Assignment – 5th (Variables & Data types)

1. What is Statically typed and Dynamically typed Programming Language?

Ans: - Statically typed: - Statically typed programming languages are those in which the type of a variable is known at compile time, and it remains unchanged during the program execution. For example, in Java, we need to declare the type of a variable before we use it, and the type can't be changed afterwards.

Dynamically typed: - Dynamically typed programming languages, on the other hand, determine the type of a variable at runtime, and the type can change during the program execution. For example, in Python, we don't need to declare the type of a variable, and the type of the same variable can change during the execution of the program.

2. What is the variable in Java?

Ans: - In simpler terms, a variable in Java is like a container that holds a value and has a specific data type, like a number or a string. You give the variable a name, and the data type tells the computer what kind of value it should hold. The value stored in a variable can be changed during the program execution. Before using a variable, you have to declare it with a data type and a name, and you can't change its data type after declaration

3. How to assign a value to variable?

Ans: - To assign a value to a variable in Java, you simply write the name of the variable, followed by the assignment operator (=), followed by the value you want to assign to it. Here's an example:

```
age = 30;
```

In this example, we're saying "take the variable ***`age`*** and give it the value ***`30`***." It's that simple!

4. What are Primitive Data type in Java?

Ans: - Java primitive data types are the most basic data types that are used to store single values in a program. There are 8 primitive data types in Java. These data types include:

byte, short, int, long, float, double, char, and boolean

They are not objects, and do not have methods or behaviours. They are simply values stored in memory.

5. What are the Identifiers in Java?

Ans: - In Java, an identifier is a name used to identify a variable, class, method, or other element in a program. The rules for creating valid identifiers in Java are:

- An identifier must start with a letter, a dollar sign ($), or an underscore (_).
- Subsequent characters can be letters, digits, dollar signs, or underscores.
- Java is case-sensitive, so MyVariable and myVariable are different identifiers.
- Identifiers cannot be a reserved keyword in Java, such as int, class, public, etc.
- Identifiers can be any length, but it is recommended to use meaningful, descriptive names that are not too long.

Examples of valid identifiers in Java include: **`firstName`, `_private`, `interestRate`, `count`, `and CUSTOMER_ID`**.

6. List the Operators in Java?

Ans: - In Java, operators are symbols that perform operations on values (operands) and produce a result. Operators are used in expressions to manipulate values and control the flow of a program. They can be used in combination with variables, constants, and other expressions to perform various operations. Some common types of operators in Java include:

1. Arithmetic operators: **+, -, *, /, %, ++, --**
2. Relational operators: **>, <, >=, <=, ==, !=**
3. Logical operators: **&&, ||, !**
4. Bitwise operators: **&, |, ^, ~, <<, >>, >>>**
5. Assignment operators: **=, +=, -=, *=, /=, %=, &=, |=, ^=, <<=, >>=, >>>=**
6. Ternary operator: **? :**

7. Explain about Increment and Decrement operators and give an example?

Ans: - In Java, the increment and decrement operators are used to increase or decrease the value of a variable by 1.

- **Increment Operator (++): -** The increment operator is represented by **`++`** and it increases the value of a variable by 1. It can be used as a prefix **(++x)** or postfix **(x++)** operator.

Example: -

```
int x = 10;
x++;
System.out.println(x);   // Output: 11
```

- **Decrement Operator (--): -** The decrement operator is represented by **`--`** and it decreases the value of a variable by 1. It can be used as a prefix **(--x)** or postfix **(x--)** operator.

Example: -

```
int x = 10;
x--;
System.out.println(x);   // Output: 9
```