**Java Tokens:-**

**Java Tokens are the smallest individual building block or smallest unit of a Java program.**

**int a =b+c\*d;**

**Tokens are int, a, =, b,+, c, \*, d, ;**

**There is total 9 tokens.**

**Tokens:**

**1)Data Types: int**

**2)Identifiers: a, b, c, d**

**3)Operators: =, +, \***

**4)Special Symbol: ;**

**Types of tokens: 4.**

**Identifier:-**

**A name in java program is called identifier. It may be class name, method name, variable name and label name etc.**

**We use identifier for identification purpose of class, method, variable, label etc.**

**Example:**

**public class Test**

**{**

**public static void main(String[] args)**

**{**

**int x=20;**

**}**

**}**

**There are totals five identifier.**

**Rules to define java identifiers:-**

**Rule 1:-The only allowed characters in java identifiers are:**

**1) a to z**

**2) A to Z**

**3) 0 to 9**

**4) \_ (underscore)**

**5) $**

**Rule 2:- If we are using any other character we will get compile time error.**

**Example:-**

**1) max\_number-------valid**

**2) max#------------------invalid**

**Rule 3:-identifiers are not allowed to starts with digit.**

**Example:**

**1) ABC123---------valid**

**2) 123ABC---------invalid**

**Reserved Key:- In java some identifiers are reserved to associate some functionality or meaning, such type of reserved identifiers are called reserved words.**

**All reserved keyword divided into different section:-**

**Reserved words for data types: (8)**

**1) byte 2) short 3) int 4) long 5) float 6) double 7) char 8) boolean**

**Reserved words for flow control:(11)**

**1) if 2) else 3) switch 4) case 5) default 6) for 7) do 8) while**

**9) break 10) continue 11) return**

**Keywords for modifiers:(11)**

**1) public 2) private 3) protected 4) static 5) final 6) abstract**

**7) synchronized 8) native 9) strictfp(1.2 version) 10) transient**

**11) volatile**

**Keywords for exception handling:(6)**

**1) try 2) catch 3) finally 4) throw 5) throws 6) assert(1.4 version)**

**Class related keywords:(6)**

**1) class 2) package 3) import 4) extends 5) implements 6) interface**

**Object related keywords:(4)**

**1) new 2) instanceof 3) super 4) this**

**void return type keyword:**

**If a method won't return anything then that method must be declared with the void return type.**

**Unused keywords:**

**goto: Create several problems in old languages and hence it is banned in java.**

**const: Use final instead of this.**

**By mistake if we are using these keywords in our program, we will get compile time error.**

**Reserved literals:**

**1) true values for boolean data type.**

**2) false**

**3) null----------------- default value for object reference.**

**enum:**

**This keyword introduced in 1.5v to define a group of named constants**

**Data Types:-**

**Every variable, every expression has some type and all types are strictly defined.**

**In java all assignment must be checked by compiler for type compatibility hence java language is considered as strongly typed programming language.**

* **Primitive Data types.**

1. **Numeric Datatypes.**
2. **Integral Data types.**
3. **byte .**
4. **short.**
5. **int.**
6. **long.**
7. **Floating point Datatypes.**
8. **Float.**
9. **Double.**
10. **Character Datatypes.**
11. **Boolean Datatypes.**

**Integral data types:- By using this types data we represent integer value.**

**byte:- byte data type is best suitable if we are handling data in terms of streams either from the file or from the network.**

**Size: 1byte (8bits)**

**Range:-128to 127[-27 to 27-1]**

**Example:-**

**byte b=10;**

**byte b2=130;//C.E:possible loss of precision**

**byte b=10.5;//C.E:possible loss of precision**

**byte b=true;//C.E:incompatible types**

**byte b="abc";//C.E:incompatible types**

**Short:-**

**The most rarely used data type in java is short.**

**Short data type is best suitable for 16-bit processors like 8086 but these processors are completely outdated, so short data type is also out data type.**

**Size: 2 bytes**

**Range: -32768 to 32767(-215 to 215-1)**

**Example:-**

**short s=130;**

**short s=32768;//C.E:possible loss of precision**

**short s=true;//C.E:incompatible types**

**int:-This is most commonly used data type in java.**

**Size: 4 bytes**

**Range:-2147483648 to 2147483647 (-231 to 231-1)**

**Example:**

**int i=130;**

**int i=10.5;//C.E:possible loss of precision**

**int i=true;//C.E:incompatible types**

**long:-Whenever int is not enough to hold big values then we should go for long data type.**

**Suppose we require to count all character present in book we will use long data type.**

**Size: 8 bytes**

**Range:-263 to 263-1**

**Floating point Datatypes:-If we want to represent decimal or fraction value then we should go for Floating point Datatypes.**

|  |  |
| --- | --- |
| **Float** | **double** |
| **If we want to 5 to 6 decimal places of accuracy then we should go for float.** | **If we want to 14 to 15 decimal places of**  **accuracy then we should go for double.** |
| **Size:4 bytes.** | **Size:8 bytes.** |
| **Range:-3.4e38 to 3.4e38.**  **Where e=10.** | **Range:- -1.7e308 to 1.7e308.**  **Where e=10.** |
| **Suffix with f or F but not both.** | **Suffix with d or D but not both.** |

**Example:-**

**float sal1=10;//valid**

**float sal2=10.0f;//valid**

**float sal3=20.0F; //valid**

**double sal4=30.33d; //valid**

**double sal5=28.67D; //valid**

**float sal6=50.53;//invalid**

**boolean data type:-If we to represent true or false then we use boolean data type.**

**Size: Not applicable (virtual machine dependent)**

**Range: Not applicable but allowed values are true or false.**

**Example:-**

**boolean b=true;**

**boolean b=True;//C.E:cannot find symbol**

**boolean b="True";//C.E:incompatible types**

**boolean b=0;//C.E:incompatible types**

**char data type:-In java character data type is unicode based.**

**Size: 2 bytes**

**Range: 0 to 65535**

**Example:-**

**char ch1=97;**

**char ch2=65536;//C.E:possible loss of precision**

**Summary of java primitive data type:-**

|  |  |  |  |
| --- | --- | --- | --- |
| Data Type | Size | Range | Default Value |
| Byte | **1 byte** | **-128 to 127** | **0** |
| Short | **2 bytes** | **-32768 to 32767** | **0** |
| Int | **4 bytes** | **-2147483648**  **to 2147483647** | **0** |
| Long | **8 bytes** | **-263 to 263-1** | **0** |
| Float | **4 bytes** | **-3.4e38 to 3.4e38** | **0.0** |
| Double | **8 bytes** | **-1.7e308 to 1.7e308** | **0.0** |
| Boolean | **Not applicable** | **Not applicable but allowed values true|false** | **false** |
| Char | **2 bytes** | **0 to 65535** | **Single blank space** |

**Literals:-**

**Any constant value which can be assigned to the variable is called literal.**

**EX:**

**int a=10;**

**int ----> data types**

**a ------> variables/ identifier**

**= ------> Operator**

**10 -----> constant[Literal].**

**; ------> Special symbol.**

**Integral Literal:-**

**Example:-1**

**byte b1=127; //valid**

**byte b2=128;//invalid**

**short s1=32767; //valid**

**short s2=32768;//invalid**

**int i1=1900; //valid**

**long l1=100l; //valid**

**long l2=100L; //valid**

**char literals:-**

**A char literal can be represented as single character within single quotes.**

**Example:-**

**char ch='a';(valid)**

**char ch=a;//C.E:cannot find symbol(invalid)**

**char ch="a";//C.E:incompatible types(invalid)**

**char ch='ab';//C.E:unclosed character literal(invalid)**

**Floating Point Literals:-**

**Floating point literal is by default double type but we can specify explicitly as float type by suffixing with f or F.**

**Example:-**

**float f=123.456;//C.E:possible loss of precision(invalid)**

**float f=123.456f;(valid)**

**double d=123.456;(valid)**

**We can specify explicitly floating point literal as double type by suffixing with d or D.**

**Example:-**

**double d=123.456D;**

**double d=123.456d;**

**Boolean Literals:- The only allowed values for the boolean type are true (or) false.**

**Example:-**

**boolean b=true;(valid)**

**boolean b=0;//C.E:incompatible types(invalid)**

**boolean b=True;//C.E:cannot find symbol(invalid)**

**boolean b="true";//C.E:incompatible types(invalid)**

**String Literals:-**

**Any sequence of characters with in double quotes is treated as String literal.**

**Example:-**

**String s="india"; (valid)**

**Operator:-**

**Operator is a symbol; it will perform a particular operation over the provided operands.**

**int x=y+z;**

**where y and z is operand and + is one type of operator.**

**All operators divided into three parts:**

1. **Unary operator**
2. **Binary operator**
3. **Ternary operator**

**Unary operator:-the operator which act on single variable (operand) called unary operator**

**Unary operators are:-**

1. **Postfix operator**

* **Postfix increment operator X++**
* **Postfix decrement operator X--**

1. **Prefix operator**

* **Prefix increment operator ++X**
* **Prefix decrement operator --X**

**Postfix operator:-In this operator value is print first then value will be increment/decrement.**

**Postfix increment operator X++**

**Example:-**

**int x=10;**

**int y=x++;**

**Result:-**

**initial value of x=10**

**value of y=10**

**final value of x=11**

**Postfix decrement operator X--**

**Example:-**

**int x=10;**

**int y=x--;**

**Result:-**

**initial value of x=10**

**value of y=10**

**final value of x=9**

**Prefix operator:- In this operator value is increment/decrement first then values will print.**

**Prefix increment operator ++X**

**Example:-**

**int x=10;**

**int y=++x;**

**Result:-**

**initial value of x=10**

**value of x=11**

**final value of y=11**

**Prefix decrement operator --x**

**Example:-**

**int x=10;**

**int y=--x;**

**Result:-**

**initial value of x=10**

**value of y=9**

**final value of x=9**

**Case1:-Increment/Decrement operator we can apply only on variable not on constant.**

**Example:-**

**int x=5;**

**int y=x++;//valid**

**int z=10++;//invalid**

**Result:-**

**error: unexpected type int y=++10;**

**required: variable**

**found: value**

**Binary operator:-The operator which acts on two variables (operand) called Binary operator.**

**Binary operators are:-**

1. **Arithmetic operator (+,-,\*,/,%)**
2. **String Concatenation operator(+)**
3. **Relational operator (<,<=,>,>=)**
4. **Equality Operator (==,!=)**
5. **instanceof operator**
6. **Bitwise operator(&,|,^)**
7. **Bitwise complement operator(~ tild)**
8. **Boolean complement operator(!)**
9. **Short-circuit operator(&&,||)**

**Arithmetic operator (+,-,\*,/,%):-When we apply arithmetic operator between two variable then we get result type in the form of following**

**max(int type of a, type of b)**

**byte+byte=int**

**byte+short=int**

**byte+long=long**

**int+int=int**

**int+long=long**

**float+long=float**

**Example1:-**

**byte x=10;**

**int y=20;**

**int z;**

**z=x+y;**

**System.out.println(z);**

**//result in the form of int type if z is byte type then we will get compile time error.**

**Example2:-**

**int a=10;**

**float b=12.03f;**

**float c=a\*b;**

**System.out.println(c);**

**//result in the form of float type if z is int type then we wiil get compile time error.**

**Example3:-**

**int x=10;**

**float y=250.78f;**

**int z=y/x;**

**System.out.println(z);**

**Result:-**

**error: incompatible types: possible lossy conversion from float to int**

**int z=y/x;**

**String Concatenation:- + (plus) operator sometimes performed arithmetic addition operation and sometimes it is performed concatenation operation.**

**If at least one variable with +(plus) operator is String then +(plus) operator performed String concatenation otherwise its performed arithmetic addition.**

**Example:-**

**String s=”String”;**

**int x=10;**

**int y=20;**

**System.out.println(a+x+y);**

**Result:-String1020**

**Note :-Calculation started from left to right if all operator priority is same.**

**Example:-**

**int x=10;**

**int y=20;**

**String str="abc";**

**System.out.println(x+y+str);**

**Result:-30abc**

**Relational operator (<,<=,>,>=,):-**

**Relational operator we can apply for every primitive type except boolean if we try any other type then we will compile time.**

**Relational operator always results in a boolean (true/false) value.**

**Example:-**

**int x=10;**

**int y=20;**

**boolean b1=true;**

**boolean b2=true;**

**Test t1=new Test();**

**Test t2=new Test();**

**System.out.println(x<y);**

**System.out.println(b1<b2);//compile time error**

**System.out.println(t1<t2);// compile time error**

**equality operator(==,!=):- equality operator we can apply on every primitive’s types including boolean type also.**

**Example:-**

**int x=10;**

**int y=20;**

**boolean b1=true;**

**boolean b2=true;**

**System.out.println(x==y);**

**System.out.println(b1==b2);**

**Result:-**

**false**

**true**

**Assignment Operator(=):- ( = equal )we use assignment operator to assign the value to the variable like primitive ,reference.**

**Assignment operator we can divide into three parts:-**

**Simple/Normal assignment operator:-**

**Example:-**

**String =”vikas”;**

**int x=5;**

**Chained assignment operator**

**Example:-**

**int x,y,z;**

**x=y=z=10;**

**System.out.println(x+””+y+””+z);**

**Result:-10 10 10**

**Case1:-chaining assignment operator we can’t apply at the time of declaration otherwise we will get compile time error.**

**Example:-**

**int x=y=z=10;**

**Compound assignment operator:-**

**when a assignment operator is attached with some other operator called compound assignment operator.**

**The most commonly used compound assignment operators are:-**

**+=, -=,\*=,/=,%=,&=.**

**Example:-**

**int x=5;**

**x+=5;//it is equivalent to x=x+5;**

**System.out.println(x);//result:-10**

**x-=5;//it is equivalent to x=x-5;**

**System.out.println(x);//result:-0**

**x\*=5;//it is equivalent to x=x\*5;**

**System.out.println(x);//result:-25**

**x/=5;//it is equivalent to x=x/5;**

**System.out.println(x);//result:-1**

**x%=5;//it is equivalent to x=x%5;**

**System.out.println(x);//result:-0**

**Bitwise Operator(&,|,^):-This operator we can apply on all primitive except float.**

**There are three Bitwise operators:-**

**&-AND:-return true if both arguments are true.**

**|-OR:-return true if at least one argument is true.**

**^-X-OR:-return true if both argument are different .**

**Example:-**

**System.out.println(true&true);**

**System.out.println(false|true);**

**System.out.println(false^true);**

**Case1:-Bitwise operator we can also apply on integer primitive .**

**Example:-**

**System.out.println(3&2);//2**

**System.out.println(3|2);//3**

**System.out.println(3^2);//1**

**Bitwise complement operator(It is considered in unary operator):-**

**(~ tild)This operator we can only apply on primitive integral types only.**

**Example:-**

**System.out.println(~2);**

**Result:- -3.**

**boolean complement operator(it is also considered as unary operator):-(! boolean invert)This operator we can only apply on boolean primitive.**

**Example:-**

**System.out.println(!true);//result false.**

**Short-Circuit operator (&&,||):-It is similar to the bitwise operator but some basic difference**

1. **In Bitwise operator both arguments will be evaluated but in case short-circuit operator both argument evolution is optional.**
2. **Sort-circuit operator performance wise fast compare to Bitwise operator**
3. **Short circuit operator we can only apply on boolean type but Bitwise operator we can apply on integer and boolean type both.**

**&&-Short-circuit operator:-In this operator second argument will be evaluate if first argument is true.**

**class Test**

**{**

**public static void main(String[]args)**

**{**

**int x=10;**

**int y=20;**

**if(x<11&&++x<10)**

**{**

**System.out.println(y);**

**}**

**else**

**{**

**System.out.println(x);**

**}**

**}**

**}**

**Result:-11**

**||-Short-circuit operatior:-In this operator second argument will be evaluate if first argument is false.**

**Example:-**

**class Test**

**{**

**public static void main(String[]args)**

**{**

**int x=10;**

**int y=20;**

**if(x<11||++x<10)**

**{**

**System.out.println(x);**

**}**

**else**

**{**

**System.out.println(y);**

**}**

**}**

**}**

**Result:-10**

**Ternary operator:- The operator which acts on three variables (operand) is called Ternary operator.**

**Conditional operator:-( ? )This operator is considered in ternary operator and is used to evaluate boolean expression.**

**Syntax:-**

**x=(boolean expression)? This value assign if condition is true: This value assign if condition is false**

**Example:-**

**int age=18;**

**String validate=(age<=18)? "you are eligible for voting":"you are not eligible for voting";**

**System.out.println(validate);**

**Result:-you are eligible for voting**

**new operator:-we can use new operator in java to create an object.**

**Example:-**

**Test t=new Test ();**

**[] operator (square bracket open and close operator):-we can use square bracket open close operator to declare and create array.**

**Example:-**

**int [] x=new int[5];**

**Typecasting in java:-If we want to convert one data type to another data type is called Typecasting.**

**There are two types of Typecasting:-**

1. **Implicit typecasting**
2. **Explicit typecasting**

**Implicit Typecasting:-**

* **In implicit Typecasting java compiler is responsible to performed.**
* **If we are assigning smaller data type value to higher data type variable is called implicit Typecasting**
* **In this there is no chance of loss of information.**
* **It is also known as widening or upcasting.**

**Example:-**

**byte b=12;**

**short s=b;**

**System.out.println(s);**

**Result:- 12**

**Example:-**

**int x=10;**

**float f=x;**

**System.out.println(f);**

**Result:-10.0**

**Explicit Typecasting:-**

* **Explicit typecasting programmer is responsible to performed.**
* **If we to assigning higher data type value to smaller data type variable value is called explicit typecasting.**
* **In this there may be chance of loss of information**
* **It is also known as narrowing or dawn casting**

**Example:-Without type casting we get compile time error.**

**int x=10;**

**byte b=x;**

**System.out.println(b);**

**Result:- error: incompatible types: possible lossy conversion from int to byte**

**byte b=x;**

**Example:-To solve the above problem by using type casting.**

**int x=10;**

**byte b=(byte)x;**

**System.out.println(b);**

**Result:-10**