

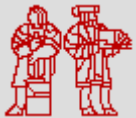


Lecture 2: Variables and Primitive Data Types

MIT-AITI Kenya 2007

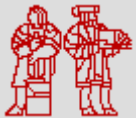
In this lecture, you will learn...

- **Variables**
 - **Types**
 - **Naming**
 - **Assignment**
- **Primitive data types**
- **Other data types**
- **Type casting**



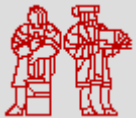
What is a Variable?

- In basic algebra, ***variables*** are symbols that can represent values in equations.
 - $y = x + 5$
- Similarly, ***variables*** in computer program represent data.
 - In computer programming, variables can represent more than just numbers



An Analogy

- Think of variables as a 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.
 - 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.
- Think of a variable’s type as the size or shape of the empty box.



Java Types

- Integer Types:
 - *int*: Most numbers you'll deal with.
 - *long*: Big integers; science, finance, computing.
 - *short*: Smaller integers. Not as useful.
 - *byte*: Very small integers, useful for small data.
- Floating Point (Decimal) Types:
 - *float*: Single-precision decimal numbers
 - *double*: Double-precision decimal numbers.
- Other Types:
 - *String*: Letters, words, or sentences.
 - *boolean*: True or false.
 - *char*: Single 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;`
- Note that we have not assigned values to these variables; we 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;`
- We can declare and assign in one step:
 - `int x = 100;`
 - `String greeting = "Jambo";`



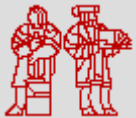
Naming Variables

- 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.
 - `int`, `double`, `String`, etc.



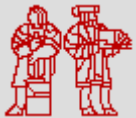
Naming Variables

- Java is 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”.



Review

- 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



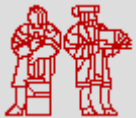
Integer Types

- There are four primitive integer data types: `byte`, `short`, `int`, `long`.
- Each types has a maximum value, based on their binary representation:
 - Bytes: ± 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)
- *Integer Overflows*: What happens if we store Bill Gates' net worth in an int?



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.
- We must be careful with integer division:
 - `double z = 1/3; // z = 0.0 ... Why?`
- Use a trailing 'd' to force a value to be double:
 - `double t = 1d/3; // t = .333333...`
- This is not always the best way
 - Need a way to do this for variables



Type Casting

- When we want to convert one type to another, we use **type casting**
- **The syntax is as follows:**

```
(new type)variable
```

- **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 characters 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`



Appendix I: Reserved Keywords

<code>abstract</code>	<code>assert</code>	<code>boolean</code>	<code>break</code>	<code>byte</code>
<code>case</code>	<code>catch</code>	<code>char</code>	<code>class</code>	<code>const</code>
<code>continue</code>	<code>default</code>	<code>do</code>	<code>double</code>	<code>else</code>
<code>extends</code>	<code>final</code>	<code>finally</code>	<code>float</code>	<code>for</code>
<code>goto</code>	<code>if</code>	<code>implements</code>	<code>import</code>	<code>instanceof</code>
<code>int</code>	<code>interfac e</code>	<code>long</code>	<code>native</code>	<code>new</code>
<code>package</code>	<code>private</code>	<code>protected</code>	<code>public</code>	<code>return</code>
<code>short</code>	<code>static</code>	<code>strictfp</code>	<code>super</code>	<code>switch</code>
<code>synchronized</code>	<code>this</code>	<code>throw</code>	<code>throws</code>	<code>transient</code>
<code>try</code>	<code>void</code>	<code>violate</code>	<code>while</code>	



Appendix II: Primitive Data Types

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

Data Type	Description
<code>byte</code>	Variables of this kind can have a value from: -128 to +127 and occupy 8 bits in memory
<code>short</code>	Variables of this kind can have a value from: -32768 to +32767 and occupy 16 bits in memory
<code>int</code>	Variables of this kind can have a value from: -2147483648 to +2147483647 and occupy 32 bits in memory
<code>long</code>	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

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

Other Primitive Data Types

<code>char</code>	Variables of this kind can have a value from: A single character
<code>boolean</code>	Variables of this kind can have a value from: <i>True or False</i>

