Java Variables and Data Types

1.What is a Statically Typed and Dynamically Typed programming language?
Statically typed programming languages, type checking occurs at compile time. At compile time, source code in a specific programming language is converted to a machine-readable format.
Dynamically typed languages, type checking takes place at runtime or execution time. This means that variables are checked against types only when the program is executing.

2.What is Variable in Java?
A variable is a container which holds the value while the Java program is executed. A variable is assigned with a data type.
Variable is a name of memory location. There are three types of variables in java: local, instance and static.

3.How to assign value to Variable?
Syntax is:type variableName = value;

4.What are Primitive Data Types in Java?
Primitive data types specify the size and type of variable values. They are the building blocks of data manipulation and cannot be further divided into simpler data types.
There are 8 types of Primitive data types in Java – Boolean, char, byte, int, short, long, float, and double.

5.What are the Identifiers in Java?
Identifiers in Java are symbolic names used for identification. They can be a class name, variable name, method name, package name, constant name, and more. However, In Java, There are some reserved words that can not be used as an identifier.

6.List of Operators in Java?
Operator in Java is a symbol that is used to perform operations.There are many types of operators in Java which are given below:

- Unary Operator,
- Arithmetic Operator,
- Shift Operator,
- Relational Operator,
- Bitwise Operator,
- Logical Operator,
- Ternary Operator and
- Assignment Operator.

7.Explain about Increment and Decrement operators and give an example.
Increment Operator (++): The increment (++) operator (also known as increment unary operator) in Java is used to increase the value of a variable by 1. Since it is a type of a unary operator, it can be used with a single operand.

The syntax for increment operator is a pair of addition signs ie;

++x;
x++;
The operator can be applied either before or after the variable. Both will have the same increment of 1. However, they both have separate uses and can be categorized as the following types.

- Pre-Increment Operator
- Post-Increment Operator

Pre-Increment Operator (++x;)

If the increment operator (++) is specified before the variable like a prefix (++x), then it is called pre-increment operator. In this case, the value of the variable is first incremented by 1, and then further computations are performed.

Example:

```
public class PreIncrementOperator {

        public static void main(String[] args) {

                int a = 5;

                System.out.println("Initial value of Variable = " + a);

                // using pre-increment operator

                int b = ++a;

                System.out.println("variable = " + a);

                System.out.println("preIncrement = " +bt);

                System.out.println("++preIncrement = " + ++b);

        }

}
```

Post-Increment Operator (x++;)

If the increment operator (++) is specified after the variable like a postfix (x++), then it is called post-increment operator. In this case, the original value of the variable (without increment) is used for computations and then it is incremented by 1.

Example:

```
public class PostIncrementOperator {

        public static void main(String[] args) {

                int a = 100;

                System.out.println("Initial Variable = " +a);
                // using post-increment operator
                int b = a++;
                System.out.println("postIncrement = " + b);
                System.out.println("variable = " +a + "\n");
        // postIncrement = 101
                System.out.println(“postIncrement ++ = " + b++);
        // postIncrement = 102
                System.out.println("postIncrement++ = " + b++);
        // postIncrement = 103
                System.out.println("postIncrement++ = " + b++);
                System.out.println("\n postIncrement = " + b);
        }
}
```