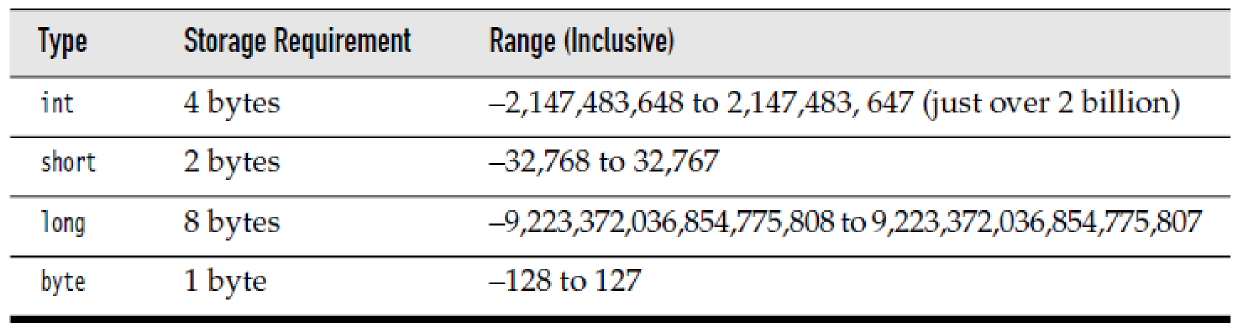
JAVA Basics _ Part 1

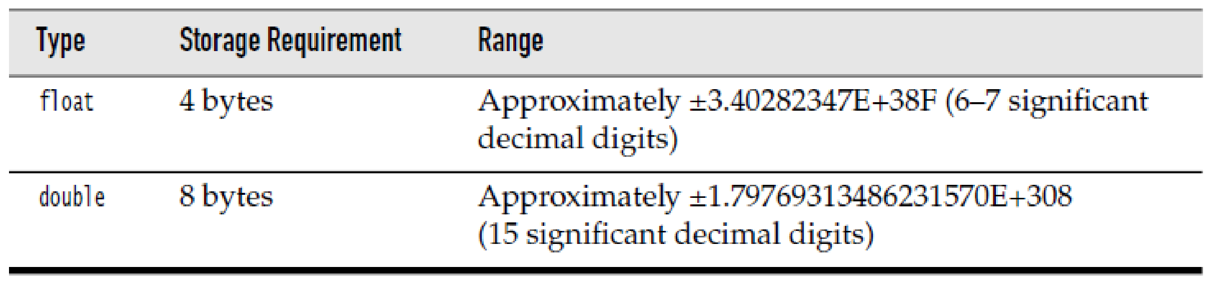
Java is a *strongly typed language*. This means that every variable must have a declared type. There are eight *primitive types* in Java.
Four of them are integer types;
two are floating-point number types;
one is the character type char,
and one is a boolean type for truth values.

Table 3.1 Java Integer Types



Type	Storage Requirement	Range (Inclusive)
`int`	4 bytes	–2,147,483,648 to 2,147,483, 647 (just over 2 billion)
`short`	2 bytes	–32,768 to 32,767
`long`	8 bytes	–9,223,372,036,854,775,808 to 9,223,372,036,854,775,807
`byte`	1 byte	–128 to 127

Table 3.2 Floating-Point Types



Type	Storage Requirement	Range
`float`	4 bytes	Approximately ±3.40282347E+38F (6–7 significant decimal digits)
`double`	8 bytes	Approximately ±1.79769313486231570E+308 (15 significant decimal digits)

Variables

In Java, every variable has a type. You declare a variable by placing the type first,Followed by the name of the variable. Here are some examples:

```
double salary;
int vacationDays;
long earthPopulation;
boolean done;
```

A variable name must begin with a letter and must be a sequence of letters or digits

You can declare multiple variables on a single line:

```
int i, j; // both are integers
```

Initializing Variables

After you declare a variable, you must explicitly initialize it by means of an assignment statement—you can never use the value of an uninitialized variable.

```
int vacationDays;
System.out.println(vacationDays); // ERROR--variable not initialized
```

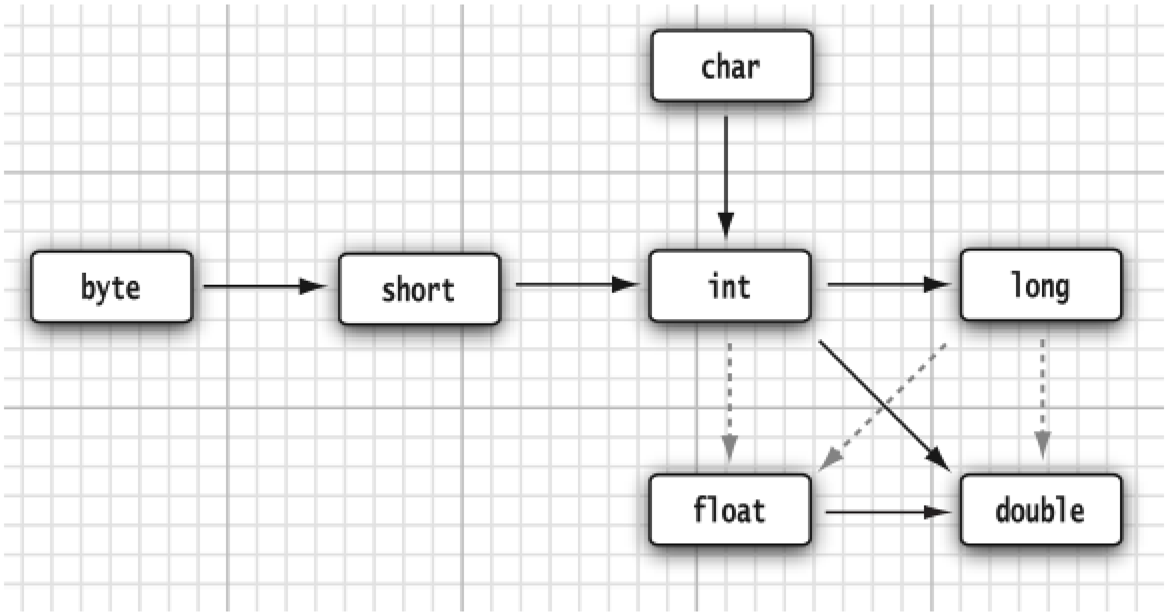
You can both declare and initialize a variable on the same line. For example:

```
int vacationDays = 12;
```

Operators

The usual arithmetic operators +, -, *, / are used in Java for addition, subtraction, multiplication, and division.

The / operator denotes integer division if both arguments are integers, and floating-point division otherwise



Numeric conversions are possible in Java, but of course information may be lost. Conversions in which loss of information Is possible are done by means of casts. The syntax for casting is to give the target type in parentheses, followed by the variable name. For example:

```
double x = 9.997;
int nx = (int) x;
```

Now, the variable nx has the value 9 because casting a floating-point value to an integer discards the fractional part.

Combining Assignment with Operators

There is a convenient shortcut for using binary operators in an assignment.

For example,

```
x += 4;
```

is equivalent to

```
x = x + 4;
```

Example for arithmetic operators

```
class BasicMath {
public static void main(String args[]) {
// arithmetic using integers
System.out.println("Integer Arithmetic");
int a = 1 + 1;
int b = a * 3;
int c = b / 4;
int d = c - a;
int e = -d;
System.out.println("a = " + a);
System.out.println("b = " + b);
System.out.println("c = " + c);
System.out.println("d = " + d);
System.out.println("e = " + e);
// arithmetic using doubles
System.out.println("\nFloating Point Arithmetic");
double da = 1 + 1;
double db = da * 3;
double dc = db / 4;
double dd = dc - a;
double de = -dd;
System.out.println("da = " + da);
System.out.println("db = " + db);
System.out.println("dc = " + dc);
System.out.println("dd = " + dd);
System.out.println("de = " + de);
}
}
```

The modulus operator, %, returns the remainder of a division operation

```
// Demonstrate the % operator.
class Modulus {
public static void main(String args[]) {
int x = 42;
System.out.println("x mod 10 = " + x % 10);
}
}
```

Strings

Conceptually, Java strings are sequences of Unicode characters

```
String e = ""; // an empty string
String greeting = "Hello";
```

You can extract a substring from a larger string with the substring method of the string class. For example,

```
String greeting = "Hello";
String s = greeting.substring(0, 3);
```

Creates a string consisting of the characters "Hel".

Java, like most programming languages, allows you to use + to join (concatenate) two strings.

Control Flow

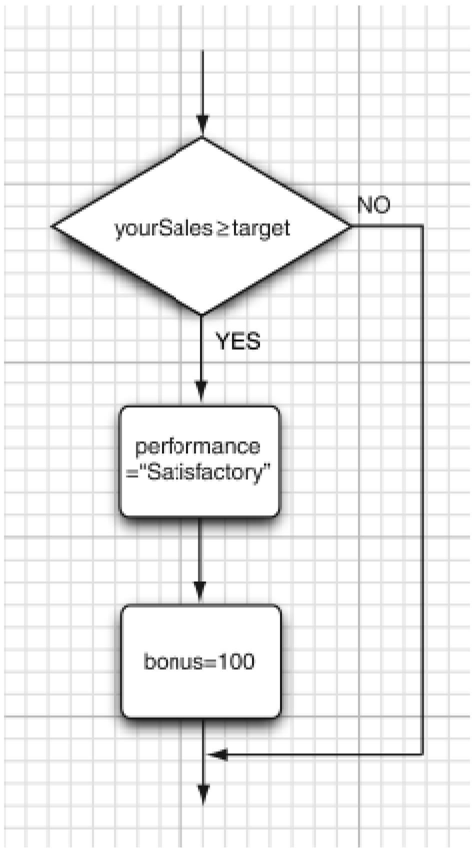
Java, like any programming language, supports both conditional statements and loops to determine control flow

The conditional statement in Java has the form

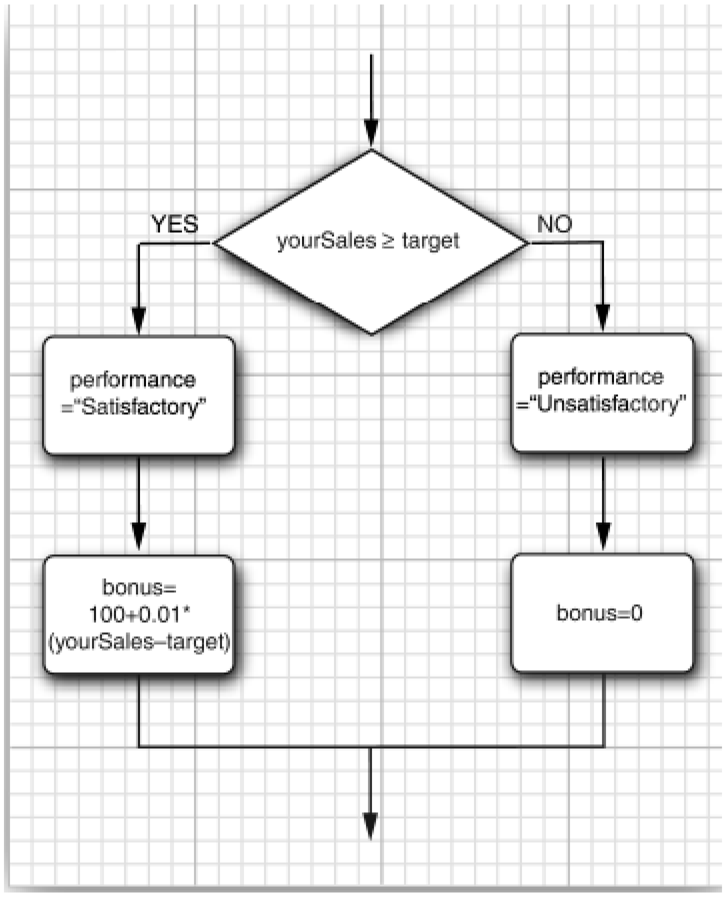
```
if (condition) statement
```

The condition must be surrounded by parentheses.

```
if (yourSales >= target)
{
performance = "Satisfactory";
bonus = 100;
}
```



The more general conditional in Java looks like this
if (condition) statement1
else
statement2


yourSales ≥ target
YES
NO
performance ="Satisfactory"
performance ="Unsatisfactory"
bonus= 100+0.01* (yourSales–target)
bonus=0

```
if (yourSales >= target)
{
performance = "Satisfactory";
bonus = 100 + 0.01 * (yourSales - target);
}
else
{
performance = "Unsatisfactory";
bonus = 0;
}
```

```
class IfElse {
public static void main(String args[]) {
int month = 4; // April
String season;
if(month == 12 || month == 1 || month == 2)
season = "Winter";
else if(month == 3 || month == 4 || month == 5)
season = "Spring";
else if(month == 6 || month == 7 || month == 8)
season = "Summer";
else if(month == 9 || month == 10 || month == 11)
season = "Autumn";
else
season = "Bogus Month";
System.out.println("April is in the " + season + ".");
}
}
```