1. What is statically typed and dynamically typed programming language?

Statically Typed Programming Language: It is the memory of the variable which is given during compilation time itself. e.g.: C, C++, java

Dynamically Typed Programming Language: It is the memory of the variable which is given during execution time itself. e.g.: Python, PHP, JavaScript

2. What is the variable in Java?

- Variable is the title of a reserved region allocated in memory. In other words, it may be referred to as the name of a memory location.
- It is a container that holds the value while the java program is executed.
- Each variable should be given a unique name to indicate the storage area.
- A variable is assigned with a data type

3. How to assign a value to variable?

To assign a variable are the following steps:

Step1: To know which type of the value are you going to assign that variable

Step2: To write the suitable data type according to step1.

Step3: To write a suitable variable name according to naming conversation for variable in java.

Step4: (Optional) To assign the value what you had decided earlier before writing the suitable data type.

4. What are Primitive Data Type in Java?

Primitive data types are predefined by the language and is named by a reserved keyword.

1) Boolean Type: The Boolean data type can have two values- true or false and hence are typically used in true or false situations. For example, boolean flag = true;
2) Byte type: Values for the byte data type range from -128 to 127(8-bit signed two's component integer). A byte type is used in place of an int to save memory when it is certain that the value of a variable will be between -128 and 127. For example, byte range = 105;
3) Short type: The short data type can have values ranging from -32768 to 32767(16-bit signed two's component integer). If the value of a variable is certain to be between -32768 and 32767, short is used in place of other inter data types. For example, short loss = -50;
4) Int Type: Values for the int data type range from 2^{31} to 2^{31} -1(32-bit signed two's component integer). In java SE 8 and later, you can use the int data type to represent an unsigned 32-bit integer, which has a minimum value of 0 and a maximum value of 2^{32}-1. For example, int profit= 5000;
5) Long Type: Values for the long data type range from -263 to 263-1(63-bit signed two's component integer). You can use an unsigned 64-bit integer with a minimum value of 0 and a maximum value of 264-1 if you are using java 8 or later. For example: long profit = 8988989798769L;
6) Double Type: The double type is a 64-bit floating-point type with double precision. It should never be used for exact values like currency. For example: double height = 12.5;
7) Float Type: The float data type is a 32-bit single-precision floating-point value. It should never be used for precise values like money. For example: float depth = 32.3f;

8) Char Type: It is a Unicode (an international character encoding standard that provides a unique for every character across languages and scripts)16-bit characters. The char data type has a minimum value of ‘u0000’ (0) and a maximum value of ‘uffff’ For example: char temp = ‘a’;

5. What are the Identifiers in Java?

An identifier is a name given to a package, class, interface, method, or variable. All identifier must have different names.

In java, there a few points to remember while dealing with identifiers:

Rule1- All identifiers should begin with a later(A to Z or a to z), $ and _ and must be unique.

Rule2- after the first character/letter, identifiers can have any combination of characters.

Rule3- A keyword cannot be used as an identifier.

Rule4- The identifiers are case sensitive.

Rule5- Whitespaces are not permitter.

Examples of legal identifiers: rank, $name, _rate, __2_mark.

Example of illegal identifiers: 102pqr, -name.

6. List the operators in Java?

Operators in java can be classified into 6 types:

1. Arithmetic Operators:

- Addition (+)
- Subtraction (-)
- Multiplication (*)
- Division (/)
- Modulus (%)
- Increment (++)
- Decrement (--)

2. Relational Operators:

- Equal to (==)
- Not equal to (!=)
- Greater than (>)
- Less than (<)
- Greater than or equal to (>=)
- Less than or equal to (<=)

3. Logical Operators:

- Logical AND (&&)
- Logical OR (||)
- Logical NOT (!)

4. Assignment Operators:

- Assignment (=)
- Addition assignment (+=)
- Subtraction assignment (-=)
- Multiplication assignment (*=)
- Division assignment (/=)
- Modulus assignment (%=)

5. Unary Operators:

- Unary plus (+)
- Unary minus (-)
- Increment (++)
- Decrement (--)
- Logical NOT (!)
- Bitwise complement (~)

6. Bitwise Operators:

- Bitwise AND (&)
- Bitwise OR (|)
- Bitwise XOR (^)
- Bitwise complement (~)
- Left shift (<<)
- Right shift (>>)
- Unsigned right shift (>>>)

7. Explain about Increment and Decrement operators and give an examples

Increment Operators: In Java, increment operators are used to increase the value of a variable by 1. There are two types of increment operators: pre-increment and post-increment

- Pre-increment (++variable): In pre-increment, the value of the variable is incremented before the value is used in the expression. This means that the variable is incremented first, and then the expression is evaluated with the updated value.
- Post-increment (variable++): In post-increment, the value of the variable is used in the expression first, and then it is incremented. This means that the expression is evaluated with the current value of the variable, and then the variable is incremented.

Decrement Operators: Decrement operators in Java are used to decrease the value of a variable by 1. There are two types of decrement operators: pre-decrement and post-decrement.

- Pre-decrement (--variable): In pre-decrement, the value of the variable is decremented before the value is used in the expression. This means that the variable is decremented first, and then the expression is evaluated with the updated value.
- Post-decrement (variable--): In post-decrement, the value of the variable is used in the expression first, and then it is decremented. This means that the expression is evaluated with the current value of the variable, and then the variable is decremented.