Operators and Statements

Master Dev SS4

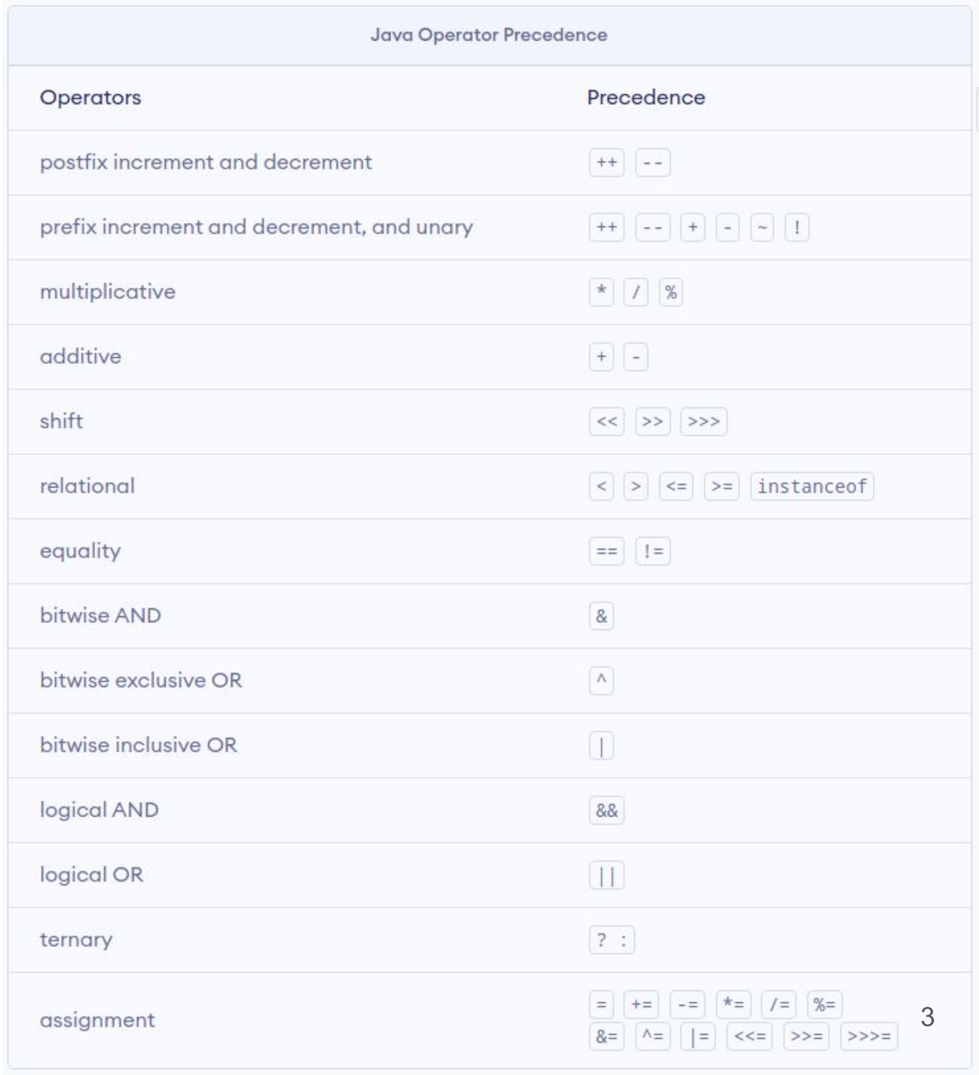
Java Operators

- What is operand?
- What is operator?
- Three flavors of operators: unary, binary, and ternary.

E.g: What are the final x and y values?

```
int y = 3;

double x = 3 + 3 * --y;
```

Java Operators



Java Operator Precedence

Operators	Precedence
postfix increment and decrement	`++` `--`
prefix increment and decrement, and unary	`++` `--` `+` `-` `~` `!`
multiplicative	`*` `/` `%`
additive	`+` `-`
shift	`<<` `>>` `>>>`
relational	`<` `>` `<=` `>=` `instanceof`
equality	`==` `!=`
bitwise AND	`&`
bitwise exclusive OR	`^`
bitwise inclusive OR	`|`
logical AND	`&&`
logical OR	`||`
ternary	`? :`
assignment	`=` `+=` `-=` `*=` `/=` `%=` `&=` `^=` `|=` `<<=` `>>=` `>>>=`

Numeric Promotion

What is the data type of x * y ?

```
int x = 9;
long y = 3;
```

What is the data type of a * b / c ?

```
short a = 3;
float b = 8;
double c = 12;
```

Primitive Numeric Promotion

- ➢ If two values have different data types, Java will automatically promote one of the values to the larger of the two data types.
- ➢ If one of the values is integral and the other is floating-point, Java will automatically promote the integral value to the floating-point values data type.
- ➢ Smaller data types, namely byte , short , and char , are first promoted to int any time they’re used with a Java binary arithmetic operator, even if neither of the operands is int .
- ➢ After all promotion has occurred and the operands have the same data type, the resulting value will have the same data type as its promoted operands.

Primitive Numeric Promotion

E.g 1: What is the data type of the following expressions?

```
int a = 6; long b  = 3; float c = 2.5F; short d = 2;
a + b                   a / d
a * c
```

E.g 2: What data type (or types) will allow the following code snippet to compile?

```
byte x = 5;

byte y = 10;

______ z = x + y;
```

A. int B. long C. boolean D. double E. short F. byte

Assignment Operators

E.g:

```
int x = 1.0;

short y = 1921222;

int z = 9F;

long t = 192301398193810323;
```

Casting Primitive Values

E.g:

```
int x = 1.0;                          int x = (int)1.0;

short y = 1921222;                    short y = (short)1921222;

int z = 9F;                           int z = (int)9F;

long t = 192301398193810323;          long t = 192301398193810323L;
```

➔ Casting primitives is required any time you are going from a larger numerical data type to a smaller numerical data type, or converting from a floating-point number to an integral value.

Compound Assignment Operators

➢ = , += , -= , *= , /= , %= , &= , ^= , != , <<= , >>= , >>>=

Ex:

```
int x = 2, z = 3;
x = x * z; // Simple assignment operator
x *= z;     // Compound assignment operator
```

```
long x = 10;
int y = 5;
y = y * x;  // DOES NOT COMPILE
y *= x;
```

- ➔ Only be applied to a variable that is already defined and cannot be used to declare a new variable.
- ➔ Compiler will automatically cast the resulting value to the data type of the value on the left-hand side of the compound operator

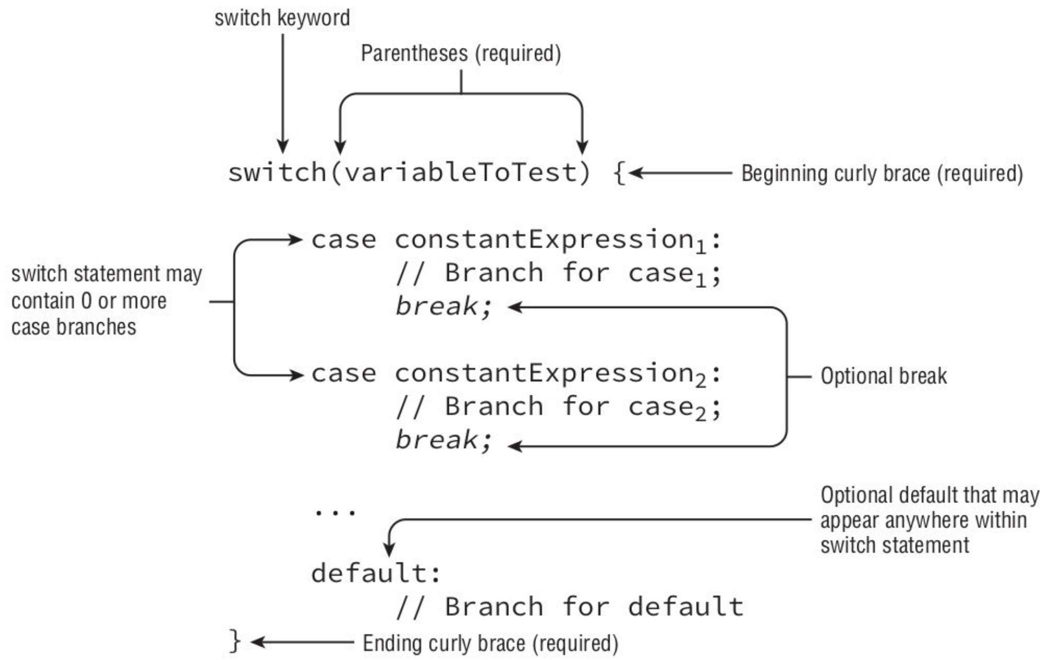
Equality Operators

- ➢ Comparing two numeric primitive types.
- ➢ Comparing two boolean values.
- ➢ Comparing two objects, including null and String values.

Java Statements

- ➢ The if-then Statement
- ➢ The if-then-else Statement
- ➢ The switch Statement
- ➢ The while Statement
- ➢ The do-while Statement
- ➢ The for Statement

The switch Statement


switch keyword
Parentheses (required)
Beginning curly brace (required)
switch statement may contain 0 or more case branches
Optional break
Optional default that may appear anywhere within switch statement
Ending curly brace (required)

```
switch(variableToTest) {
    case constantExpression1:
        // Branch for case1;
        break;
    case constantExpression2:
        // Branch for case2;
        break;
    ...
    default:
        // Branch for default
}
```

The switch Statement

Data types supported by switch statements include:

- ➢ int and Integer
- ➢ byte and Byte
- ➢ short and Short
- ➢ char and Character
- ➢ String
- ➢ enum values

The values in each case statement must be compile-time constant values of the same data type as the switch value.

- ➢ literals
- ➢ enum constants
- ➢ final constant variables

What are literals in Java?

- A literal is a source code representation of a fixed value.

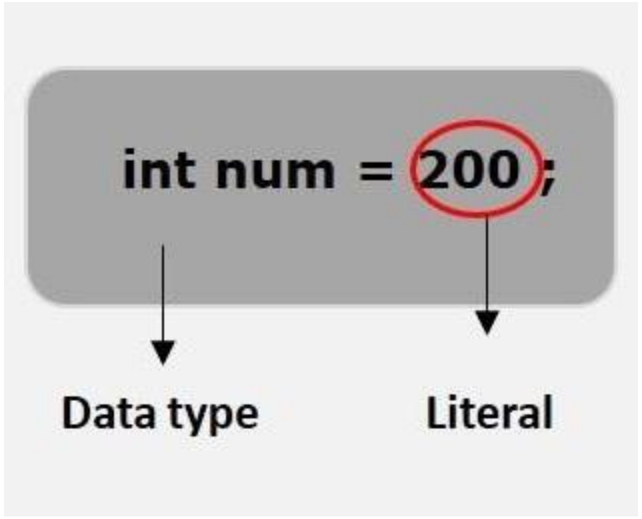
E.g

int a = 100;

long b = 68L;

char c =’J’;

String d = “Hello World”;

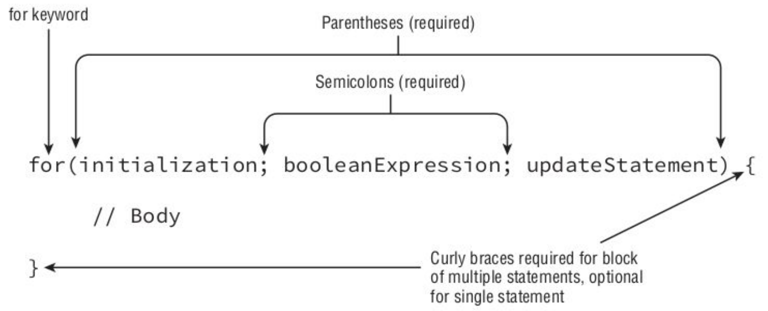


Enum Types

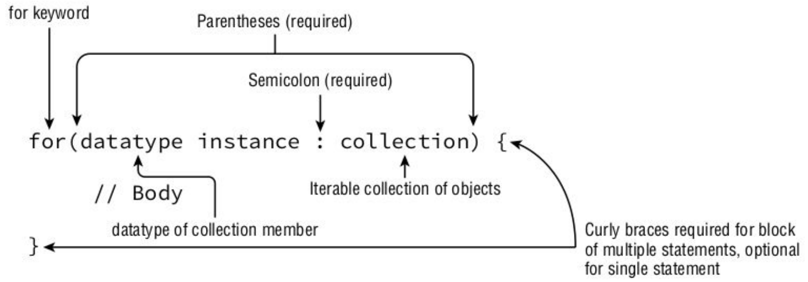
- ➢ An enum type is a special data type that enables for a variable to be a set of predefined constants.
- ➢ All enums implicitly extend java.lang.Enum.
- ➢ The constructor for an enum type must be package-private or private access. It automatically creates the constants that are defined at the beginning of the enum body. You cannot invoke an enum constructor yourself.

The for Statement

The Basic for Statement


for keyword
Parentheses (required)
Semicolons (required)
for(initialization; booleanExpression; updateStatement) {
// Body
}
Curly braces required for block of multiple statements, optional for single statement

The for-each Statement

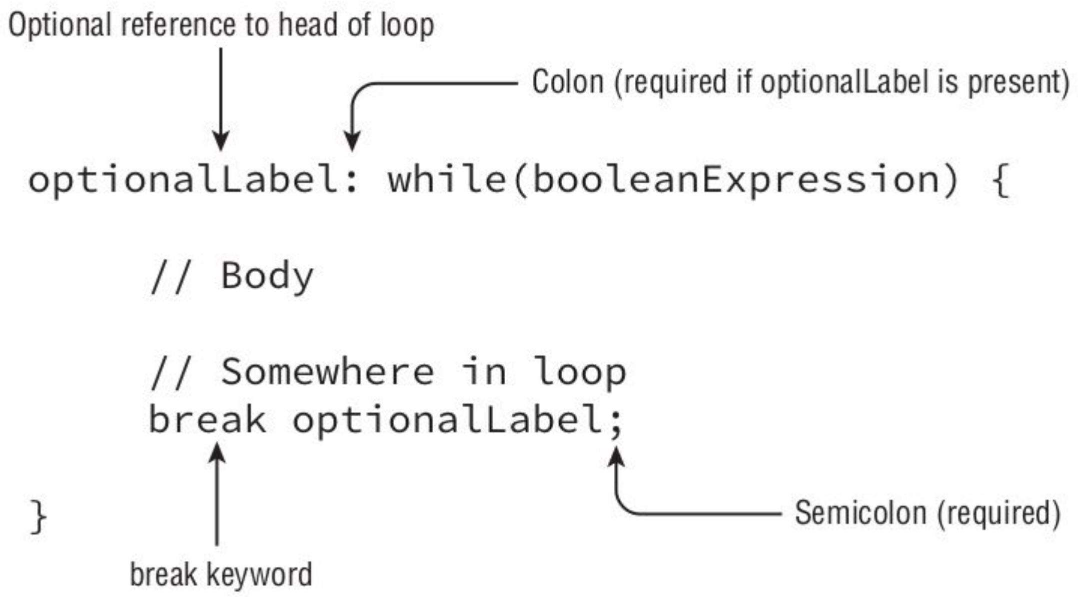

for keyword
Parentheses (required)
Semicolon (required)
for(datatype instance : collection) {
// Body
}
Iterable collection of objects
datatype of collection member
Curly braces required for block of multiple statements, optional for single statement

The for Statement

- With var names is String array with data:

```
for(String name : names) {
        System.out.print(name + ", ");
}

for(int i=0; i < names.length; i++) {
        String name = names[i];
        System.out.print(name + ", ");
}
```

The break Statement


Optional reference to head of loop
Colon (required if optionalLabel is present)
Semicolon (required)
break keyword

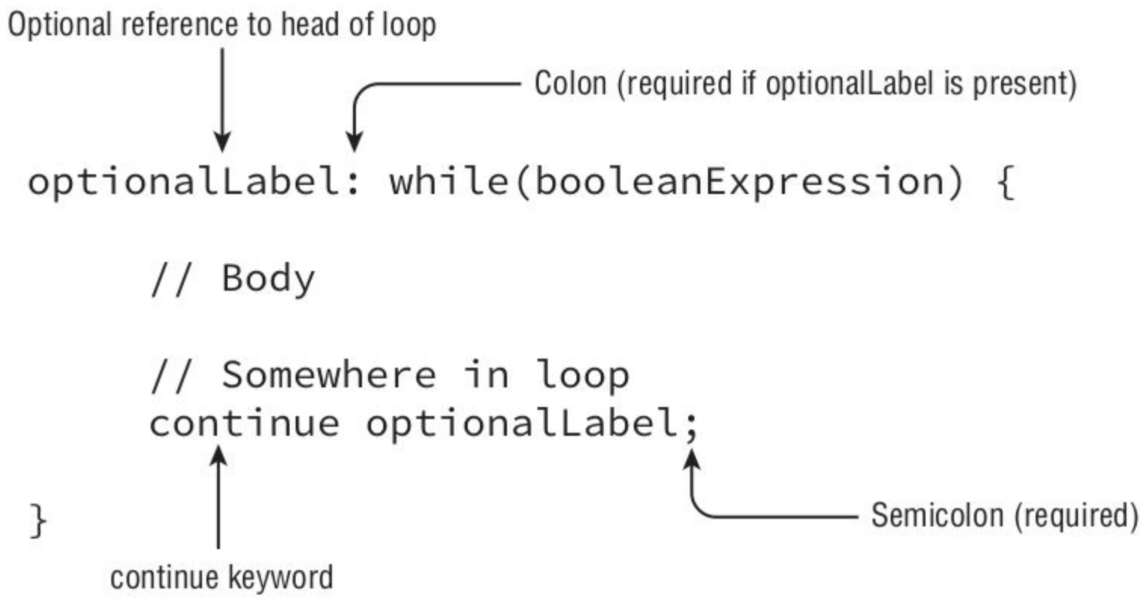
```
optionalLabel: while(booleanExpression) {

     // Body

     // Somewhere in loop
     break optionalLabel;

}
```

The break Statement

E.g: What are the values of positionX and positionY?

```
int[][] list = {{1,13,5},{1,2,5},{2,7,2}};
int searchValue = 2;
int positionX = -1;
int positionY = -1;
PARENT_LOOP: for(int i=0; i<list.length; i++) {
        for(int j=0; j<list[i].length; j++) {
                if(list[i][j]==searchValue) {
                        positionX = i;
                        positionY = j;
                        break PARENT_LOOP;
                }
        }
}
```

The continue Statement


Optional reference to head of loop
Colon (required if optionalLabel is present)
Semicolon (required)
continue keyword

```
optionalLabel: while(booleanExpression) {

     // Body

     // Somewhere in loop
     continue optionalLabel;

}
```

The continue Statement

E.g: What will the output show?

```
FIRST_CHAR_LOOP: for (int a = 1; a <= 4; a++) {
        for (char x = 'a'; x <= 'c'; x++) {
                if (a == 2 || x == 'b') {
                        continue FIRST_CHAR_LOOP;
                }
                System.out.print(" " + a + x);
        }
}
```

Bài tập thực hành

1. Viết một chương trình nhận vào 1 trong 12 tháng làm input. Output là số ngày trong tháng từ input. Giả sử tháng 2 luôn chỉ có 28 ngày. Yêu cầu làm theo 2 cách:
   Cách 1: Sử dụng enum và cấu trúc switch.
   Cách 2: Không sử dụng enum.
2. Viết chương trình tính ra số ngày giữa 2 mốc thời điểm người dùng nhập vào. Dữ liệu nhập vào của người dùng sẽ có dạng y1 m1 d1 y2 m2 d2. Với điều kiện mốc thời gian 1 lớn hơn mốc thời gian 2.
3. Viết một chương trình tính tổng các số nguyên tố từ 1 đến 10_000 với 3 cách. Mỗi cách yêu cầu sử dụng 1 cấu trúc lặp khác nhau bao gồm: while, do while và for.