CORE JAVA 8 - DAY 2

EXPRESSIONS

- An expression is a combination of constants (like 100), operators (like +), variables(section of memory) and parentheses (like “(” and “)”) used to calculate a value.
- x = 1; y = 100 + x;
- This is the stuff that you know from algebra, like: (32 - y) / (x + 5)
- *There are rules for this, but best rule is that an expression must look OK as algebra.*
- Based on what you know about algebra, what is the value of *12 – 4/2 + 2 ?*
- *Spaces don’t much matter.*

OPERATORS

- Unary,Binary and Ternary Operators
(
Arithmetic,Relational,Bitwise,Logical)
- Arithmetic Calculations
- Operator Precedence

OPERATORS

- Operators are special symbols that perform specific operations on one, two, or three *operands*, and then return a result.
- There are several types of Operators in Java:
  - **The Simple Assignment Operator**
  - **The Arithmetic Operators**
  - **The Unary Operators**
  - **The Equality and Relational Operators**
  - **The Conditional Operators**
  - **The Type Comparison Operator instanceof**

- Most common operators that you'll use is the simple assignment operator "=“,
- it assigns the value on its right to the operand on its left:

- int marks = 56;
- int total = 100;
- float percent = 56.0;

THE ARITHMETIC OPERATORS

Following arithmetic oprators are provided by Java:

+ additive operator (also used for String concatenation)

- subtraction operator

* multiplication operator

/ division operator – returns a quotient

% remainder/modulus operator – returns a reminder

EXAMPLE

```
class ArithmeticExample {

public static void main (String[] args){
        int number = 1 + 8;
      System.out.println(number);
       number = number - 1;
      System.out.println(number);

       number = number * 2;
      System.out.println(number);

       number = number / 2;
      System.out.println(number);

       number = number + 8;
       number = number % 7;
      System.out.println(number);
                }
}
```

THE UNARY OPERATORS

- The unary operators require only one operand; they perform various operations such as incrementing/decrementing a value by one, negating an expression, or inverting the value of a Boolean.
- + Unary plus operator; indicates positive value (numbers are positive without this, however)
- - Unary minus operator; negates an expression
- ++ Increment operator; increments a value by 1
- -- Decrement operator; decrements a value by 1
- ! Logical complement operator; inverts the value of a boolean

THE EQUALITY AND RELATIONAL OPERATORS

- The equality and relational operators determine if one operand is greater than, less than, equal to, or not equal to another operand.
- Also known as Comparison Operators
  - int number1 = 1;
  - int number2 = 2;
  - System.out.println(number1 == number2);
  - System.out.println(number1 != number2);
  - System.out.println(number1 > number2);
  - System.out.println(number1 < number2);
  - System.out.println(number1 <= number2);

Operator	
==	Equal to
!=	Not equal to
>	Greater
>=	Greater than or equal to
<	Less than
<=	Less than or equal to

THE CONDITIONAL OPERATORS

- The && and || operators perform Conditional-AND and Conditional-OR operations on two boolean expressions. These operators exhibit "short-circuiting" behavior, which means that the second operand is evaluated only if needed.
  - int number1 = 1;
  - int number2 = 2;
  - System.out.println((number1 == 1) && (number2 == 2));
  - System.out.println((number1 == 1) || (number2 == 1));

TERNARY OPERATOR '?:'

- Another conditional operator is ? : , which can be thought of as shorthand for an if-then-else statement
- It is called ternary because it uses three operands.
  - int number1 = 1;
  - int number2 = 2;
  - String s = number1>1 ? "Greater than one" : "Less than one";
  - System.out.println(s);

OPERATOR PRECEDENCE

- Operators can be combined into complex expressions

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

- Operators have a well-defined precedence which determines the order in which they are evaluated
- DMAS(Division Multiplication Addition Subtraction) rule is applied in Java Operator Precedence
- Multiplication, division, and remainder are evaluated prior to addition, subtraction, and string concatenation
- Arithmetic operators with the same precedence are evaluated from left to right, but parentheses can be used to force the evaluation order

OPERATOR PRECEDENCE

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

```
a + b + c + d + e      a + b * c - d / e
  1   2   3   4          3   2   4   1
```

```
a / (b + c) - d % e
  2     1     4   3
```

```
a / (b * (c + (d - e)))
  4     3     2     1
```

```
int a=3,b=3,c=3,d=3,e=3;
int f = a + b * c - d / e;

System.out.println(f);
```

OUTPUT????

OPERATOR PRECEDENCE

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

ARITHMETIC OPERATORS

- What is the value of -12 + 3
- An **arithmetic operator** is a symbol that asks for doing some arithmetic.

Operator	Meaning	Precedence
-	Unary minus	highest
+	Unary Plus	highest
*	Multiplication	middle
/	Division	middle
%	Modulus	middle
+	addition	low
-	Subtraction	low

BASICOPERATION.JAVA

```
class basicOperation {
//arithmetic using variables
        public static void main() {
                int a = 1+ 1;
                int b = 3 * 3;
                int c =  1 + 8/ 4;
                int d = -2;
                System.out.println( "a = " + a);
                System.out.println("b = " + b);
                System.out.println("c = " + c);
                System.out.println("d = " + d);
        }

}
```

BASICMATH.JAVA

```
class basicMath {
//arithmetic using variables
        public static void main() {
                int a = 1+ 3;
                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);
        }

}
```

PARENTHESES

- Difference between `-1 * ( 9 - 2 ) * 3` and `-1 * 9 - 2 * 3`
- To say **exactly** what numbers go with each operator, use parentheses.
- Nested Parentheses: The expression is evaluated starting at the most deeply nested set of parentheses (the "innermost set"), and then working outward until there are no parentheses left. If there are several sets of parentheses at the same level, they are evaluated left to right.
- `( ( ( 32 - 16 ) / ( 2 * 2 ) ) - ( 4 - 8 ) ) + 7`
- Are arithmetic expressions the only kind of expression?

INTERGERDIVISION.JAVA

```
class integerDivision {
  public static void main ( String[] args ) {
        System.out.println("The result is: " +
  (1/2 + 1/2) );
     }
}
```

- What is the value of 99/100 ?
- Change print statement to "print" (1.0/2 + 1/ 2 .0)

FINAL RESERVE WORD

```
class calculateTax {
   public static void main ( String[] arg ) {
   l final char c = 'A'; f
     final  int count = 0;
   . . . . . .
   }
}
```

ASSIGNMENT NO.1

- Write a program that averages the synsets created for three months April, May and June. Declare and initialize variable to the synset entered for each month. Compute the average and write out the results, something like this:

  Synsets Entered for April: 12
  Synsets Entered for May : 14
  Synsets Entered for June: 8
  Average Synset Entered: 11.333333

Check that your program prints the correct results.

ASSIGNMENT NO. 2

- Write a program that computes summation of 1 to n numbers where n can be any positive number. Declare and initialize variables and print the output as

  Value of n is : 5

  Sum of first 5 positive numbers : 15

Check the correctness of the result.