Language Fundamentals

01. Identifiers

02. Reserved words

03. Data Types

04. Literals

05. Arrays

06. Types of variables

07. Varg-args method

08. Main method

09. Command line argument

10. Java coding Standards.

01. Identifiers

-> A name in java program is called Identifier.

-> It can be class name, method name, variable name or label name.

Ex-

|  |
| --- |
| Class Test(1){ public static void main(2)(String(3) args[](4)){  Int x(5)=10;  }} |

* Rules to define java identifiers:-

1. The only allowed characters in java identifiers are atoz, AtoZ, 0to9, $, \_ by mistake if we are using any other symbol then we will get compiler time error.

Ex- total\_number//ok

total#//wrong due to # symbol

1. Identifiers should not start with digit.

Ex- total123 //ok

123total//wrong due to start with digit

1. Java Identifiers are case sensitive; of course java language itself is case sensitive programming language.

Ex-

|  |  |
| --- | --- |
| |  | | --- | | Class Test{  Int number=10;  Int Number=10;  Int NUMBER=10; | |

* We can differentiate variable names on the basis of case of the variable.

1. There is no length limit for java identifiers but it is not recommended to use lengthy identifiers.

Ex-

|  |
| --- |
| **int** xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxyyyyyyyyzzzzz=10 System.***out***.println(xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxyyyyyyyyzzzzz);//10 |

1. We can’t use reserved words as identifiers otherwise we will get compile time errors.

Ex-

Int x=10;//ok

Int if=10;//wrong due to reserved word if

1. All predefined java class names and Interface names can be used as identifiers.

|  |
| --- |
| **Ex-**  **int** String=777;  System.***out***.println(String);    **int** Runnable=888;  System.***out***.println(Runnable); |

* Even it is legal to use predefined java class and interface names as identifiers but it is not a good programming practice.

Q1. Which are the following are the valid java identifiers:

1. Ca$h //ok
2. Total\_number//ok
3. Total# //wrong
4. 123total//wrong
5. Total123//ok
6. all@hands//wrong
7. Java2store//ok
8. \_$\_$\_$\_$\_$\_$\_//ok
9. int//wrong
10. Int//right
11. Integer//right

02. Reserved Words

-> In java some words are reserved to represent some meaning or functionality; such types of words are called reserved words.

Fig1.

|  |
| --- |
| rw.png |

Reserved words for data types:

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

Reserved words for flow controls:

1. if
2. else
3. switch
4. case
5. default
6. while
7. do
8. for
9. break
10. continue
11. return

Keywords for Modifiers:

1. public
2. private
3. protected
4. final
5. abstract
6. static
7. synchronized
8. native
9. strictfp(1.2v)
10. transient
11. volatile

Keywords for Exception Handling:

1. try
2. catch
3. finally
4. throw
5. throws
6. assert

Class related keywords(6):

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

Object Related Keywords(4):

1. new
2. instanceof
3. super
4. this

Unused keywords:

1. goto

* Uses of goto created several problems in old language; hence SUN people band this keyword in java.

1. const

* Use final instead of const.

Reserved Literals:

|  |  |
| --- | --- |
| 1. true |  |
| 1. false |  |
| 1. null | Default values of object reference |

void return type keyword:

* In java return type is mandatory, if we are not returning anything then that method should be declared with void return type.

Note: In c language return type is optional and default return type is int.

enum keyword(1.5v):

* If we want to define group of named constants then we should go for enum.

Ex-

enum Month{JAN,FEB,……..,DEC;}

enum Beer{FO,KF;}

Conclusion:

* All 53 reserved words in java contain only lower case alphabets symbols.
* In Java, we have only new keyword but not delete because destruction of useless object is the responsibility of Garbage Collector.
* The following are newly added keywords in Java:-

1. strictfp(1.2v)
2. assert(1.4v)
3. enum(1.5v)

* goto and const are unused keywords in java; by mistake if we are using these keywords then we are going to get compile time errors.

|  |
| --- |
| 1. const but not constant 2. instanceof but not instanceof 3. strictfp but not strictFp 4. extends but not extend 5. implements but not implement 6. import but not imports 7. synchronized but synchronize |

* Which are the following list contains only java reserved words:-

1. new
2. delete
3. goto, constant
4. break, continue, return, exit
5. final, finally, finalize
6. throw, throws, thrown
7. notify, notifyAll
8. implements, extends, imports
9. sizeof, instanceof
10. instanceof, strictFp
11. byte, short, Int
12. None of the above

Ans. 12

* Which are the following are Java valid keywords:-

1. public
2. static
3. void
4. main
5. string
6. args Ans: 01,02,03

Note:

* In Java, it is impossible to find size of the object.
* In Java, it is impossible to find address of object.

Data Types

* In Java, every variable and every expression has some types.
* Every type is clearly defined.
* Each and every assignment should be checked by compiler for type compatibility and hence Java language is treated as STRONGLY TYPED PROGAMMING LANAGUAGE (type is very important and type checking is very strong).
* Java is not treated as pure Object Oriented Programming Language because several OOP features are not supported by Java (Operator overloading, multiple inheritance and etc...).
* Moreover we are depending on primitive data types which are non-object.
* Except boolean and char the remaining data types are considered as signed data type because we can represent both positive and negative numbers.

Fig1:

|  |
| --- |
| dataypes1.png |

Byte data type:

* Size 8bits(1byte)
* Max value : +127
* Min value: -128
* Range : -128 to 127
* Sign bit: 0🡪+ve, 1🡪 -ve

Note:

* The most significant bit acts as sign bit where 0 means +ve number and 1 means –ve number.
* +ve numbers will be represented directly in the memory whereas –ve numbers will be represented in 2’s complement form.

|  |
| --- |
| byte b=10.5;  CE: Possible loss of precision  Found: double required: byte |
| byte b=true;  CE: incompatible type  Found:boolean required: byte |
| byte b=”dugra”;  CE: incompatible type  Found: j.l.String required: byte |

* Byte is the best choice if we are handling data in Streams either from the file or from the network because file and network supported form is byte.

Short data type

* It is most rarely used in Java.
* Size : 2 bytes (16 bits)
* Range : -215 to 215-1

Ex1.

|  |
| --- |
| short s=32767;  short s=32768;//CE:P L P found: int required: short |
| short s=10.5;//CE: P L P found: double required: short |
| short s=”durga”;//CE: incompatible type  Found: java.lang.String required: byte |

* Short data type is the best choice if we are using 16 bit processor like 8086 but these processors are completely out dated and hence corresponding short data type is also out dated data type.

Int data type

* The most commonly used data type is int.
* Size: 4 bytes (32 bits).
* Range : 231 to 231 -1

Thursday, March 26, 2015

Long Data Type

* Sometimes int may not hold enough big value then we should go for long data type.
* Ex-

|  |
| --- |
| To hold amount of distance travel by light in thousands miles, for this int may not enough; compulsory we should go for long type. |

Ex-

long l=1,26,00\*60\*60\*24\*1000 miles.

* To hold the number of characters present in a big file; int may not enough hence the return type of length method present in file class is long.

long l=file.length();

* Size : 8bytes(64bits)
* Range -263 to 263

Floating Point Data Types

* All the above data types such as byte, short, int, long meant for representing integral values; if we want to represent floating point values then we should go for floating point data types.
* Floating Point Data type:

|  |  |
| --- | --- |
| Float | Double |
| 1. 5 to 6 decimal precision | 1. 10 to 15 decimal precision |
| 1. It follows single precession. | 1. Follows double precession. |
| 1. Range: -3.4e38 to 3.4e38 | 1. Range: 1.7e308 to 1.7e308 |

* If we want to use 5 to 6 decimal places of accuracy then we should go for float whereas if we want to 14 to 15 decimal places accuracy then we should go for double.

Boolean Data Type:

* Size : Not Applicable(Virtual Machine Dependent)
* Range: Not Applicable(But allowed values are true or false)

Char Data Type:

* Old language (C , C++) are ASCII code based and the number of ASCII characters are 256; to represent these 256 characters 8bits are enough and hence size of char in old language is 8bits(1byte).
* But Java is UNOCODE based and the number of Unicode characters are >256 and <=65536; to represent these many characters 8bit may not enough; compulsory we should go for 16bits (2bytes) and hence the size of char is 2 bytes.
* The below table represent the complete details about the primitive data types:-

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Data type | Size(in bytes) | Range | Corresponding Wrapper class | Default value |
| Byte | 1 | -27 to 27-1  (-128 to 127) |  |  |
| Short | 2 | -215 to 215-1  (-32768 to 32767) |  |  |
| Int | 4 | -231 to 231-1 |  |  |
| Long | 8 | -3.4e38 to 3.4e38 |  |  |
| Float | 4 |  |  |  |
| Double | 8 |  |  |  |
| Boolean | N/A |  |  |  |
| Char | 2 |  |  |  |

* Null is the default values for Object reference and we can’t use null for primitives types; by mistake if we are using null for primitive we will get compile time error.

Ex-

Char ch=null;

//CE: incompatible type found: <null type> required: char

Friday, 27 March, 2015

Literals

* Any constant values which can be assigned to the variable are called Literal.

Ex-

|  |  |  |  |
| --- | --- | --- | --- |
| Data type | Variable or identifier | Assignment  Operator | Literal or constant value |
| Int | X | = | 10; |

Integral Literals

* We can specify literals for integral by the following ways:-

1. Decimal Literals(Base 10)

Ex- int x=10;

1. Octal Literals(Base 8)

* Literals value should be prefix with 0(zero).  
  Ex- int x=010;

1. Hexadecimal Literals(Base 16)

* Allowed digits are 0 to 9, a to f.
* For extra digit (A to F), we can use both lower case and upper case characters.
* This is one of very few areas where Java is not case sensitive.
* Literals value can be prefix 0(zero) with x/X.
* These are the only area to specify to integral values for integral data types.
* Which of the following are valid:-

1. Int x=10;//ok
2. Int x=0786;//CE: integer number too large due to wrong octal digit 8(only 0 to 7 allowed)
3. Int x=0766;//ok due to right octal digit
4. Int x=0777;//ok due to right octal digit
5. Int x=0XFace; //ok due to right hex literals
6. Int x=0XBeef;//ok due to right hex literals
7. Int x=0xbeef;//ok due to right hex literals
8. Int x=0XBeer;//CE: ; expected due to r(only a to f allowed in hexadecimal literals)

Ex-

|  |
| --- |
| public class LiteralsDemo {  public static void main(String args[]){  int x=10;//decimal values  int y=010;//octal values  int z=0X10;//Hexadecimal System.out.println(x+"..."+y+"......"+z);//10....8.....16  //JVM always produce the result in the decimal form only  //whichever form I will assign integral values but jvm will convert into decimal and produce the result  }  } |

* By default every integral is of int type but we can specify explicitly as long type by suffix with l or L.

Byte and Short Literals

* There is no direct way to specify byte and short literals explicitly but indirectly we can specify.
* Whenever we are assigning integral literals to the byte variable and if the value within the range of the byte then compiler treats it as byte literal automatically and similarly short literals also.

Ex-

|  |
| --- |
| public class LiteralsByteShort {  public static void main(String args[]){  byte b=10;  byte b2=127;  byte b3=128;// int y=10L;//CE: Possible loss of precession , found int, required: byte  short s1=10;  short s2=32767;  // short s3=32768;// int y=10L;//CE: Possible loss of precession, found int, required: byte  }} |

Floating Point Literals

* Every floating point literal is by default double type but we can specify explicitly as float type by suffix with f or F.
* We can specify floating point explicitly as double type by suffix with d or D; of course this convention is not required.

Ex-

|  |
| --- |
| public class LiteralsFloatingPoint{  public static void main(String args[]){  float f=1234.50; //CE: Possible loss of precession, found double, required: float  double d=29893.90;  double d1=2422.90d;  int x=10i;//CE: ; expected  }  } |

* We can specify floating point literals only in decimal form but not in octal and hexadecimal forms.

Ex-

|  |
| --- |
| public class LiteralsDecimalOrHexOrOctalChecking{  public static void main(String args[]){    // float f=1234.50; //CE: Possible loss of precession  //found double, required: float  double d=029893.90;//It is treated as double  // double d1=0X2422.90d;//CE: malformed floating point literal    // double d2=0786; integer number too large  double d3=0XFace;//ok  double d4=0786.0;//ok  double d5=0XFace;//ok  }  } |

* We can assign integral literal to floating point literals directly and that integral literals can be specified either in decimal or octal or hexadecimal form.
* But we can’t assign floating point literals to the integral type explicitly.

Ex-

|  |
| --- |
| Int x=10.0;//CE: Possible loss of precision  //found: double required: int |

Saturday, 28 March, 2015

Char literals

* We can specify char literals as single character within single quotes.
* Eg:-
* The integral literals can be specified either decimal or octal or hexadecimal forms.
* Allowed range is 0 to 65535.

Ex-

|  |
| --- |
| Char ch=0XFace;  Char ch=0777;  Char ch=65335;  Char ch=65536;//CE: P L P found: int required:int |

* We can represent char literal in Unicode representation which is nothing but ‘\uxxx’ where xxxx(4digit hexadecimal number).

Ex-

Char ch=’\u0061’;’’(97)10-🡪 (0061)16  -🡪 a

* very escape character in java is a char literals.

Ex-

Char ch=’\n’;

Char ch=’\m’;

Char ch=’\t’;

Ex-

* \n---new line
* \k-----horizontal tab
* \r------carrise return
* \b------back space
* \f-------form feed
* \’-------single quote
* \”-------double quote
* [\\-------back](file:///\\-------back) slace

String literals

* Any sequence of character within double quotes is called string literals.

Ex-

String s=”java”;

Note:-

* Jdk1.7v enhancement with respect to literals :-

1. Binary literals:

* Until 1.6 version for integral data types, we can specify literal values in the following ways:

1. Decimal form
2. Octal form
3. Hexadecimal

* But from 1.7 version onwards, we can specify even in binary form also.
* Allowed digits are 0 an d1.
* Literals values should be prefix with 0 followed by b or B.

Ex-

Int x=0b100; Sopln(x);//4

1. Usage of \_(under Score) between digit of numeric literals

* From 1.7v onwards, we can \_ symbol between the digit of numeric literals.

Ex-

Double d=123456.789;

Double d=1\_23\_456.7\_8\_9;

Double d=123\_

* The main advantage of this approach is readability is improved.
* At the compilation time complier will remove automatically and hence after compilation the above code will become in the form double d=123456.789;

Conclusion:

* We can write multiple underscore symbol between the symbol also.
* Ex-

Double d=1\_\_2\_34\_45.9\_0;

Double d=1\_2\_\_3\_\_\_\_4456.9\_8\_90;

* We can use \_ symbol only between the digits by mistake if we are using anywhere else we will get compile time error.

Ex-

Double d=\_1\_2\_34\_9.9\_9;

Double d=1\_2\_34\_5\_.90;

Double d=1\_34\_5.9\_;