# Programming Fundamentals

# Objectives

At the end of the lesson, the student should be able to:

- Identify the basic parts of a Java program
- Differentiate among Java literals, primitive data types, variable types ,identifiers and operators
- Develop a simple valid Java program using the concepts learned in this chapter

# Dissecting my First Java Program

```
public class Hello
{
   /**
    * My first Java program
    */
   public static void main( String[] args ){

         //prints the string Hello world on screen
         System.out.println(“Hello world”);

   }
}
```

# Coding Guidelines

1. Your Java programs should always end with the `.java` extension.

2. Filenames should match the name of your public class. So for example, if the name of your public class is Hello, you should save it in a file called `Hello.java`.

3. You should write comments in your code explaining what a certain class does, or what a certain method does.

# Java Comments

- Comments
  - These are notes written to a code for documentation purposes.
  - Those texts are not part of the program and does not affect the flow of the program.

# Java Identifiers

- Identifiers
  - are tokens that represent names of variables, methods, classes, etc.
  - Examples of identifiers are: Hello, main, System, out.

- Java identifiers are case-sensitive.
  - This means that the identifier **Hello** is not the same as **hello**.

# Java Identifiers

- Identifiers must begin with either a letter, an underscore "_", or a dollar sign "$". Letters may be lower or upper case. Subsequent characters may use numbers 0 to 9.

- Identifiers cannot use Java keywords like class, public, void, etc. We will discuss more about Java keywords later.

# Java Identifiers
# Coding Guidelines

1. For names of classes, capitalize the first letter of the class name. For example,

   ThisIsAnExampleOfClassName

2. For names of methods and variables, the first letter of the word should start with a small letter. For example,

   thisIsAnExampleOfMethodName

# Java Identifiers
# Coding Guidelines

3. In case of multi-word identifiers, use capital letters to indicate the start of the word except the first word. For example,

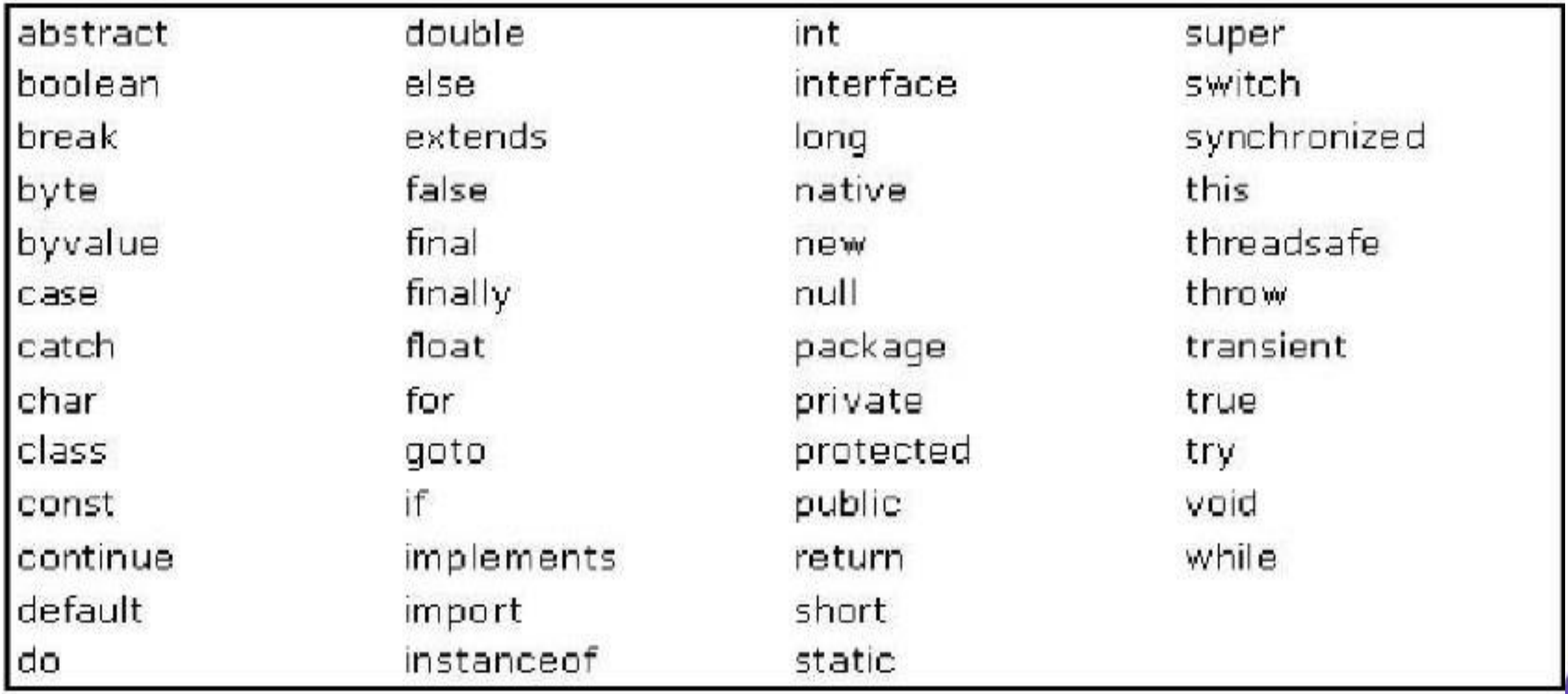
   charArray, fileNumber, ClassName.

4. Avoid using underscores at the start of the identifier such as _read or _write.

# Java Keywords

- Keywords are predefined identifiers reserved by Java for a specific purpose.
- You cannot use keywords as names for your variables, classes, methods ... etc.
- The next slide contains the list of the Java Keywords.

# Java Keywords



abstract
boolean
break
byte
byvalue
case
catch
char
class
const
continue
default
do
double
else
extends
false
final
finally
float
for
goto
if
implements
import
instanceof
int
interface
long
native
new
null
package
private
protected
public
return
short
static
super
switch
synchronized
this
threadsafe
throw
transient
true
try
void
while

# Java Literals

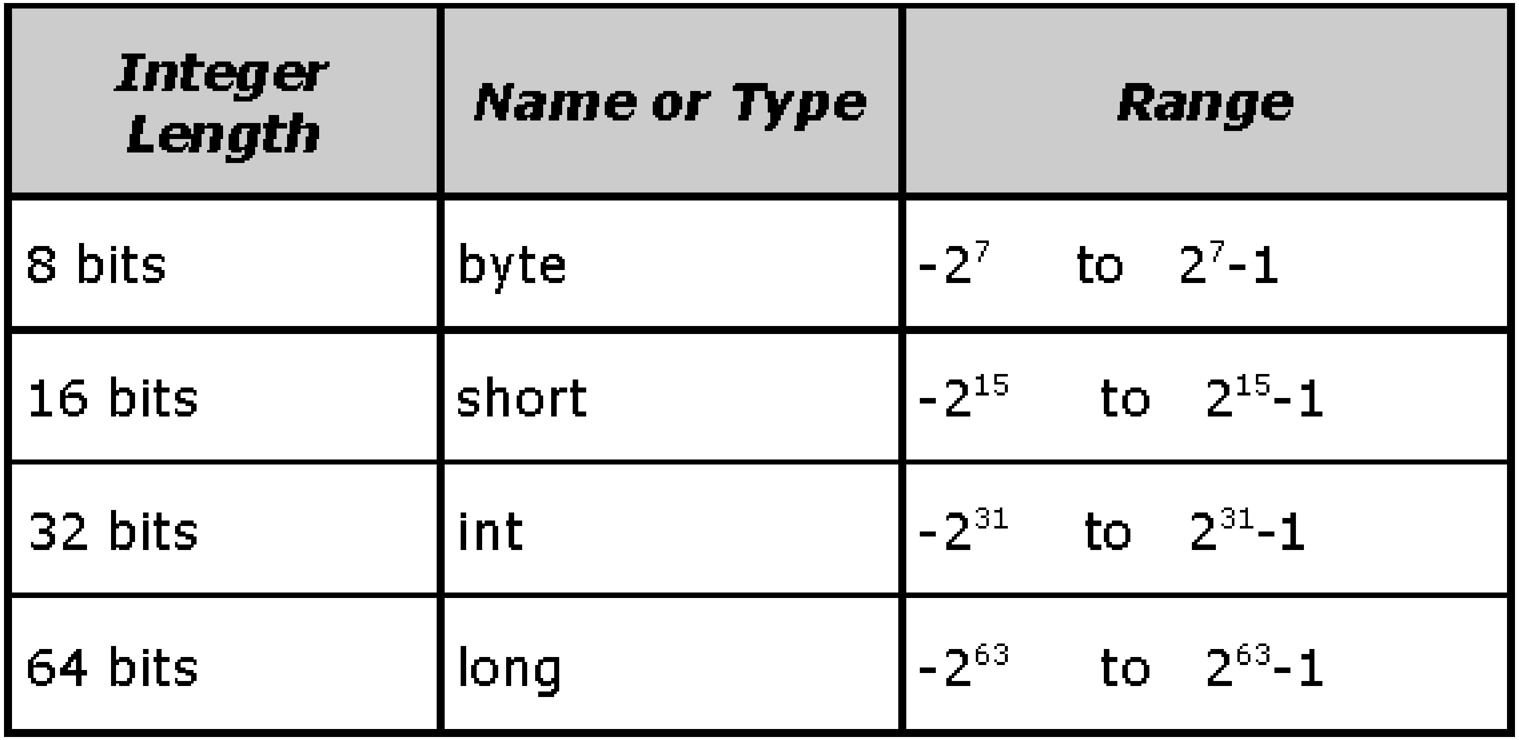
- Literals are tokens that do not change - they are constant.
- The different types of literals in Java are:
  - Integer Literals
  - Floating-Point Literals
  - Boolean Literals
  - Character Literals
  - String Literals

# Primitive Data Types

- The Java programming language defines eight primitive data types.
  - boolean (for logical)
  - char (for textual)
  - byte
  - short
  - int
  - long (integral)
  - double
  - float (floating point).

# Primitive Data Types: Integral – byte, short, int & long

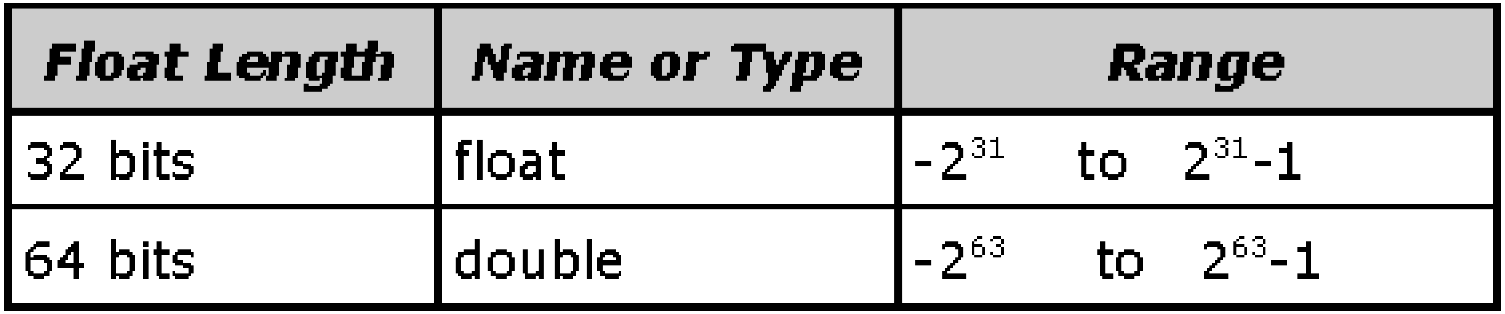
- Integral data type have the following ranges:



| *Integer Length* | *Name or Type* | *Range* |
|---|---|---|
| 8 bits | byte | $-2^{7}$ to $2^{7}-1$ |
| 16 bits | short | $-2^{15}$ to $2^{15}-1$ |
| 32 bits | int | $-2^{31}$ to $2^{31}-1$ |
| 64 bits | long | $-2^{63}$ to $2^{63}-1$ |

# Primitive Data Types: Floating Point – float and double

- Floating-point data types have the following ranges:



| Float Length | Name or Type | Range |
|---|---|---|
| 32 bits | float | $-2^{31}$ to $2^{31}-1$ |
| 64 bits | double | $-2^{63}$ to $2^{63}-1$ |

# Variables

- A variable is an item of data used to store the state of objects.

- A variable has a:
  - data type
    - The data type indicates the type of value that the variable can hold.
  - name
    - The variable name must follow rules for identifiers.

# Declaring and Initializing Variables

- Declare a variable as follows:

```
<data type>  <name> [=initial value];
```

- Note: Values enclosed in <> are required values, while those values in [] are optional.

# System.out.println() vs. System.out.print()

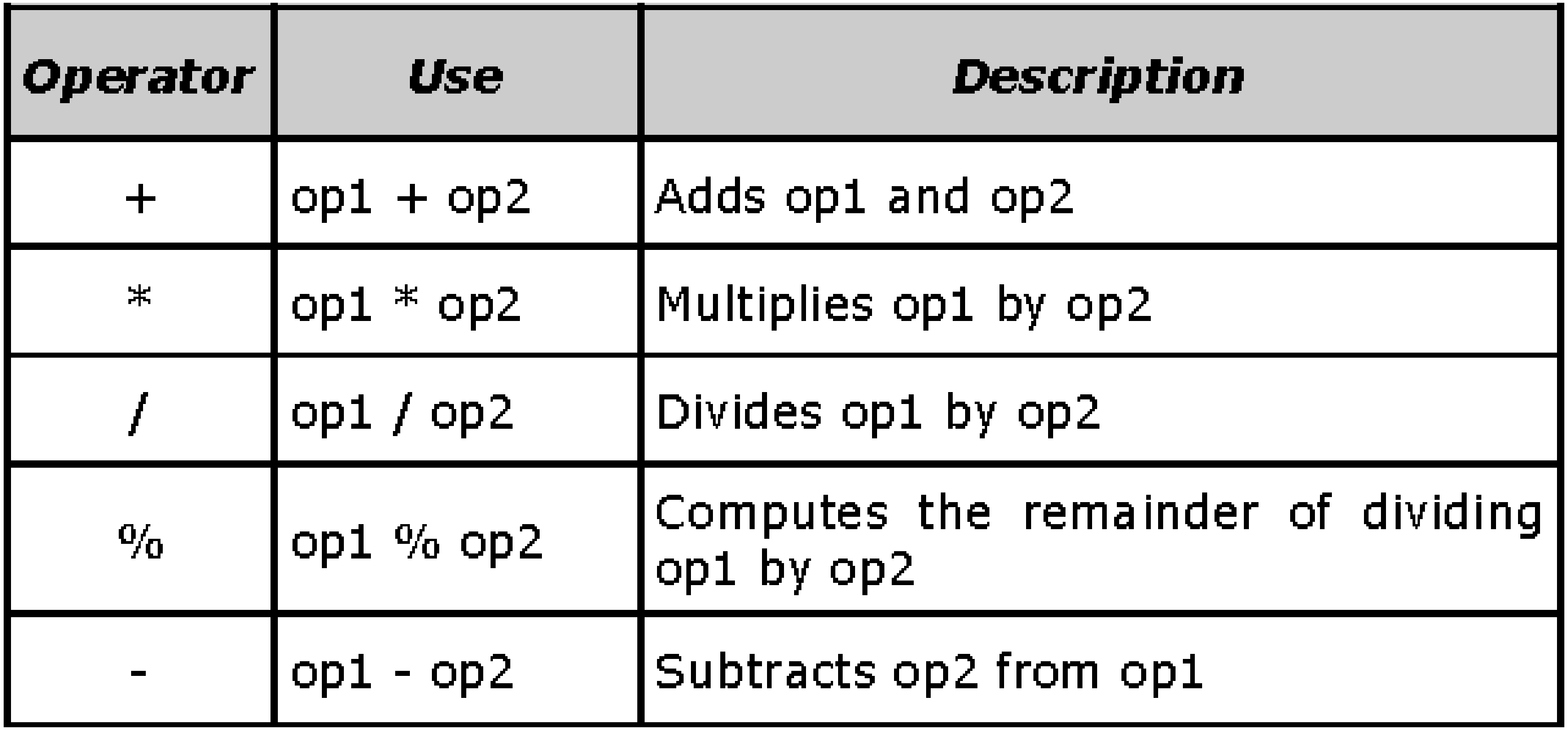
- System.out.println()
  - Appends a newline at the end of the data output
- System.out.print()
  - Does not append newline at the end of the data output

# Operators

- Different types of operators:
  - arithmetic operators
  - relational operators
  - logical operators
  - conditional operators

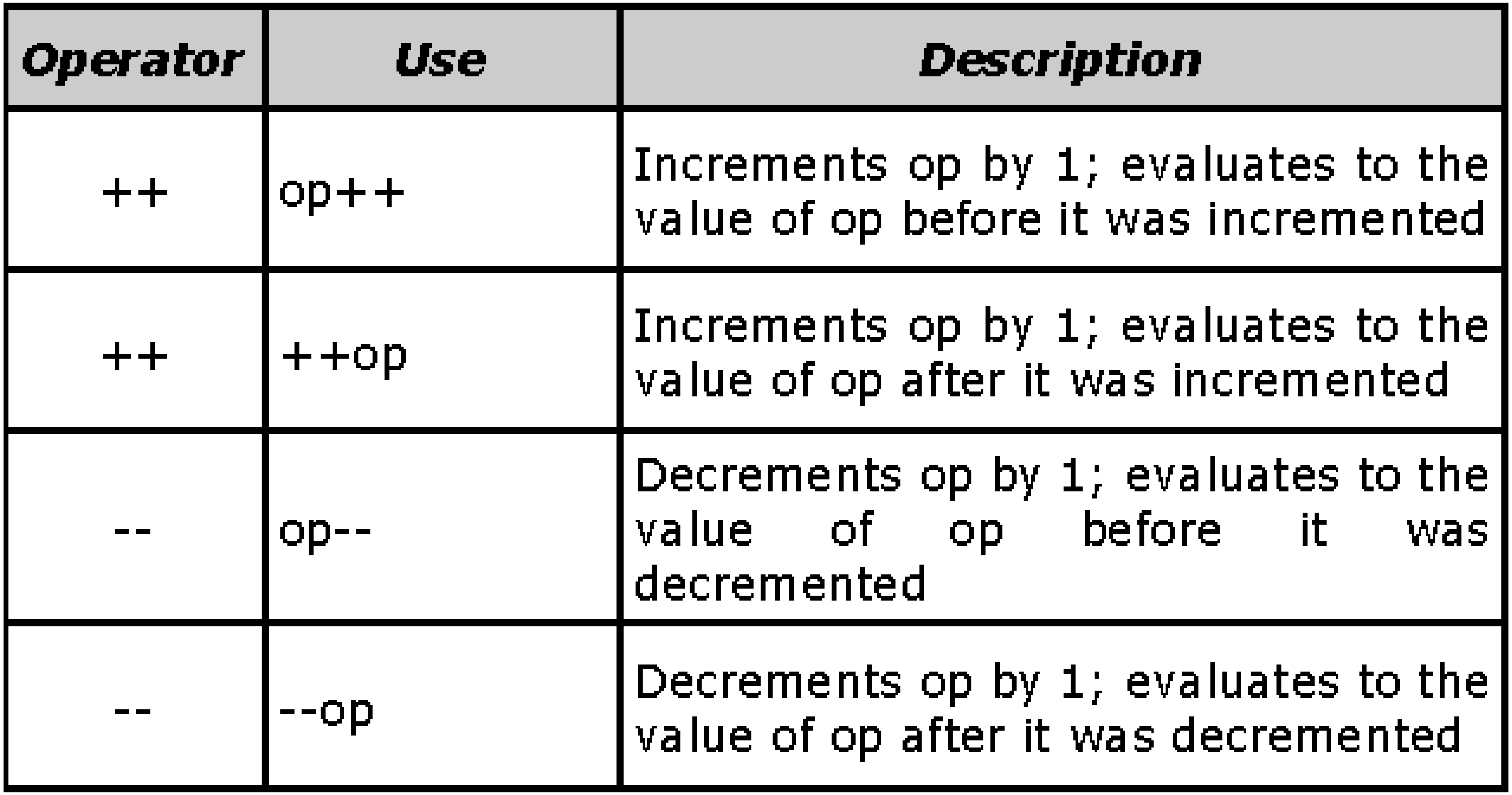
- These operators follow a certain kind of precedence so that the compiler will know which operator to evaluate first in case multiple operators are used in one statement.

# Arithmetic Operators



| Operator | Use | Description |
|---|---|---|
| + | op1 + op2 | Adds op1 and op2 |
| * | op1 * op2 | Multiplies op1 by op2 |
| / | op1 / op2 | Divides op1 by op2 |
| % | op1 % op2 | Computes the remainder of dividing op1 by op2 |
| - | op1 - op2 | Subtracts op2 from op1 |

# Increment and Decrement Operators



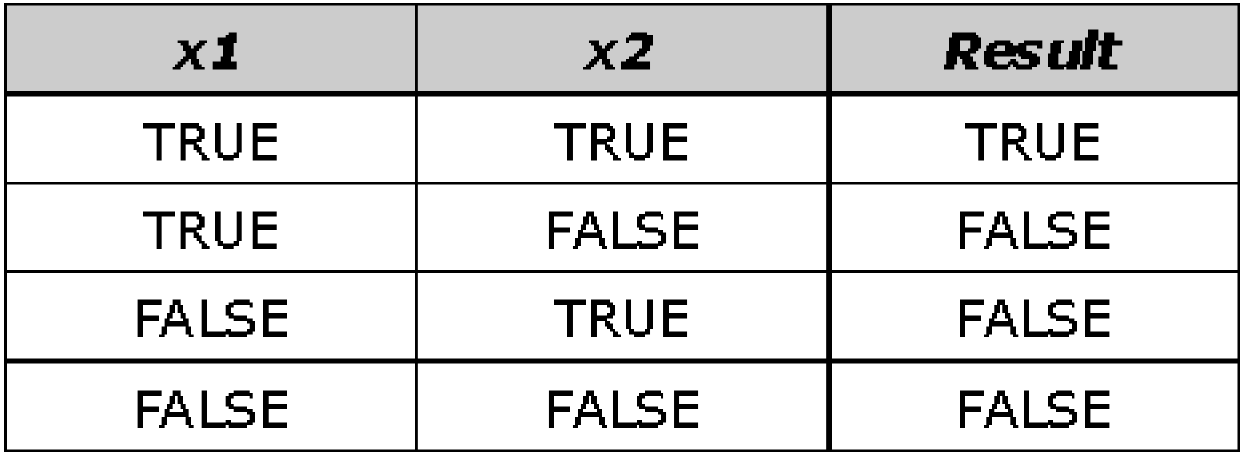
| Operator | Use | Description |
|---|---|---|
| ++ | op++ | Increments op by 1; evaluates to the value of op before it was incremented |
| ++ | ++op | Increments op by 1; evaluates to the value of op after it was incremented |
| -- | op-- | Decrements op by 1; evaluates to the value of op before it was decremented |
| -- | --op | Decrements op by 1; evaluates to the value of op after it was decremented |

# Logical Operators

- Logical operators have one or two boolean operands that yield a boolean result.
- There are six logical operators:
  - && (logical AND)
  - & (boolean logical AND)
  - || (logical OR)
  - | (boolean logical inclusive OR)
  - ^ (boolean logical exclusive OR)
  - ! (logical NOT)

# Logical Operators: &&(logical) and &(boolean logical) AND

- Here is the truth table for && and&,



| *x1* | *x2* | *Result* |
|---|---|---|
| TRUE | TRUE | TRUE |
| TRUE | FALSE | FALSE |
| FALSE | TRUE | FALSE |
| FALSE | FALSE | FALSE |

# Summary

- Java Comments (C++-Style Comments, C-Style Comments, Special Javadoc Comments)
- Java statements, blocks, identifiers, keywords
- Java Literals (integer, floating point, boolean, character, String)
- Primitive data types( boolean, char, byte, short, int, long, float, double)

# Summary

- Variables (declare, initialize, output)
- System.out.println() vs. System.out.print()
- Reference Variables vs. Primitive Variables
- Operators (Arithmetic operators, Increment and Decrement operators, Relational operators, Logical operators, Conditional Operator (?:), Operator Precedence)