Introduction to Java Basics

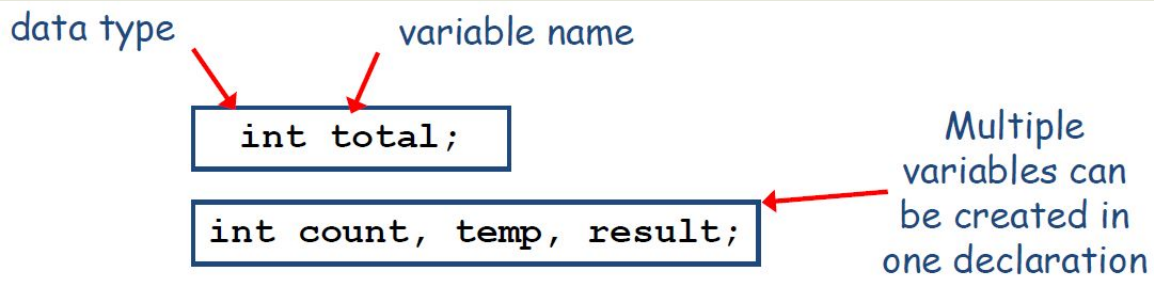
Programming in Java

Contents

1. Comments
2. Identifiers
3. Indentation
4. Primitive Types
5. Variables
6. Output
7. Numeric Console Input
8. Assignment
9. Arithmetic Expressions
10. More Assignment Operators
11. Assignment Compatibility
12. Strings
13. String Console Input

5- Variables

- A name for a location in memory
- Used to store information (ex: price, size, …)
- Must be declared before it is used
  - indicate the variable's name
  - indicate the type of information it will contain
  - declaration can be anywhere in the program (but before its first access)



5- Variables: Initialize Variables

- A variable that has been declared but has not yet been given a value is said to be ***uninitialized***
- <u>In certain cases</u> an uninitialized variable is given a default value
  - It is best not to rely on this
  - Explicitly initialized variables have the added benefit of improving program clarity

5- Variables: default values

- For type int, the default value is zero, that is, 0.
- For type long, the default value is zero, that is, 0L.
- For type float, the default value is positive zero, that is, 0.0f.
- For type double, the default value is positive zero, that is, 0.0d.
- For type char, the default value is the null character, that is, ‘\u0000’.
- For type boolean, the default value is false.

5- Variables: Initialize at declaration

- A variable can be given an initial value in the declaration

```
int sum = 0;
int base = 32, max = 149;
```

- When a variable is used in a program, its current value is used

5- Variables: Example: PianoKeys.java

```
//*******************************************************
//  PianoKeys.java
//  Demonstrates the declaration and initialization of an
//  integer variable.
//*******************************************************
public class PianoKeys
{
   //  Prints the number of keys on a piano.
   public static void main (String[] args)
   {
      int keys = 88;

      System.out.println("A piano has" + keys + "keys.");
   }
}
```

filename??

???

Output

5- Variables: Constants

- Similar to a variable but can only hold one value while the program is active
- The compiler will issue an error if you try to change the value of a constant during execution
- Use the **final** modifier

```
final int MIN_AGE = 18;
```

- Constants:
  - give names to otherwise unclear literal values
  - facilitate updates of values used throughout a program
  - prevent inadvertent attempts to change a value

6- Output

System.out.print
Displays what is in parenthesis

System.out.println
Displays what is in parenthesis
Advances to the next line

Examples:

```
System.out.print("hello");
System.out.print("you");

System.out.println("hello");
System.out.println("you");

System.out.println();

int price = 50;
System.out.print(price);

char initial = 'L';
System.out.println(initial);
```

Output

6- Output: multiple outputs

```
System.out.println("hello" + "you");
double price = 9.99;
int nbItems = 5;
System.out.println("total = " + price*nbItems + "$");
```

```
???
```

Output

in **print** and **println**, + is the concatenation…
you need parenthesis for the + to be addition

```
int x = 1, y = 2;
System.out.println("x+y="+x+y);
System.out.println("x+y="+(x+y));
```

```
???
```

Output

6- Output: multiple outputs

cannot *cut* a string over several lines

```
System.out.println("this is a
                    long string"); // error!

System.out.println("this is a" +
                    "long string"); // ok
```

6- Output: Escape sequences

- to print a double quote character

```
System.out.println ("I said "Hi" to her.");
```

```
???
```

Output

- Use an escape sequence
  - sequence is a series of characters that represents a special character
  - begins with a backslash character (
  - considered as 1 single character

```
System.out.println ("I said \"Hi\" to her.");
```

```
???
```

Output

6- Output: Escape sequences

- Some Java escape sequences:

Escape Sequence	Meaning
`\b`	backspace
`\t`	tab
`\n`	newline
`\"`	double quote
`\'`	single quote
`\\`	backslash

6- Output: Your turn!

- What will the following statement print?

```
System.out.print("one\ntwo\nthree\n");
```

A) one two three

B) one\ntwo\nthree\n

C) "one\ntwo\nthree\n"

D) one
two
three

E) onetwothree

6- Output: Your turn!

- What will the following statement print?

```
System.out.print("one\ntwo\nthree\n");
```

A) one two three

B) one\ntwo\nthree\n

C) "one\ntwo\nthree\n"

D) one
two
three

E) onetwothree

6- Output: Your turn!

- What statement will result in the following print?

```
Read the file "c:\windows\readme.txt"
```

System.out.print

A) (“Read the file “c:\windows\readme.txt”);

B) (“Read the file “c:\windows\readme.txt””);

C) (“Read the file “c:\\windows\\readme.txt”);

D) (“Read the file

\”c:\\windows\\readme.txt\””);

E) (“Read the file \”c:\\ windows\\readme.txt \"");

6- Output: Your turn!

- What statement will result in the following print?

```
Read the file "c:\windows\readme.txt"
```

System.out.print

A) (“Read the file “c:\windows\readme.txt”);

B) (“Read the file “c:\windows\readme.txt””);

C) (“Read the file “c:\\windows\\readme.txt”);

D) (“Read the file

\”c:\\windows\\readme.txt\””);

E) (“Read the file \”c:\\ windows\\readme.txt \"");

7- Numeric Console Input: Class Libraries & Packages

- A *class library* (ex. Java Standard class library) is a collection of classes that we can use when developing programs
- The classes of the Java standard class library are organized into packages

Package	Purpose
java.lang	General support
java.text	Format text for output
java.applet	Creating applets for the web
java.awt	Graphics and graphical user interfaces
java.util	Utilities

7- Numeric Console Input: The import Declaration

- To use a class from a package
  - You can import only the class DecimalFormat from the package java.text

```
import java.text.DecimalFormat;
```

  - Or import all classes from the package java.text

```
import java.text.*;
```

- All classes of the java.lang package are imported automatically into all programs
- That's why we didn't have to import the **System** or **String** classes explicitly

7- Numeric Console Input: Class Libraries & Packages

- Since Java 5.0, use the **Scanner** class
- The keyboard is represented by the **System.in** object

```
import java.util.Scanner;
.
.
.
Scanner myKeyboard = new Scanner(System.in);
```

7- Numeric Console Input: To read from a scanner

- To read tokens, use a nextSomething() method
  - nextBoolean (),
  - nextByte ,
  - nextInt (),
  - nextFloat (),
  - nextDouble (),
  - next(), Will see later in the chapter
  - nextLine(), Will see later in the chapter
  - ...

tokens are delimited by whitespaces (blank spaces, tabs, and line breaks)

```
Scanner myKeyboard = new Scanner(System.in);
System.out.println("Your name:");
String name = myKeyboard.next();
System.out.println("Welcome " + name + " Enter your age:");
int age = myKeyboard.nextInt();
```

7- Numeric Console Input: Example 1 (ScannerDemo1.java)

```
//*****************************************************
// Author: W. Savitch
//
//  This program demonstrates how to read numeric tokens from
//  the console with the Scanner class
//*****************************************************
import java.util.Scanner; // we need to import this class

public class ScannerDemo1
{
   public static void main(String[] args)
   {
      // let's declare our scanner
      Scanner keyboard = new Scanner(System.in);
```

7- Numeric Console Input: Example 1 (ScannerDemo1.java)

```
// let's ask the user for some input
  System.out.println("Enter the number of pods followed by");
  System.out.println("the number of peas in a pod:");
// let's read the user input (2 integers that we assign to 2
// variables)
int numberOfPods = keyboard.nextInt( );
int peasPerPod = keyboard.nextInt( );
int totalNumberOfPeas = numberOfPods*peasPerPod;
```

7- Numeric Console Input: Example 1 (ScannerDemo1.java)

```
    // let's display some output
     System.out.print(numberOfPods + " pods and ");
     System.out.println(peasPerPod + " peas per pod.");
     System.out.println("The total number of peas = "
                                + totalNumberOfPeas);
     // close the Scanner
     keyboard.close();
     } // end of main()
   } // end of class ScannerDemo1
```

7- Numeric Console Input: Example 2 (ScannerDemo2.java)

```
//*****************************************************
// Author: W. Savitch
//
//  This program demonstrates how to read various types
// of token with the Scanner class
//*****************************************************
import java.util.Scanner;
public class ScannerDemo2
{
 public static void main(String[] args)
 {
   // let's try to read integers
   int n1, n2; // let's declare 2 variables for our tests
   // let's declare our scanner object
   Scanner scannerObject = new Scanner(System.in);
```

7- Numeric Console Input: Example 2 (ScannerDemo2.java)

```
// let's read two whole numbers (integers)
System.out.println("Enter two whole numbers ");
System.out.println("separated by one or more spaces:");

// we read 1 integer and assign it to n1
n1 = scannerObject.nextInt( );

// we read another integer and assign it to n2
n2 = scannerObject.nextInt( );

System.out.println("You entered " + n1 + " and " + n2);
```

7- Numeric Console Input: Example 2 (ScannerDemo2.java)

```
    System.out.println("Next enter two numbers with a decimal
point.");
    System.out.println("Decimal points are allowed.");
    // let's try to read doubles now
    double d1, d2;
    d1 = scannerObject.nextDouble( );
    d2 = scannerObject.nextDouble( );
    System.out.println("You entered " + d1 + " and " + d2);
    // close the Scanner
     scannerObject.close();
  } // end of main()
} // end of class ScannerDemo2
```

7- Numeric Console Input: close() method

- Good habit to close a Scanner object, at the end of your program
  - Ex: Say created a Scanner object called keyIn as follows
- Good habit to close a Scanner object, at the end of your program

```
Scanner keyIn = new Scanner(System.in);
```

Good habit to include the following statement just before the } last for the main method

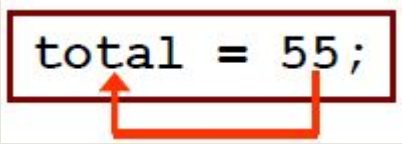
```
keyIn.close();
```

Will keep Eclipse happy! :)

8- Assignment

- Used to change the value of a variable
- The assignment operator is the = sign
- Syntax: Variable = Expression;
- Semantics:

  1. the expression on the right is evaluated
  2. the result is stored in the variable on the left (overwrite any previous)
  3. The entire assignment expression is worth the value of the RHS



8- Assignment: Example

```
public class Geometry
{
   //  Prints the number of sides of several geometric shapes.
   public static void main (String[] args)
   {
     int sides = 7;  // declaration with initialization
     System.out.println("A heptagon has " + sides + " sides.");

     sides = 10;  // assignment statement
     System.out.println("A decagon has " + sides + " sides.");
     sides = 10+2;
     System.out.println("A dodecagon has " + sides + " sides.");
   }
}
```

filename???

```
???
```

Output

8- Assignment: difference with the math =

- In Java, = is an operator
- In math, = is an equality relation

```
In math… a+6 = 10   ok
In Java…  a+6 = 10;
```

```
In math… a = a+1  always false
In Java… a = a+1;
```

```
In math… a = b and b = a     same thing!
In Java… a = b;  and b = a;
```

8- Assignment: Difference with the math =

- Declarations:

```
int x;
int y = 10;
char c1 = 'a';
char c2 = 'b';
```

- Statements:

```
x = 20+5;
y = x;
c1 = 'x';
c2 = c1;
```

8- Assignment: Swap content of 2 variables

- Write a series of declarations & statements to swap the value of 2 variables…

```
int  x = 10;  int  y = 20;
```

A) x = y; y = x;

B) y = x; x = y;

C) Both A) & B) will work

D) Neither A) nor B) will work

9- Arithmetic Expressions

- An ***expression*** is a combination of one or more operands and their operators
- ***Arithmetic*** operators:

Addition	+
Subtraction	-
Multiplication	*
Division	/
Remainder	%

9- Arithmetic Expressions: Division and Remainder

The division operator (/) can:

- Integer division
  - if both operands are integers

```
10 / 8      equals?   1
8 / 12      equals?   0
```

- Real division
  - otherwise

```
10.0 / 8      equals?     1.25
8 / 12.0      equals?   0.6667
```

9- Arithmetic Expressions: Division and Remainder

The remainder operator (%) returns the remainder after the integer division

10 % 8	equals?	2	(10÷8 = 1 remainder 2)
8 % 12	equals?	8	

9- Arithmetic Expressions: Operator Precedence

- Operators can be combined into complex expressions

```
result  =  total + count / max - offset;
```

- Precedence determines the order of evaluation
  - 1st: expressions in parenthesis
  - 2nd: unary + and -
  - 3rd: multiplication, division, and remainder
  - 4th: addition, subtraction, and string concatenation
  - 5th: assignment operator

9- Arithmetic Expressions: Operator Associativity

- **<u>Unary</u>** operators of equal precedence are grouped right to left

  +-+rate is evaluated as +(-(+rate))

- **<u>Binary</u>** operators of equal precedence are grouped left-to-right

  base + rate + hours is evaluated as (base + rate) + hours

- Exception: Exception: A string of assignment operators is grouped right-to-left

  n1 = n2 = n3; is evaluated as (n1 = (n2 = n3));

9- Arithmetic Expressions: Operator Associativity

- What is the order of evaluation in the following expressions?

```
a + b + c + d + e
```

```
a + b * c - d / e
```

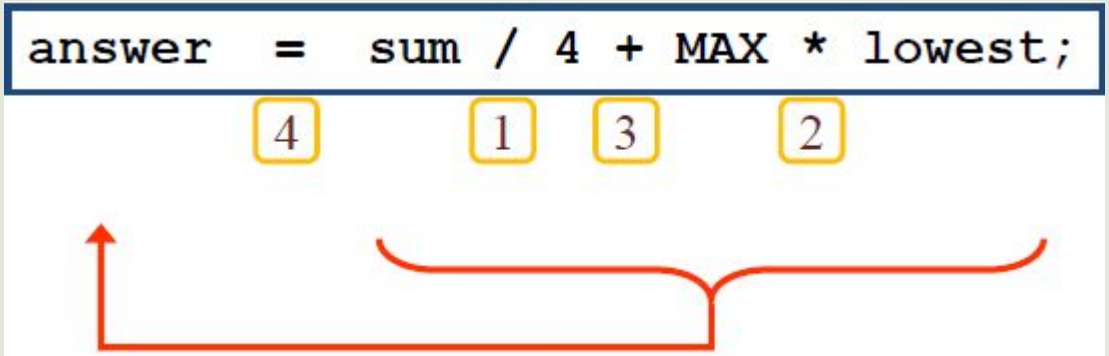
```
a / (b + c) - d % e
```

```
a / (b * (c + (d - e)))
```

9- Arithmetic Expressions: Assignment Revisited

- The assignment operator has a lower precedence than the arithmetic operators

First the expression on the RHS is evaluated



Then the result is stored in the variable on the LHS

9- Arithmetic Expressions: Exercise

- What is stored in the integer variable num1 after this statement?

```
num1 = 2 + 3 * 5 - 5 * 2 / 5 + 10;
```

A) 0

B) 18

C) 25

D) 10

9- Arithmetic Expressions: Exercise

- What is stored in the integer variable num1 after this statement?

```
num1 = 2 + 3 * 5 - 5 * 2 / 5 + 10;
```

A) 0

B) 18

C) 25

D) 10