Variables, Types, Operations on Numbers

CSE 1310 – Introduction to Computers and Programming
University of Texas at Arlington

Updated 8/28/18
• See the summary document as well.
• Variable declaration, initialization, and use.
  – Possible syntax errors
    • Syntax – rules for code that can compile.

• Types:
  – Define what are legal values for a variable.
  – Both variables and values have types.
  – A variable can only have one type throughout the program.
  – Common errors:
    • Initialize a variable with a value of another type
    • Use data of a ‘wrong’ type for an operator(e.g. "catapult" - "cat" )
      – Wrong type: operator – is not defined for strings
    • Use data of a ‘wrong’ type as function argument
      – Function takes arguments of a different type.

• ++/--, +=/= (not recommended in exams)

• Type conversion:
  – Casting: (int)
  – Casting and: Math.round(), Math.floor() or Math.ceil()

• Constant variables: "final int DAYS_PER_WEEK = 7; “
• Literals – hardcoded data in a program
• Operators (+-/*%) and the Math class
Declaring a Variable

• A program uses *variables* to refer to data.
• You can think of a variable as a ‘box’ that holds data and has a label. You can only get to the box using the label.

• You create a variable, by doing a **variable declaration**.

• There are two ways to declare a variable:

  ```
  type variable_name;  // declare only: name and type
  type variable_name = initial_value;  // declare and initialize
  ```

• For example:

  ```
  int x;  //declaration only
  int numberOfFingers = 5;  //declare and initialize
  double radius = 20.231;
  ```
Declaration/Initialization before Use

• Java executes the code line-by-line from top to bottom.

• A variable must be declared before we try to use it.
• This code does not compile.
• Gives error: "cannot find symbol age".

```java
// incorrect code: Variable age is not declared before use.
age = 5;
System.out.println(age);
```

• A variable must be declared and initialized before we use it.
• This code does not compile.
• Gives error: " variable age might not have been initialized ".

```java
// incorrect code: Variable age is declared but not initialized.
int age;
System.out.println(age);
```
Solution 1: provide an initial value for the variable at the same line where you declare the variable.

See code below
- The red line declares and initializes a variable called `age1`.

```java
// correct code version 1
int age = 5;
System.out.println(age);
```

Solution 2: declare the variable in one line, and then set the value of the variable in another line.

See code below:
- The first red line declares a variable called `age2`.
- The second red line sets the value of `age2` to 5.

```java
// correct code version 2
int age;  // declared
age = 5;  // initialized
System.out.println(age);
```
Rules for Variable Names

• The textbook describes the rules for variable names.

• Here is a simplified version:
  – variable names should start with a letter (upper or lower case).
  – variable names should only include letters, numbers, and underscores.
  – variable names are case-sensitive.
  – variable names cannot be equal to reserved words, such as double, class, int, public, ...
Using Variables

• What is wrong with each piece of code below?

```java
hours = 120;
days = hours / 24;
System.out.println(days);
```

```java
double days = hours/24;
double hours = 120;
System.out.println(days);
```

```java
double hours = 120;
double days = Hours/24;
System.out.println(days);
```
Using Variables

• Answers:

```java
hours = 120;
days = hours / 24;
System.out.println(days);
```

```java
double hours = 120;
double days = hours / 24;
System.out.println(days);
```

```java
double days = hours / 24;
double hours = 120;
System.out.println(days);
```

```java
double hours = 120;
double days = hours / 24;
System.out.println(days);
```

```java
double hours = 120;
double days = Hours/24;
System.out.println(days);
```

```java
double hours = 120;
double days = hours/24;
System.out.println(days);
```
Types

• The **type** defines:
  – what are **legal values** for a variable of that type.
  – What operations can be done with data of that type (either hardcoded or from a variable)
  – (Extra: How the number is represented in binary)

• Java will never allow you to set a variable to a value incompatible with the type of the variable.

• **Both variables and values have a type.**
Five Types that we will use

• In this course we will use numbers and text (and boolean values).

• They will be represented by five basic types.
  – int
  – double
  – boolean
  – String
  – char
Five Types that we will use

- **int**
  - legal values? integers, like 0, 57, -1896...
  - Smallest: $-2^{31}$, largest $2^{31}-1$

- **double**
  - legal values? real numbers, like 3.0, 5.2, -0.23...
    - illegal: 5,300 (comma). Use: 5300

- **boolean**
  - legal values? only two: **true** and **false**.

- **String**
  - legal values? text, like "hello", "a cat jumped on the table", ...
  - NOTE: text for strings must be enclosed in **double quotes**.

- **char**
  - legal values? singe characters, like 'c', '3', 'A', '#', ...
  - NOTE: text for strings must be enclosed in **single quotes**.

Types Are **NOT** Interchangeable

- Not paying attention to types makes programming very hard and confusing.

- The following four values (also called literals) are NOT interchangeable:
  
  2
  
  2.0
  
  "2"
  
  '2'

- Why?
Types Are **NOT** Interchangeable

- Not paying attention to types makes programming very hard and confusing.

- The following four values are NOT interchangeable:
  2  this is an int
  2.0  this is a double
  "2"  this is a string
  '2'  this is a character

- Why? **Because they are different types.**
Types Are **NOT** Interchangeable

- For example:

<table>
<thead>
<tr>
<th>Incorrect</th>
<th>Correct</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>String a1 = 2.5;</code></td>
<td><code>String a1 = &quot;2.5&quot;;</code></td>
</tr>
<tr>
<td><code>double a2 = &quot;2.5&quot;;</code></td>
<td><code>double a2 = 2.5;</code></td>
</tr>
<tr>
<td><code>int num = '5';</code></td>
<td><code>int num = 5;</code></td>
</tr>
<tr>
<td><code>char c1 = 5;</code></td>
<td><code>char c1 = '5';</code></td>
</tr>
<tr>
<td><code>String str = '5';</code></td>
<td><code>String str = &quot;5&quot;;</code></td>
</tr>
<tr>
<td><code>int my_int = 2.0;</code></td>
<td><code>int my_int = 2;</code></td>
</tr>
<tr>
<td><code>boolean v = &quot;true&quot;;</code></td>
<td><code>boolean v = true;</code></td>
</tr>
<tr>
<td><code>String v = true;</code></td>
<td><code>String v = &quot;true&quot;;</code></td>
</tr>
</tbody>
</table>
Converting *double* values to *int* values

You cannot store a value of type double in a variable of type int (a ‘box’ for int).

```java
double price = 18.53;
int dollars = price;  // Java gives an error
```

There are 4 ways to convert a value of type *double* to a value of type *int*.

<table>
<thead>
<tr>
<th>Description</th>
<th>Examples</th>
</tr>
</thead>
<tbody>
<tr>
<td>Casting (removes the decimal part)</td>
<td>int n = (int) 18.53;  // n is 18</td>
</tr>
<tr>
<td>Round up</td>
<td>int n = (int) Math.ceil (18.53);  // n is 19</td>
</tr>
<tr>
<td></td>
<td>int n = (int) Math.ceil (18.22);  // n is 19</td>
</tr>
<tr>
<td>Round down</td>
<td>int n = (int) Math.floor (18.53);  // n is 18</td>
</tr>
<tr>
<td></td>
<td>int n = (int) Math.floor (18.99);  // n is 18</td>
</tr>
<tr>
<td>Round (to closest integer)</td>
<td>int n = (int) Math.round (18.53);  // n is 19</td>
</tr>
</tbody>
</table>

Student question: “n is already of type int. Why do I need to convert to an int?”

- You cannot store a value of type double in a ‘box’ for int.
- You are converting the value (18.53) to an int value and then it can be stored in a ‘box’ for int.
Constants

• Some variables should never change value.

• Examples:
  – Number of days in a week.
  – Constants from math and physics such as pi and e.
  – Other data specific to your application (e.g. minimum number of staff at a desk during a work day)

• To tell Java that a variable is a constant, use the keyword `final` when you declare the variable.

Syntax:
```
final type variable_name = value;
```

```
int weeks = 12;
final int DAYS_PER_WEEK = 7;
int days = weeks * DAYS_PER_WEEK;
System.out.println(days);
```
Using Constants – correct practice

• Using a **constant** (declared with **final**) has these advantages:
  
  – The **code is easier to read and understand**
    
    • It is clear that DAYS_PER_WEEK will not change throughout the program
    • There is a meaningful name, as opposed to just the number 7.
  
  – The **compiler will check and report errors** such as modifying DAYS_PER_WEEK.

```java
final int DAYS_PER_WEEK = 7;
int weeks = 12;
DAYS_PER_WEEK ++;  // Java prevents it. It will give an error
int days = weeks * DAYS_PER_WEEK;
System.out.println(days);
```
Literals, constants, variables

- **Literals** are hard-coded data. E.g.: 2, "Hello", 3.7
- Constants are special variables that cannot change value and have the keyword ‘final’ in their declaration
  - Constants are names given to literals
- **Variables**

- Hard-coded data is data that cannot change as the program runs (e.g. instead of getting user input, the program simply uses the same data value every time it runs).
Math Operators and the Math class

• Operators: + - * / %
  – * is multiplication
  – % is the remainder of integer division
  – / between 2 integers does integer division
  – / between an integer and a real number gives correct answer (with decimals)

• Math class provides useful mathematical functions:
  – Math.PI,
  – Math.floor(4.7),
  – Math.min(4,1), Math.max(4,1)
The ++ and -- Operators

• The ++ operator increments the value of a variable by 1.

• Syntax:
  \texttt{variable\_name++};

• The -- operator decrements the value of a variable by 1.

• Syntax:
  \texttt{variable\_name--};
The `+=` and `-=` operators

```java
public class example1 {
    public static void main(String[] args) {
        double x = 5.5;
        x += 3.2;  // same as x = x + 3.2;
        int y = 20;
        y -= 5;    // same as y = y - 5;
        System.out.println(x);
        System.out.println(y);
    }
}
```

- The `+=` operator adds some value to a variable.
- Syntax:
  ```java
  variable_name += value;
  ```
- The `-=` operator subtracts some value from a variable.
- Syntax:
  ```java
  variable_name -= value;
  ```
- Whether you use `+=` and `-=` or not is entirely up to you.
  - Do NOT use these in an exam (miss a symbol => wrong answer)

Output
8.7
15
Multiple Ways to Add/Subtract 1

• If we want to add 1 to x, in how many ways can we do it?
  x++;  
  x += 1;  
  x = x+1;

• If we want to subtract 1 from x, in how many ways can we do it?
  x--;  
  x -= 1;  
  x = x-1;