**JAVA VARIABLES AND DATA TYPES**

**Assignment#3**

**COURSE- JAVA WITH DSA SYSTEM DESIGN**

****

**SUBMITTED BY:**

**SAMIM MONDAL**

**SUBMITTED TO:**

**HYDER ABBAS**

**Q.1 What is Statically typed and dynamically typed programming language?**

**Ans.** Statically typed and dynamically typed are two different approaches to type checking in programming languages.

In a statically typed language, the type of a value (such as a variable or a function argument) is known at compile time, and the type of the value must match the type specified in the program. Examples of statically typed languages include Java, C, and C++.

In a dynamically typed language, the type of a value is determined at runtime and can change during the execution of the program. Examples of dynamically typed languages include Python, Ruby, and JavaScript.

**Advantages of static typing include:**

Type errors can be detected early, at compile time, rather than at runtime.

The type information can be used by the compiler to generate more efficient code.

Advantages of dynamic typing include:

More flexible, as the type of a value can change at runtime.

Easier to write code quickly, as the programmer does not have to specify the type of every value.

Often results in less verbose code.

The choice between static and dynamic typing depends on the specific requirements of the project and the personal preferences of the programmer. Some languages, like Scala, provide features of both static and dynamic typing, giving developers the best of both worlds.

**Q.2 What is variable in JAVA?**

**Ans.** In Java, a variable is a named storage location for holding a value. The value stored in a variable can change during the execution of a program.

A variable in Java must be declared before it can be used. The declaration specifies the type of the variable (such as int, float, or String), its name, and an optional initial value. For example:

The type of a variable determines the kind of values it can store, and the operations that can be performed on its values. For example, an int variable can store an integer value, while a float variable can store a floating-point value.

In Java, variables can be either primitive or reference types. Primitive types (such as int, float, and Boolean) hold their values directly, while reference types (such as objects) hold a reference to the memory location where the actual value is stored.

It is important to choose meaningful names for variables, as this makes the code more readable and easier to understand. In Java, variable names follow the rules for Java identifiers, which include starting with a letter, dollar sign, or underscore, and consisting of letters, digits, dollar signs, or underscores.

**Q.3 How to assign a value to variable?**

**Ans.** In Java, you can assign a value to a variable using the assignment operator (=). For example:

You can also assign a value to a variable that has already been declared, by using the assignment operator (=) again:

It's important to make sure that the type of the value being assigned to a variable is compatible with the type of the variable. For example, you cannot assign a string value to an int variable, as the two types are not compatible. Java will generate a type mismatch error if you try to do this.

**Q.4 What are primitive Data types in JAVA?**

**Ans**. In Java, primitive data types are basic data types that are built into the language and represent single values, as opposed to objects. There are eight primitive data types in Java:

* **byte:** a 8-bit signed integer
* **short:** a 16-bit signed integer
* **int:** a 32-bit signed integer
* **long:** a 64-bit signed integer
* **float:** a single-precision 32-bit floating point number
* **double:** a double-precision 64-bit floating point number
* **char:** a single 16-bit Unicode character
* **Boolean:** a logical value that can be either **True** or **False**

Primitive data types are the simplest data types in Java and are used to represent basic values, such as numbers, characters, and logical values. For example, you can use an int variable to store an integer value, and a float variable to store a floating-point value.

When you use a primitive data type, you are storing the value directly in the variable, rather than a reference to an object that contains the value. This means that primitive data types are more efficient for basic values, but less flexible, as they cannot be extended or modified like objects.

It's important to choose the appropriate primitive data type for your needs, as each type has a different size and range of values. For example, if you need to store a large integer value, you should use a long rather than an int, as the latter has a smaller range of values.

**Q.5 What are the identifiers in JAVA?**

**Ans.** In Java, an identifier is a name used to identify a variable, method, class, interface, or label. Identifiers follow a set of rules and conventions in Java and must be unique within the scope in which they are declared.

The rules for naming identifiers in Java include:

They must start with a letter, dollar sign ($), or underscore (\_), and can be followed by any combination of letters, digits, dollar signs, or underscores.

They cannot be a reserved word in the Java language.

They are case-sensitive, meaning that **myVariable** and **myvariable** are treated as different identifiers.

It's a good practice to choose meaningful and descriptive names for identifiers, as this makes the code more readable and easier to maintain. Java has a naming convention for identifiers, known as CamelCase, where the first letter of each word in an identifier is capitalized. For example, **myVariable** and **studentRecord**.

**In Java, there are several different types of identifiers, including:**

1. **Variable names:** used to identify a variable and store its value.
2. **Method names:** used to identify a method and specify the action it performs.
3. **Class names:** used to identify a class and define its properties and methods.
4. **Interface names:** used to identify an interface and define its methods.
5. **Package names:** used to organize classes and interfaces into packages.

It's important to choose meaningful and descriptive names for identifiers, as this makes the code more readable and easier to maintain. By following the rules and conventions for naming identifiers in Java, you can make your code more organized and professional.

**Q.6 List the Operators in JAVA?**

**Ans.** In Java, operators are symbols that perform operations on values (operands) and produce a result. There are several types of operators in Java, including:

**Arithmetic operators: perform mathematical operations such as addition, subtraction, multiplication, and division.**

* +: addition operator
* -: subtraction operator
* \*: multiplication operator
* /: division operator
* %: modulus operator (returns the remainder after division)

**Relational operators: compare values and determine the relationship between them.**

* >: greater than operator
* <: less than operator
* >=: greater than or equal to operator
* <=: less than or equal to operator
* ==: equal to operator
* !=: not equal to operator

**Logical operators: perform logical operations on Boolean values and produce a Boolean result.**

* &&: logical AND operator
* ||: logical OR operator
* !: logical NOT operator

**Bitwise operators: perform operations on the binary representation of values.**

* &: bitwise AND operator
* |: bitwise OR operator
* ^: bitwise XOR operator
* ~: bitwise NOT operator
* <<: left shift operator
* >>: right shift operator
* >>>: right shift with zero fill operator

**Assignment operators: assign a value to a variable.**

* =: simple assignment operator
* +=: addition and assignment operator
* -=: subtraction and assignment operator
* \*=: multiplication and assignment operator
* /=: division and assignment operator
* %=: modulus and assignment operator

**Ternary operator: a shorthand for an if-else statement that returns a value based on a condition**

* ? :: ternary operator

**Other operators:**

* instanceof: determines if an object is an instance of a specific class or interface
* []: array access operator
* .: member access operator
* (): method invocation operator

Each operator has a specific priority, determining the order in which operations are performed. It's important to understand the priority of operators, as it can affect the result of an expression. You can use parentheses to control the order of operations and ensure the expression is evaluated as you intend.

**Q.7 Explain about increment and decrement operators and give an example?**

**Ans.** The increment and decrement operators in Java are used to increase or decrease the value of a variable by a certain amount.

**Increment operator (++):** This operator increments the value of a variable by 1. There are two forms of the increment operator: pre-increment (++x) and post-increment (x++). The pre-increment operator increments the value of the variable before it is used in the expression, while the post-increment operator increments the value of the variable after it is used in the expression.

**Example:**

JAVA

int x = 5;

int y = ++x; // x is 6, y is 6

int z = x++; // x is 7, z is 6

**Decrement operator (--):** This operator decrements the value of a variable by 1. Like the increment operator, there are two forms of the decrement operator: pre-decrement (--x) and post-decrement (x--). The pre-decrement operator decrements the value of the variable before it is used in the expression, while the post-decrement operator decrements the value of the variable after it is used in the expression.

**Example:**

JAVA

int a = 5;

int b = --a; // a is 4, b is 4

int c = a--; // a is 3, c is 4

It's important to note that the increment and decrement operators can also be used in compound assignments, such as x += 1 (equivalent to x = x + 1) and x -= 1 (equivalent to x = x - 1). These operators are often used in loops and other control structures to perform repetitive operations.

**THANK YOU!!**