Lecture 2: Variables and Primitive Data Types

MIT-AITI Kenya 2005
In this lecture, you will learn…

• What a variable is
  – Types of variables
  – Naming of variables
  – Variable assignment

• What a primitive data type is

• Other data types (ex. String)
What is a Variable?

• In basic algebra, *variables* are symbols that can represent values in formulas.

• For example the variable $x$ in the formula $f(x) = x^2 + 2$ can represent any number value.

• Similarly, variables in computer program are symbols for arbitrary data.
A Variable Analogy

• Think of variables as an empty box that you can put values in.
• We can label the box with a name like “Box X” and re-use it many times.
• Can perform tasks on the box without caring about what’s inside:
  - “Move Box X to Shelf A”
  - “Put item Z in box”
  - “Open Box X”
  - “Remove contents from Box X”
Variables Types in Java

- Variables in Java have a *type*.
- The type defines what kinds of values a variable is allowed to store.
- Think of a variable’s type as the size or shape of the empty box.
- The variable $x$ in $f(x)=x^2+2$ is implicitly a number.
- If $x$ is a symbol representing the word “Fish”, the formula doesn’t make sense.
Java Types

• Integer Types:
  - *int*: Most numbers you’ll deal with.
  - *long*: Big integers; science, finance, computing.
  - *byte*: Very small integers, useful for generic data.

• Floating Point (Decimal) Types:
  - *float*: Single-precision decimal numbers
  - *double*: Double-precision decimal numbers.

• Other Types:
  - *String*: Text strings.
  - *boolean*: True or false.
  - *char*: Latin Alphanumeric Characters
Declaring Variables in Java

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

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

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

• We have not assigned values to these variables; just made empty boxes.
Assigning Values to Variables

• Assign **values** to variables using the syntax:
  - `name = value;`

• For example:
  - `x = 100;`
  - `greeting = "Jambo";`

• Illegal to assign a variable the wrong type:
  - `x = "Jambo";`
  - `x = 1.2;`
  - `greeting = 123;`

• Can declare and assign in one step:
  - `int x = 100;`
  - `String greeting = "Jambo";`
Naming Variables

• Variable names (or identifiers) may be any length, but must start with:
  – A letter (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.

• For example, “class”, “static”, “int”, etc.
Naming Variables

• Java is a case-sensitive - capitalization matters.
• 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”.
POP QUIZ

• Which of the following are valid variable names?

1. $amount
2. 6tally
3. my*Name
4. salary
5. _score
6. first Name
7. total#
8. short
Integer Types

• There are four primitive integer data types: byte, short, int, long.

• Each types has a maximum value, based on their binary representation:
  – Bytes: 8-bits, ± 128
  – Short: 16-bits, ± $2^{15} ≈ 32,000$
  – Int: 32-bits, ± $2^{31} ≈ 2$ billion
  – Long: 32-bits, ± $2^{63} ≈$ really big

• *Integer Overflows*: What happens if we store Bill Gates’ net worth in an int?
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 are constants and cannot be changed after they are created.
• Strings have a special append operator + that creates a new String:
  - String greeting = “Jam” + “bo”;
  - String bigGreeting = greeting + “!”;
Floating Point Types

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

• Use a trailing ‘d’ to force a value to be double:
  - double y = 1d/3; // y = .333333333
  - double z = 1/3; // z = 0.0 ... Why?

• Floats can be initialized like doubles, but need a trailing ‘f’:
  - float z = 1.23f;

• Doubles are more precise than Floats, but may take longer to perform operations.
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 characters such as a letter, number, punctuation mark, or other symbol.

• Characters are a single letter enclosed in single quotes. Don’t confuse with Strings.

• Example:
  - char firstLetterOfName = 'e' ;
  - char myQuestion = '?' ;
POP QUIZ

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

1. Population of Kenya  
   - int
2. World Population  
   - long
3. Approximation of π  
   - double
4. Open/closed status of a file  
   - boolean
5. Your name  
   - String
6. First letter of your name  
   - char
7. $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.
- Assigning a value to 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.
## Appendix I: Reserved Words

<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>interfac e</td>
<td>long</td>
<td>native</td>
<td>new</td>
</tr>
<tr>
<td>package</td>
<td>private</td>
<td>protected</td>
<td>public</td>
<td>return</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>
</table>
| byte      | Variables of this kind can have a value from: 
-128 to +127 and occupy 8 bits in memory |
| short     | Variables of this kind can have a value from: 
-32768 to +32767 and occupy 16 bits in memory |
| int       | Variables of this kind can have a value from: 
-2147483648 to +2147483647 and occupy 32 bits in memory |
| long      | Variables of this kind can have a value from: 
-9223372036854775808 to +9223372036854775807 and occupy 64 bits in memory |
# Appendix II: Primitive Data Types

## Real Numbers

<table>
<thead>
<tr>
<th>Data Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>float</code></td>
<td>Variables of this kind can have a value from: 1.4e(-45) to 3.4e(+38)</td>
</tr>
<tr>
<td><code>double</code></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>Data Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>char</code></td>
<td>Variables of this kind can have a value from: A single character</td>
</tr>
<tr>
<td><code>boolean</code></td>
<td>Variables of this kind can have a value from: True or False</td>
</tr>
</tbody>
</table>
EC.S01 Internet Technology in Local and Global Communities
Spring 2005-Summer 2005

For information about citing these materials or our Terms of Use, visit: http://ocw.mit.edu/terms.