**Assignment: (Core Java)**

**2 : Data Types, Variables, and Operators**

**Que.1 Primitive Data Types in Java (int, float, char, etc.)**

**Ans.1** Primitive data types are fundamental building blocks in programming that directly store simple values in memory. The image lists several common primitive data types along with their size in bits and bytes, and in some cases, their value ranges:

* Boolean: Stores True or False values (T/F).
* Char: Typically 16 bits or 2 bytes.
* Byte: 8 bits or 1 byte, with a value range from -128 to 127.
* Short: 16 bits or 2 bytes, with a value range from -32,768 to 32,767.
* Int: 32 bits or 4 bytes.
* Long: 64 bits or 8 bytes.
* Float: 32 bits or 4 bytes.
* Double: 64 bits or 8 bytes.

**Que.2 Variable Declaration and Initialization**

**Declaration** means telling the compiler that a variable exists and what type of data it will hold. It reserves memory space but doesn’t necessarily assign a value.

**Initialization** means assigning an initial value to a variable at the time of declaration.

Exa.

int age; // declaration

age = 20; // initialization

**Que.3 Operators: Arithmetic, Relational, Logical, Assignment, Unary, and Bitwise**

**Ans.3 1. Arithmetic Operators**

|  |  |  |
| --- | --- | --- |
| Operator | Meaning | Example |
| + | Addition | A + b |
| - | Subtraction | A - b |
| \* | Multiplication | A \* b |
| / | Division | A / b |
| % | Modulus (remainder) | A % b |

**2. Relational (Comparison) Operators :** Relational operators are used in programming to compare two values and determine the relationship between them, such as equality, greater than, or less than. They return a boolean value (true or false) based on the comparison.

|  |  |  |
| --- | --- | --- |
| Operator | Meaning | Example |
| == | Equal to | a == b |
| != | Not equal to | a != b |
| > | Greater than | a > b |
| < | Less than | a < b |
| >= | Greater than or equal to | a >= b |
| <= | Less than or equal to | a <= b |

**3. Logical Operators :** Logical operators are used to perform "logical AND" and "logical OR" operations.

They exhibit a short-circuiting effect, meaning the second condition is not evaluated if the first condition is false or true

**Logical AND (&& or &):** Returns true only when both conditions are true.

**Logical OR (|| or |):** Returns true if at least one condition is true.

**Logical NOT (!):** Returns true when the condition is false, and vice-versa

**4. Assignment Operators:** Used to assign values to variables.

|  |  |  |
| --- | --- | --- |
| Operator | Meaning | Example |
| = | Assign | a = 10 |
| += | Add and assign | a += 10 |
| -= | |  | | --- | |  |  |  | | --- | | Subtract and assign | | a -= 10 |
| \*= | Multiply and assign | a \*= 10 |
| /= | |  | | --- | |  |   Divide and assign | a /= 10 |
| %= | Modulus and assign | a% = 10 |

**5. Unary Operators:** Work on a single operand.

| Operator | Meaning | Example (a=5) |
| --- | --- | --- |
| ++a | Pre-increment (increase, then use) | ++a |
| a++ | Post-increment (use, then increase) | a++ |
| --a | Pre-decrement (decrease, then use) | --a |
| a-- | Post-decrement (use, then decrease) | a-- |
| +a | Unary plus (no effect) | +a |
| -a | Unary minus (negation) | -a |
| ! | Logical NOT | !true |

**6. Bitwise Operators:**

| Operator | Meaning | Example | Binary Result |
| --- | --- | --- | --- |
| & | AND | a & b | 0001 |
| ` | ` | OR | `a |
| ^ | XOR | a ^ b | 0110 |
| ~ | NOT (1’s complement) | ~a | 1010 |
| << | Left shift | a << 1 | 1010 |
| >> | Right shift | a >> 1 | 0010 |
| >>> | Unsigned right shift | a >>> 1 | 0010 |

**Que.4 Type Conversion and Type Casting**

Type conversion and type casting both involve changing an entity of one data type into another, but they differ in how the conversion is performed and the potential for data loss.

| **S.NO** | **TYPE CASTING** | **TYPE CONVERSION** |
| --- | --- | --- |
| 1. | In type casting, a data type is converted into another data type by a programmer using casting operator. | Whereas in type conversion, a data type is converted into another data type by a compiler. |
| 2. | Type casting can be applied to compatible data types as well as incompatible data types. | Whereas type conversion can only be applied to compatible datatypes. |
| 3. | In type casting, casting operator is needed in order to cast a data type to another data type. | Whereas in type conversion, there is no need for a casting operator. |
| 4. | In type casting, the destination data type may be smaller than the source data type, when converting the data type to another data type. | Whereas in type conversion, the destination data type can't be smaller than source data type. |
| 5. | Type casting takes place during the program design by programmer. | Whereas type conversion is done at the compile time. |
| 6. | Type casting is also called narrowing conversion because in this, the destination data type may be smaller than the source data type. | Whereas type conversion is also called widening conversion because in this, the destination data type can not be smaller than the source data type. |
| 7. | Type casting is often used in coding and competitive programming works. | Whereas type conversion is less used in coding and competitive programming as it might cause incorrect answer. |
| 8. | Type casting is more efficient and reliable. | Whereas type conversion is less efficient and less reliable. |