|  |
| --- |
| **Set 3: Numeric Operations** |

|  |
| --- |
| **Skill 3.1: Declare and/or initialize variables on a single line**  **Skill 3.2: Apply the fundamental arithmetic operations to int variable types**  **Skill 3.3: Apply unary operators**  **Skill 3.4: Apply compound operators**  **Skill 3.5: Apply PEMDAS to arithmetic operations** |

|  |
| --- |
| **Skill 3.1: Declare and/or initialize variables on a single line** |

**Skill 3.1 Concepts**

Recall that the assignment operator is the standard equal sign (=) and is used to “assign” a value to a variable.

Consider the following example,

|  |
| --- |
| **Code** |
| int i = 3; //OK. Assigns the value of 3 to i. Notice the direction of data flow.  3 = i; //Illegal. Data does not flow this way.  double p;  double j = 47.12;  p = j; //OK. Assigns the value of j to p. Both p and j are now equal to 47.12 |

It is possible to declare several variables of the same type on one line. In the below example, i, x, y, and z are all declared as int type variables.,

|  |
| --- |
| **Code** |
| int i, x, y, z; |

We can also declare and initialize variables on one line. In the example below, w and j, are both declared and initialized. g however is just declared.

|  |
| --- |
| **Code** |
| int w = 1000, j = 2000, g; |

**[Skill 3.1 Exercise 1](https://hpluska.github.io/APCompSciPrinciples/ticketOutTheDoor/set35/Set35TicketOutTheDoorAPCompSciPrinciples.pdf)**

|  |
| --- |
| **Skill 3.2: Apply the fundamental arithmetic operations to int variable types** |

**Skill 3.2 Concepts**

The basic arithmetic operations are as follows,

A screenshot of a calculator

AI-generated content may be incorrect.

Examples of how each of the above operators can be applied are illustrated below,

|  |  |
| --- | --- |
| **Addition** | **Output** |
| int x = 1;  int y = 2;  int z = x + y;//3 is assigned to z  x = x + z;//4 is re-assigned to x  System.out.println(z);  System.out.println(x + y);  System.out.println(x + 10);  System.out.println(10 + 10); | 3  6  14  20 |

|  |  |
| --- | --- |
| **Subtraction** | **Output** |
| int x = 1;  int y = 2;  int z = x - y;//-1 is assigned to z  x = x - z;//0 is re-assigned to x  System.out.println(z);  System.out.println(x - y);  System.out.println(x - 10);  System.out.println(10 - 10); | -1  -2  -10  0 |

|  |  |
| --- | --- |
| **Multiplication** | **Output** |
| int x = 1;  int y = 2;  int z = x \* y;//2 is assigned to z  x = x \* z;//2 is re-assigned to x  System.out.println(z);  System.out.println(x \* y);  System.out.println(x \* 10);  System.out.println(10 \* 10); | 2  4  20  100 |

The below example illustrates how int type variables are treated. If the result of the division is a fraction, the decimal places are "cut off" - *int variable types do not round*.

|  |  |
| --- | --- |
| **Division** | **Output** |
| int x = 1;  int y = 2;  int z = y/x;//2 is assigned to z  x = z/x;//2 is re-assigned to x  System.out.println(z);  System.out.println(x/y);  System.out.println(x/10);  System.out.println(10/10); | 2  1  0  1 |

Modulus prints the remainder of a division operation. For example, System.out.println(5%3); will print 2. This is because when 5 is divided by 3, the remainder is 2. Modulus gives the remainder. Modulus also handles negatives. The answer to a%b has the same sign as a. The sign of b is ignored.

|  |  |
| --- | --- |
| **Modulus** | **Output** |
| int x = 1;  int y = 2;  int z = x%y;//1 is assigned to z  x = x%z;//0 is re-assigned to x  System.out.println(z);  System.out.println(x%y);  System.out.println(x%10); | 1  0  0 |

**[Skill 3.2 Exercise 1](https://hpluska.github.io/APCompSciPrinciples/ticketOutTheDoor/set35/Set35TicketOutTheDoorAPCompSciPrinciples.pdf)**

|  |
| --- |
| **Skill 3.3: Apply unary operators** |

**Skill 3.3 Concepts**

The unary operators are operations that 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.

Incrementing a value by 1

The code below illustrates how to increment the variable x by 1

|  |
| --- |
| **Code** |
| int x = 1;  x = x + 1;//x is now 2 |

Incrementing a value by 1 can also be done using the *++* operator,

|  |
| --- |
| **Code** |
| int x = 1;  System.out.println(x++);//1 is printed to the consol, then x is incremented  System.out.println(++x);//x is incremented first, then it's value, 3, is printed to the consol. |
| **Output** |
| 1  3 |

Notice in the above example that *++* can come before or after the variable. If it comes *before* the variable, the variable is first incremented then printed. If it comes *after* the variable, the variable is first printed then incremented.

Decrementing a value by 1

The code below illustrates how to decrement the variable y by 1

|  |
| --- |
| **Code** |
| int y = 10;  y = y - 1;//y is now 9 |

Decrementing a value by 1 can also be done using the *--* operator,

|  |
| --- |
| **Code** |
| int y = 10;  System.out.println(y--);//10 is printed to the consol, then y is decremented  System.out.println(--y);//y is decremented first, then it's value, 8, is printed to the consol |
| **Output** |
| 10  8 |

Notice in the above example that *--* can come before or after the variable. If it comes *before* the variable, the variable is first decremented then printed. If it comes *after* the variable, the variable is first printed then decremented.

**[Skill 3.3 Exercise 1](https://hpluska.github.io/APCompSciPrinciples/ticketOutTheDoor/set35/Set35TicketOutTheDoorAPCompSciPrinciples.pdf)**

|  |
| --- |
| **Skill 3.4: Apply compound operators** |

**Skill 3.4 Concepts**

A compound assignment operator is an operator that performs a calculation and an assignment at the same time. In the below example, x can be re-assigned explicitly using *x = x + 5*; x can also be re-assigned using the addition compound operator.

|  |
| --- |
| **Code** |
| int x = 10;  x = x + 5;//x is 15  x += 5;//x is now 20 |

Compound operators can be applied to all the arithmetic operations. How this is done is illustrated below,

A math equations and formulas

AI-generated content may be incorrect.

**[Skill 3.4 Exercises 1](https://hpluska.github.io/APCompSciPrinciples/ticketOutTheDoor/set35/Set35TicketOutTheDoorAPCompSciPrinciples.pdf)**

|  |
| --- |
| **Skill 3.5: Apply PEMDAS to arithmetic operations** |

**Skill 3.5 Concepts**

The algebra rule, PEMDAS, applies to computer computations as well. (PEMDAS stands for the order in numeric operations are done. P = parenthesis, E = exponents, M = multiply, D = divide, A = add, S = subtract. Actually M and D have equal precedence, as do A and S. For equal precedence operation, proceed from left to right. A mnemonic for PEMDAS is, “Please Excuse My Dear Aunt Sally)

The example below illustrates PEMDAS

|  |
| --- |
| **Code** |
| System.out.println(5 + 3 \* 4 - 7);  System.out.println(8 - 5 \* 6 / 3 + (5 - 6) \* 3); |
| **Output** |
| **10**  **-5** |

**[Skills 3.5 Exercise 1](https://hpluska.github.io/APCompSciPrinciples/ticketOutTheDoor/set35/Set35TicketOutTheDoorAPCompSciPrinciples.pdf)**