1) what is statically typed and dynamically typed programming language?

Ans:
Statically typed and dynamically typed languages are two categories of programming languages. They define how to handle variables of different data types.

Statically typed languages check the types of variables at compile time, which means before the code is run. This means that the programmer has to specify the type of each variable before using it. Examples of statically typed languages are Java and C++.

Dynamically typed languages check the types of variables at run time, which means while the code is running. This means that the programmer doesn’t have to specify the type of each variable before using it. Some examples of dynamically typed languages are Python and JavaScript.

2) what is a variable in Java?

Ans:
A variable is a container that holds a value in a program. A variable is assigned a data type, which designates the type and quantity of value it can hold. A variable is a name given to a memory location. The value stored in a variable can be changed during program execution.

Ex: int age = 18; (here age is variable which stores integer value 18)
    Char name = ‘c’; (here name is variable which stores ‘c’)

3) how to assign value to a variable?

Ans:
We can assign a value to a variable using the = operator. The value being assigned must be of the same data type as the variable or be able to be implicitly converted to that data type. Here’s an example:

```
int myVariable; // declaring a variable of type int
myVariable = 5; // assigning the value 5 to the variable
```

We can also declare and initialize a variable in one line like this:

```
int myVariable = 5; // declaring and initializing a variable of type int with the value 5
```

4) what are the primitive data types in Java?

Ans:
In Java, there are eight primitive data types: byte, short, int, long, float, double, boolean, and char. These data types are predefined by the Java language and named by reserved keywords.

Each primitive data type has a specific size and range of values it can represent.
For example,

- byte: 8-bit signed two's complement integer with a minimum value of -128 and a maximum value of 127 (inclusive).
- short: 16-bit signed two's complement integer with a minimum value of -32,768 and a maximum value of 32,767 (inclusive).
- int: 32-bit signed two's complement integer with a minimum value of -2^31 and a maximum value of 2^31-1 (inclusive).
- long: 64-bit signed two's complement integer with a minimum value of -2^63 and a maximum value of 2^63-1 (inclusive).
- float: Single-precision 32-bit IEEE 754 floating point.
- double: Double-precision 64-bit IEEE 754 floating point.
- boolean: Represents one bit of information with only two possible values: true and false.

5)what are the identifiers in Java?

Ans:
An identifier is just a name that we give to things like classes, methods, variables, and packages so that we can refer to them later.
For example, if we create a class called MyClass, then MyClass is an identifier.

There are some rules we need to follow when creating identifiers in Java. For example, we can't use special characters like @ or spaces in our identifiers. We also can't start our identifiers with a number. And we can't use any of the reserved words in Java as an identifier. Reserved words are words that have a special meaning in Java, like int, while, and class.

6) Lists the operators in Java.

Ans:
There are several types of operators in Java:

Arithmetic Operators: Used to perform basic mathematical operations such as addition, subtraction, multiplication, and division. Examples are `+`, `-`, `*`, and `/`.

Unary Operators: Require only one operand. Used to increment/decrement a value by one, negate an expression, or invert the value of a boolean. Examples are`++`, `--`, `!`, and `~`.

Assignment Operators: Used to assign a value to a variable. The most common assignment operator is `=`, but there are also compound assignment operators such as `+=` and `-=` that combine an arithmetic operation with the assignment.

Relational Operators: Used to test the relationship between two operands. These operators return a boolean value of `true` or `false` depending on whether the relationship is true. Examples are`<`, `>`, `<=`, `>=`, `==`, and `!=`.

Logical Operators: Used to combine two or more relational expressions. These operators return a boolean value of `true` or `false`. Examples are`&&` (logical AND), `||` (logical OR), and `!` (logical NOT).

Bitwise Operators: Used to perform bit-level operations on integer values. Examples are`&` (bitwise AND), `|` (bitwise OR), and `^` (bitwise XOR).

Shift Operators: Used to shift the bits of an integer value to the left or right. Examples are`<<` (left shift) and `>>` (right shift).

Ternary Operator: A conditional operator that takes three operands. It is used as a shorthand for an if-else statement.

7) Explain increment and decrement operators and give an example?

Ans:
Increment and decrement operators are used in programming languages to increase or decrease the value of a variable by a certain amount.

The increment operator "++" is used to increase the value of a variable by 1. It can be applied to numeric variables such as integers or floating-point numbers. When the increment operator is placed after the variable (post-increment), the value is first used and then incremented. When the increment operator is placed before the variable (pre-increment), the value is incremented first and then used.

```
int x = 10;
int y = ++x;
// x is now 10 and y is also 10
```

In this example, the value of `x` is initially 10. After `y` is assigned the value of `x`, `x` is incremented by 1.

The decrement operator "--" works similarly to the increment operator, but it decreases the value of a variable by 1. It follows the same rules for post-decrement (value used and then decremented) and pre-decrement (value decremented and then used).

Here's an example of the decrement operator:

```
int x =50;
int y = x++;
// x is now 51 but y is still 50
```

In this example, the value of `a` is initially 50. Before `b` is assigned the value of `a`, `a` is decremented by 1.