Programming Fundamentals

Operators and Expressions

Week 3| Iresh Bandara

Learning Outcomes

- Covers part of LO1 & LO2 for Module
- On completion of this lecture, students are expected to be able to:
  - Classify different types of operators in java.
  - Identify the correct return types for a method.
  - Develop various functions in java to perform various operations.

Operators in Java

- A symbol that tells the computer to perform a mathematical or logical manipulation.
- Used in programs to manipulate data and variables.

Types of Operators

- Arithmetic operators
- Relational operators
- Logical operators
- Assignment operators
- Increment and decrement operators
- Conditional operators
- Bitwise operators
- Special operators

Arithmetic Operators

+ - * / %

- The type of the result is determined by the types of the operands, not their values.
  - this rule applies to all intermediate results in expressions.
- If one operand is an int and another is a double, the result is a double; if both operands are ints, the result is an int.

Integer Arithmetic

- When both operands are integers, the expression is an integer expression, the operation is integer arithmetic.
- For modulo division (%), the sign of the result is always the sign of the first operand.

(Note that module division is *defined as: a % b =* { *a - (a/b) * b* } *where a/b* is *the integer division).*

Real Arithmetic

- When both operands are real, the expression is a real expression, the operation is real arithmetic.
- For modulo division (%), the operator returns the floating point equivalent of an integer division.
- Floating point values are rounded to the number of significant digits permitted.

Mixed-mode Arithmetic

- When one of the operands is real (floating) and the other is integer, the expression is called a mixed-mode arithmetic expression.
- If either operand is of the real type, then the real arithmetic is performed.

Rational Operators

<	<=
>	>=
==	!=

- Used to compare two quantities, and depending on their relation to take decisions.
- Expressions containing relational operators are relational expressions.

Relational Expression

ae-l ***relational operator*** **ae-2**

- “ae-1” and “ae-2” are arithmetic expressions
- Value of relational expression - true or false
- Arithmetic expressions are evaluated first and then the results compared.
- Relational expressions are used in decision statements – **if**, **for** and **while**

Logical Operators

&&	logical AND
||	logical OR
!	logical NOT

- Used to combine two or more relational expressions and such are called as logical expressions.
- Value of a logical expression - true or false

Logical Operators cont...

(condition1 **&&** condition2)

is true if and only if both condition1 and condition2 are true

(condition1 **||** condition2)

is true if and only if condition1 or condition2 (or both) are true

! condition1

is true if and only if condition1 is false

Exercise one

- Assuming that x = 2, y = 6 , and

  z = 3, specify whether the result is true or false.
  - (x > z) && (y > z)
  - (x <= 5) ||(y > 2)|| (z == 6)
  - (x ==2)
  - (x == 3|| ((y > 5) && (z > 2))

Assignment Operators

- Used to assign the value of an expression to a variable.
- Usual assignment operator **=**
- Shorthand assignment operators are:

+= , -= , *=, /=, %=

Increment and Decrement Operators

- Operators are: **++** and **--**
- The operator ++ adds 1 to the operand
- The operator -- subtracts 1 from the operand

++ m; or **m ++;**

-- m; or **m --;**

Prefix and Postfix Operators

- Prefix operator: **y = ++m;** or **y = --m;**
  - Adds/subtracts **1** to the operand **m**
  - Result is assigned to the variable **y** on left

- Postfix operator: **y = m++;** or **y = m--;**
  - Assigns the value to the variable **y** on left
  - Increments/decrements the operand **m**

Exercise two

- What will be the final values of following variables. Expressions are executed individually.

  int i = 3, j = 4, k = 5, l=0, m=0 ;

  - m = i ++ ;
  - l = j -- ;
  - m = ++ k % -- j ;
  - l = j ++ * -- i ;
  - m = ++ j + i ;

Conditional Operators

- The operator **? :**
- Use to construct conditional expressions

exp1 ? exp2 : exp3

If exp1 is true;

conditional expression = exp2

If exp1 is false;

conditional expression = exp3

Bitwise Operators

- Use for testing bits, or shifting them to left or right

~	Compliment
&	AND
|	OR
^	XOR (exclusive OR)
<<	Shift left
>>	Shift Right
>>>	Shift Right with zero fill

Special Operators

class instantiation	:	**new**
class test operator	:	**instanceof()**
class member access	:	**.**
method invocation	:	**( )**
string concatenation	:	**+**
array element access	:	**[ ]**

Arithmetic Operator Precedence

- High priority *** / %**
- Low priority **+ -**
- Parenthesis contents are evaluated first!!
  - Left-to-right passes
  - Innermost to outer
- Expressions are evaluated from;

  left → right

Operator Precedence

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
= *= /= %= += -=	?:	||	&&	|	^	&	== !=	< <= > >=	<< >> >> >	+ -	* / %	new (type)	++ - - - ~ !	. [ ] ()

Priority increases...

Example of Operator Precedence

Example:

74 / 10 % 2 * 5 – 10 % (5 – 1)

- First deal with ( )
- Next work from left to right on / , %and operators
- Finally perform the subtraction

Exercise three

Evaluate the following expressions, and write the final answer.

- 1+2/3*4+5;
- 2/(3/3);
- 4/3*2/5;

Class java.lang.Math

- This class has methods for trigonometric and other useful functions.
  - Math.sqrt()
  - Math.max()
  - Math.min()
  - Math.abs()
  - Math.ceil()
  - Math.floor()
  - Math.random()

How to use: Math.random()

- Math.random() return a double value
  >=0.0d and **<1.0d**

- Eg: If you want to produce a random number between 0 to 10…
  int i = (int) (Math.random() * 10);

How to Write Methods?

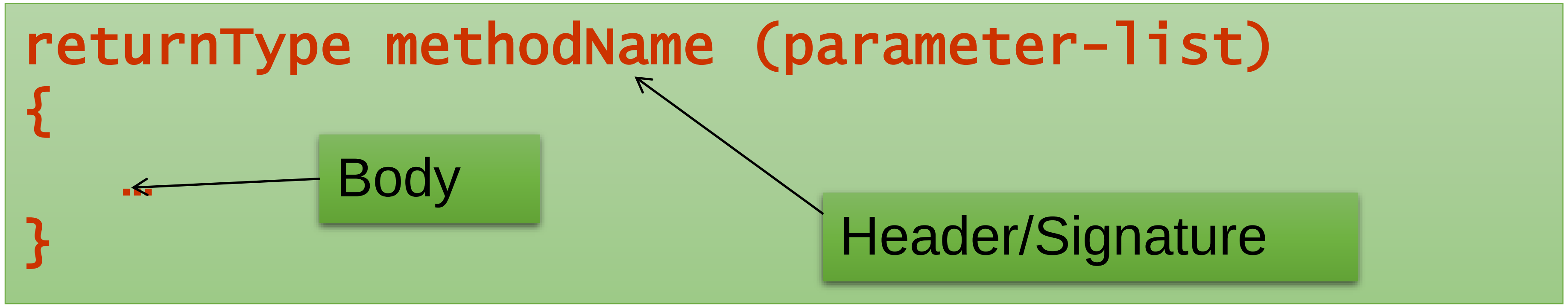
Methods

- The purpose of using methods is to break up a program into smaller, reusable pieces of software.
- While some methods are predefined - that is written and included as part of the Java environment, most methods will be written by the programmer.

How to write a Method?

- We have so far used methods such as main() and will now look at how we can create methods of our own.
- To define a method:
  - give it a name
  - specify the method's return type or choose void
  - specify the types of parameters and give them names or keep the parenthesis empty.
  - write the method's code

How to write a Method?



```
returnType methodName (parameter-list)
{
    ..
}
```

- A method is always defined inside a class.
- A method returns a value of the specified type unless it is declared void; the return type can be any primitive data type or a class type.
- A method’s parameters can be of any primitive data types or class types.

Exercise Four

- Write a method to display
  - Your favorite movie
  - Your favorite movie category
  - Your favorite actor/actress

Exercise Five

- Now modify your method to display,
  - Favorite movie
  - Favorite movie category
  - Favorite actor/actress

taken as parameters.

Invoking a Method

- We invoke (or ‘call’) a method by stating:
  - Its name (identifier)
  - The values to be taken by its parameters

- Example:

```
displayMovieDetails();
displayMovieDetails(“Kung Fu Panda”, “Romantic Comedy”,
“Selena Gomez”);
```

Passing Parameters

- How does the following really work?

```
displayMovieDetails(“Kung Fu Panda”, “Romantic
  Comedy”, “Selena Gomez”);
```

- The key point is that the method only ever receives a copy of the parameters given in the call.

Passing Parameters

- So the values that are supplied to the method as parameters can be:
  - *constant* values, such as `"Kung Fu Panda"`
  - *expressions*, such as `"Kung Fu"+"Panda"`
  - *variables*, such as in `movie="Kung Fu Panda"`
- Where an expression is used, it is evaluated first and then the result is copied to the method.
- Where a variable is used, its value is copied to the method and the variable remains unchanged.

Formal & Actual Parameters

- The formal parameters are:
  - The identifiers used when writing the method signature.
  - Their use is local to the method
- The actual parameters are:
  - the parameters in the method call (those being passed to the method).
- Actual parameters must match the formal parameters in number and type.

Local Variables

- Local variables are the variables that we declare within a method.
- These have a temporary existence and their values are discarded when the method returns control to the caller.
- So they can only be accessed within that method.

Returning Information

- The rules of Java only allow us to pass information into a method through the parameters.

- To get results out of a method, we turn it into an expression and return a value of a particular type.

Returning Information

- In exercise 1 and 2 both, the methods were of type void which means that they do not return any value.
- When calling void methods there is no need to be assigned to a variable.

Returning Information

- But when we write methods to return a value;
  - In the method we give it a **type** (such as **`int,float,etc`**) in place of **`void`**
  - At the end of the method body we give a **`return`** statement to return a value of the **selected type**.

- When calling above methods it needs to be assigned to a variable.

Exercise Six

- Write a method called **calcTotal** to add two numbers that are given as parameters and return the total.
- Invoke **calcTotal()** inside the main method.

```
public class Example {
    public static void main(String[] args) {
        int num1 = 5;
        int num2 = 7;
        int total = calcTotal(num1, num2);
        System.out.println("The total is: " + total);
    }

    public static int calcTotal(int num1, int num2) {
        int total = num1 + num2;
        return total;
    }
}
```

Summery

- Operator is a symbol that tells the computer to perform a mathematical or logical manipulation.
- Arithmetic operators,Relational operators,Logical operators,Assignment operators,Increment and decrement operators,Conditional operators,Bitwise operators and Special operators are used in JAVA.
- java.lang.Math class has methods for trigonometric and other useful functions.
- A method is always defined inside a class and returns a value of the specified type unless it is declared void; the return type can be any primitive data type or a class type

Thank you