BASIC 8

WEEKLY LESSON PLAN – WEEK 5

Strand:	Computational Thinking		**Sub-Strand:**	Introduction to programming	
Content Standard:	B8.4.1.1. Show an understanding of the concept of programming.				
Indicator (s)	B8.4.1.1.1 Describe the basic concepts in programming (Constants, Variables, Expressions, Statements /Instructions, logical and arithmetic operators, Operator precedence, etc.)			1. **Performance Indicator:** Learners can apply the concept of Programming.	
Week Ending	28-07-2023				
Class	B.S.8	**Class Size:**		**Duration:**	
Subject	Computing				
Reference	Computing Curriculum, BS7 Computing Textbook, Teachers Resource Pack, Learners Resource Pack				
Teaching / Learning Resources	Chart, Poster, Video.		**Core Competencies:**	• Communication and Collaboration • Digital Literacy	

DAYS	**PHASE 1 : STARTER**	**PHASE 2: MAIN**	**PHASE 3: REFLECTION**
THURSDAY	Discuss the meaning of Coding with the Learners.	1. Using a Presentation, explain how coding relates to Mathematics. 2. Briefly explain the type of Mathematics used in Coding. 3. Create a table to compare how the same arithmetic notations are represented in coding and in mathematics. 4. Discuss with Learners the meanings of the following terminologies as used in Programming; i. Constants ii. Variables iii. Expressions iv. Statements or instructions v. Logical and arithmetic operator. vi. Operators vii. Operator precedence. Math and coding are deeply related, and when teaching your students how to code, you are -at the same time- delivering mathematical content, and a way of thinking that they can use, later on, while calculating something specific in their math class	Through questioning strategy, Reflect on the topic treated. **Exercise;** 1.What is Coding? 2. Create a table to compare how the arithmetic notation are represented.

math is used in coding;

Binary mathematics is the heart of the computer and an essential math field for computer programming. For all mathematical concepts, the binary number system uses only two digits, 0 and 1. It simplifies the coding process and is essential for low-level instructions used in hardware programming.

Arithmetic Operators
The Java programming language supports various arithmetic operators for all floating-point and integer numbers. These operators
are + (addition), - (subtraction), * (multiplication), / (division), and % (modulo). The following table summarizes the binary arithmetic operations in the Java programming language.

Binary Arithmetic Operators

Operator	**Use**	**Description**
+	op1 + op2	Adds op1 and op2; also used to concatenate strings
-	op1 - op2	Subtracts op2 from op1
*	op1 * op2	Multiplies op1 by op2
/	op1 / op2	Divides op1 by op2
%	op1 % op2	Computes the remainder of dividing op1 by op2

Here's an example program, Arithmetic Demo, that defines two integers and two double-precision floating-point numbers and uses the five arithmetic operators to perform different arithmetic operations. This program also uses + to concatenate strings. The arithmetic operations are shown in boldface:

```
public class Arithmetic Demo {
   public static void main(String[] args) {

      //a few numbers
      int i = 37;
      int j = 42;
      double x = 27.475;
      double y = 7.22;
```

```
        System.out.println("Variable values...");
        System.out.println("    i = " + i);
        System.out.println("    j = " + j);
        System.out.println("    x = " + x);
        System.out.println("    y = " + y);

        //adding numbers
        System.out.println("Adding...");
        System.out.println("    i + j = " + (i + j));
        System.out.println("    x + y = " + (x + y));

        //subtracting numbers
        System.out.println("Subtracting...");
        System.out.println("    i - j = " + (i - j));
        System.out.println("    x - y = " + (x - y));

        //multiplying numbers
        System.out.println("Multiplying...");
        System.out.println("    i * j = " + (i * j));
        System.out.println("    x * y = " + (x * y));

        //dividing numbers
        System.out.println("Dividing...");
        System.out.println("    i / j = " + (i / j));
        System.out.println("    x / y = " + (x / y));

        //computing the remainder resulting from dividing
numbers
        System.out.println("Computing the remainder...");
        System.out.println("    i % j = " + (i % j));
        System.out.println("    x % y = " + (x % y));

        //mixing types
        System.out.println("Mixing types...");
        System.out.println("    j + y = " + (j + y));
        System.out.println("    i * x = " + (i * x));
    }
}
```
The output from this program is:

```
Variable values...
    i = 37
    j = 42
    x = 27.475
    y = 7.22
Adding...
    i + j = 79
    x + y = 34.695
Subtracting...
    i - j = -5
    x - y = 20.255
Multiplying...
```

```
    i * j = 1554
    x * y = 198.37
Dividing...
    i / j = 0
    x / y = 3.8054
Computing the remainder...
    i % j = 37
    x % y = 5.815
Mixing types...
    j + y = 49.22
    i * x = 1016.58
```

Note that when an integer and a floating-point number are used as operands to a single arithmetic operation, the result is floating point. The integer is implicitly converted to a floating-point number before the operation takes place. The following table summarizes the data type returned by the arithmetic operators, based on the data type of the operands. The necessary conversions take place before the operation is performed.

Result Types of Arithmetic Operations

Data Type of Result	**Data Type of Operands**
long	Neither operand is a float or a double (integer arithmetic); at least one operand is a long.
int	Neither operand is a float or a double (integer arithmetic); neither operand is a long.
double	At least one operand is a double.
float	At least one operand is a float; neither operand is a double.

In addition to the binary forms of + and -, each of these operators has unary versions that perform the following operations, as shown in the next table:

Unary Arithmetic Operators

Operator	**Use**	**Description**
+	+op	Promotes op to int if it's a byte, short, or char

-	-op	Arithmetically negates op

Two shortcut arithmetic operators are ++, which increments its operand by 1, and --, which decrements its operand by 1. Either ++ or -- can appear before (*prefix*) or after (*postfix*) its operand. The prefix version, ++op/--op, evaluates to the value of the operand *after* the increment/decrement operation. The postfix version, op++/op--, evaluates to the value of the operand *before* the increment/decrement operation.

The following program, called SortDemo , uses ++ twice and -- once.

```
public class SortDemo {
    public static void main(String[] args) {
        int[] arrayOfInts = { 32, 87, 3, 589, 12, 1076,
                              2000, 8, 622, 127 };

        for (int i = arrayOfInts.length; --i >= 0; ) {
            for (int j = 0; j < i; j++) {
                if (arrayOfInts[j] > arrayOfInts[j+1]) {
                    int temp = arrayOfInts[j];
                    arrayOfInts[j] = arrayOfInts[j+1];
                    arrayOfInts[j+1] = temp;
                }
            }
        }

        for (int i = 0; i < arrayOfInts.length; i++) {
            System.out.print(arrayOfInts[i] + " ");
        }
        System.out.println();
    }
}
```

This program puts ten integer values into an array — a fixed-length structure that can hold multiple values of the same type — then sorts them. The boldface line of code declares an array referred to by arrayOfInts, creates the array, and puts ten integer values into it. The program uses arrayOfInts.length to get the number of elements in the array. Individual elements are accessed with this notation: arrayOfInts[index], where index is an integer indicating the position of the element within the array. Note that indices begin at 0. You'll get more details and examples for arrays in the section Arrays.

The output from this program is a list of numbers sorted from lowest to highest:

		3 8 12 32 87 127 589 622 1076 2000
		Let's look at how the SortDemo program uses -- to help control the outer of its two nested sorting loops. Here's the statement that controls the outer loop:
		
		`for (int i = arrayOfInts.length; --i >= 0; ) {`
		`    ...`
		`}`	

FRIDAY	Assist Learners to identify the basic operations of Programming.	1. Learners brainstorm to describe 5 basic operations that Programming Languages can perform.
		2. Discuss with Learners about the principles of Programming Languages.
		3. Show Learners a YouTube video on operators and expressions.
		4. Assist Learners to explain in detail the function of each type of operator.
		Example of Arithmetic Operators in C++
		
		Here is a code in C++ which illustrates all the basic arithmetic operators:
		
		`#include <iostream>`
		`using namespace std;`
		`int main()`
		`{`
		`cout<<"Welcome to DataFlair tutorials!\n\n"<<endl<<endl;`
		`int a = 10, b = 7;`
		`cout<<"The Addition of "<< a << " and " << b << " are: " << a + b <<endl;`
		`cout<<"The Subtraction of "<< a << " and " << b << " are: " << a - b <<endl;`
		`cout<<"The Multiplication of "<< a << " and " << b << " are: " << a * b <<endl;`
		`cout<<"The Division of "<< a << " and " << b << " are: " << a / b <<endl;`
		`cout<<"The Modulus operation between "<< a << " and " << b << " is: " << a % b <<endl;`
		`cout<<"The Incremented value ++a is: "<< ++a <<endl;`
		`cout<<"The Decremented value --a is: "<< --a <<endl;`
		`return 0;`
		`}`
		
		Example of Relational Operators in C-
		
		`#include <stdio.h>`
			Through questions and answers, conclude the lesson.
			
			Exercise;
			1. State 5 basic operations that Programming Languages can perform.
			2. Write an example of arithmetic operators in C++.

```
int main()
{
int a=10, b=10, c=20;
printf("Welcome to DataFlair tutorials!\n\n");
printf("For %d == %d : The output is: %d \n", a, b, a == b); //
condition is true
printf("For %d == %d : The output is: %d \n", a, c, a == c); //
condition is false
printf("For %d != %d : The output is: %d \n", a, c, a != c); //
condition is true
printf("For %d != %d : The output is: %d \n", a, b, a != b); //
condition is false
printf("For %d > %d : The output is: %d \n", a, b, a > b); //
condition is false
printf("For %d > %d : The output is: %d \n", a, c, a > c); //
condition is false
printf("For %d < %d : The output is: %d \n", a, b, a < b); //
condition is false
printf("For %d < %d : The output is: %d \n", a, c, a < c); //
condition is true
printf("For %d >= %d : The output is: %d \n", a, b, a >= b); //
condition is true
printf("For %d >= %d : The output is: %d \n", a, c, a >= c); //
condition is false
printf("For %d <= %d : The output is: %d \n", a, b, a <= b); //
condition is true
printf("For %d <= %d : The output is: %d \n", a, c, a <= c); //
condition is true
return 0;
}
```

Example of Logical Operators in C Programming-

```
#include <stdio.h>
int main()
{
int a = 10, b = 10, c = 20, answer;
printf("Welcome to Data Flair tutorials!\n\n");
answer = (a == b) && (c > b);
printf("For (%d == %d) && (%d != %d), the output is: %d
\n",a,b,b,c,answer); //condition is true
answer = (a == b) && (c < b) && (c>0);
printf("For (%d == %d) && (%d <= %d), the output is: %d
\n",a,b,b,c,answer); //condition is false
answer = (a == b) || (b > c);
printf("For (%d == %d) || (%d < %d), the output is: %d
\n",a,b,c,b,answer); / /condition is true
answer = (a != b) || (a <= b) || (a>c);
printf("For (%d != %d) || (%d < %d), the output is: %d
\n",a,b,c,b,answer); //condition is true
answer = !(a == b);
printf("For !(%d == %d), the output is: %d \n",a,b,answer);
//condition is false
answer = !(a != b);
printf("For !(%d == %d), the output is: %d \n",a,b,answer);
//condition is true
return 0;
}
```

Name of Teacher: ***School:*** ***District:***