicit and use parentheses to avoid confusion
Integer Division

• In the previous example, we were lucky that \((10 + 15) / 5\) gives an exact integer answer (5).

• But what if we divide 63 by 35?

• Depending on the data types of the variables that store the numbers, we will get different results.
The result of integer division is just the integer part of the quotient!
Assignment Expression

- The basic assignment operator (=) assigns the value of \( \text{expr} \) to \( \text{var} \)

\[
\text{name} = \text{value}
\]

- Java allows you to combine arithmetic and assignment operators into a single statement

- Examples:
  
  \[
  x = x + 5; \quad \text{is equivalent to} \quad x += 5;
  \]
  
  \[
  y = y * 7; \quad \text{is equivalent to} \quad y *= 7;
  \]
Increment/Decrement Operators

• `++` is called the increment operator. It is used to increase the value of a variable by 1.

For example:
\[
i = i + 1; \quad \text{can be written as:}\n++i; \quad \text{or} \quad i++;
\]

• `--` is called the decrement operator. It is used to decrease the value of a variable by 1.

\[
i = i - 1; \quad \text{can be written as:}\n--i; \quad \text{or} \quad i--;
\]
Increment Operators (cont’d)

The increment / decrement operator has two forms:

- Prefix Form e.g. `++i; --i;`
- Postfix Form e.g. `i++; i--;`
Prefix increment / decrement

- The prefix form first adds/subtracts 1 from the variable and then continues to any other operator in the expression.

- Example:

```c
int numOranges = 5;
int numApples = 10;
int numFruit;
numFruit = ++numOranges + numApples;
```

numFruit has value 16
numOranges has value 6
Postfix Increment/Decrement

- The postfix form i++, i-- first evaluates the entire expression and then adds 1 to the variable

- Example:

```java
int numOranges = 5;
int numApples = 10;
int numFruit;
numFruit = numOranges++ + numApples;

numFruit has value 15
numOranges has value 6
```
**Relational (Comparison) Operators**

- Relational operators compare two values
- They produce a boolean value (**true** or **false**) depending on the relationship

<table>
<thead>
<tr>
<th>Operation</th>
<th>....Is true when</th>
</tr>
</thead>
<tbody>
<tr>
<td>a &gt; b</td>
<td>a is greater than b</td>
</tr>
<tr>
<td>a &gt;= b</td>
<td>a is greater than or equal to b</td>
</tr>
<tr>
<td>a == b</td>
<td>a is equal to b</td>
</tr>
<tr>
<td>a != b</td>
<td>a is not equal to b</td>
</tr>
<tr>
<td>a &lt;= b</td>
<td>a is less than or equal to b</td>
</tr>
<tr>
<td>a &lt; b</td>
<td>a is less than b</td>
</tr>
</tbody>
</table>

Note: == sign!
Examples of Relational Operations

```java
int x = 3;
int y = 5;
boolean result;

1) result = (x > y);
result is assigned the value false because 3 is not greater than 5

2) result = (15 == x*y);
now result is assigned the value true because the product of 3 and 5 equals 15

3) result = (x != x*y);
now result is assigned the value true because the product of x and y (15) is not equal to x (3)
```
Conditional Operators

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>&amp;&amp;</td>
<td>AND</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>!</td>
<td>NOT</td>
</tr>
</tbody>
</table>

- Conditional operators can be referred to as **boolean** operators, because they are only used to combine expressions that have a value of **true** or **false**.
<table>
<thead>
<tr>
<th>$x$</th>
<th>$y$</th>
<th>$x &amp;&amp; y$</th>
<th>$x \mid\mid y$</th>
<th>$!x$</th>
</tr>
</thead>
<tbody>
<tr>
<td>True</td>
<td>True</td>
<td>True</td>
<td>True</td>
<td>False</td>
</tr>
<tr>
<td>True</td>
<td>False</td>
<td>False</td>
<td>True</td>
<td>False</td>
</tr>
<tr>
<td>False</td>
<td>True</td>
<td>False</td>
<td>True</td>
<td>True</td>
</tr>
<tr>
<td>False</td>
<td>False</td>
<td>False</td>
<td>False</td>
<td>True</td>
</tr>
</tbody>
</table>
Examples of Conditional Operators

boolean x = true;
boolean y = false;
boolean result;

Let result = (x && y);

result is assigned the value false

Let result = (((x || y) && x);

(x || y) evaluates to true
(true && x) evaluates to true

now result is assigned the value true
Using && and ||

- false && …
- true || …

- Java performs short circuit evaluation
  - Evaluate && and || expression s from left to right
  - Stop when you are guaranteed a value
Short-Circuit Evaluation

(a && (b++ > 3));

What happens if a is false?
- Java will not evaluate the right-hand expression (b++ > 3) if the left-hand operator a is false, since the result is already determined in this case to be false. This means b will not be incremented!

(x || y);

What happens if x is true?
- Similarly, Java will not evaluate the right-hand operator y if the left-hand operator x is true, since the result is already determined in this case to be true.
1) What is the value of result?
   int x = 8;
   int y = 2;
   boolean result = (15 == x * y);

2) What is the value of result?
   boolean x = 7;
   boolean result = (x < 8) && (x > 4);

3) What is the value of z?
   int x = 5;
   int y = 10;
   int z = y++ + x + ++y;
## Appendix I: Reserved Keywords

<table>
<thead>
<tr>
<th>abstract</th>
<th>assert</th>
<th>boolean</th>
<th>break</th>
<th>byte</th>
</tr>
</thead>
<tbody>
<tr>
<td>case</td>
<td>catch</td>
<td>char</td>
<td>class</td>
<td>const</td>
</tr>
<tr>
<td>continue</td>
<td>default</td>
<td>do</td>
<td>double</td>
<td>else</td>
</tr>
<tr>
<td>extends</td>
<td>final</td>
<td>finally</td>
<td>float</td>
<td>for</td>
</tr>
<tr>
<td>goto</td>
<td>if</td>
<td>implements</td>
<td>import</td>
<td>instanceof</td>
</tr>
<tr>
<td>int</td>
<td>interface</td>
<td>long</td>
<td>native</td>
<td>new</td>
</tr>
<tr>
<td>private</td>
<td>protected</td>
<td>public</td>
<td>return</td>
<td></td>
</tr>
<tr>
<td>short</td>
<td>static</td>
<td>strictfp</td>
<td>super</td>
<td>switch</td>
</tr>
<tr>
<td>synchronized</td>
<td>this</td>
<td>throw</td>
<td>throws</td>
<td>transient</td>
</tr>
<tr>
<td>try</td>
<td>void</td>
<td>violate</td>
<td>while</td>
<td></td>
</tr>
</tbody>
</table>
## Appendix II: Primitive Data Types

This table shows all primitive data types along with their sizes and formats:

<table>
<thead>
<tr>
<th>Data Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>byte</td>
<td>Variables of this kind can have a value from: -128 to +127 and occupy 8 bits in memory</td>
</tr>
<tr>
<td>short</td>
<td>Variables of this kind can have a value from: -32768 to +32767 and occupy 16 bits in memory</td>
</tr>
<tr>
<td>int</td>
<td>Variables of this kind can have a value from: -2147483648 to +2147483647 and occupy 32 bits in memory</td>
</tr>
<tr>
<td>long</td>
<td>Variables of this kind can have a value from: -9223372036854775808 to +9223372036854775807 and occupy 64 bits in memory</td>
</tr>
</tbody>
</table>
### Appendix II: Primitive Data Types

#### Real Numbers

<table>
<thead>
<tr>
<th>Data Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>float</td>
<td>Variables of this kind can have a value from: 1.4e(-45) to 3.4e(+38)</td>
</tr>
<tr>
<td>double</td>
<td>Variables of this kind can have a value from: 4.9e(-324) to 1.7e(+308)</td>
</tr>
</tbody>
</table>

#### Other Primitive Data Types

<table>
<thead>
<tr>
<th>char</th>
<th>Variables of this kind can have a value from: A single character</th>
</tr>
</thead>
<tbody>
<tr>
<td>boolean</td>
<td>Variables of this kind can have a value from: True or False</td>
</tr>
</tbody>
</table>