Identifiers

It is a name in java identifiers

It can be a class name , variable name , method name , label name

Eg: marked in color are identifiers

class Something{

public static void main(String[] args){

System.out.println();

}

class Example{

public static void main(String[] args){

String name = “pavan” ;

String result = name.toUpperCase();

System.out.println(result);

}

}

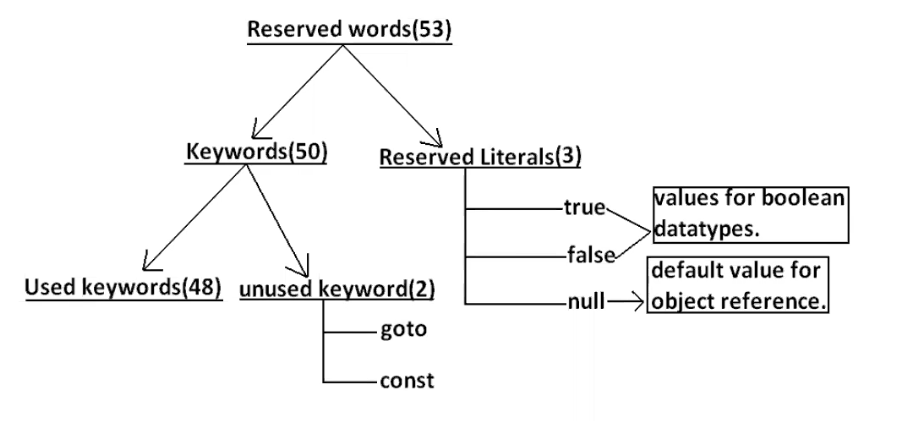
Rules for writing an identifier

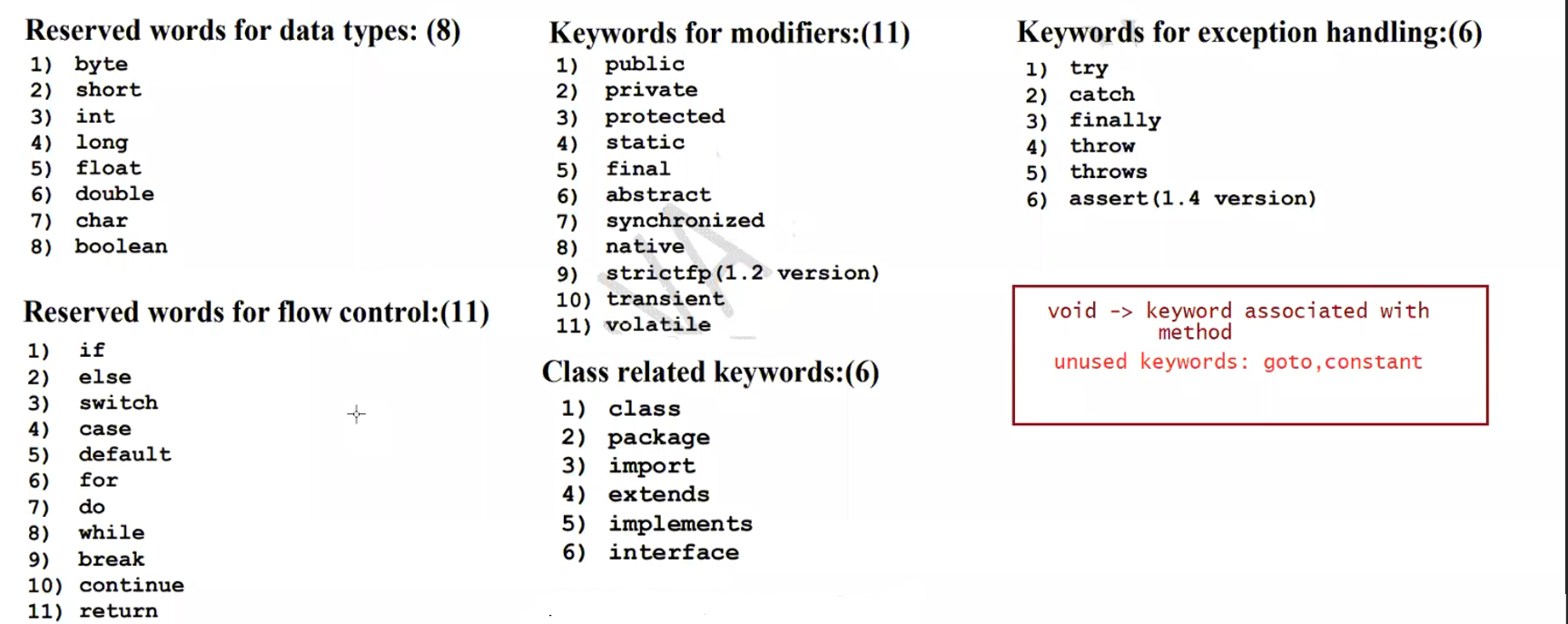
1. The only allowed character in identifiers is

A to Z , a to z , 0 to 9 , $ , \_

1. If we use any other character other than this would result in the error.
2. Identifiers should not start with digits
3. Identifiers are case sensitive
4. Cant use reserve words for identifiers
5. Predefined class names can be used as identifiers

Eg: Identifiers\_Eg1





Object related keywords (4)

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

note: All reserved words would start in lower case

All class names and interface names and class names should start with upper case

java is strictly / statically typed language. (type of variable should be known at compile time)

value assigned to variable is called literal

int a = 10 literal = 10

Data types

Every variable has type , and every expression has a type and all types are strictly typed , because java is strictly typed or statically typed language

compiler role-> compiler will check whether the value stored the variable can be handled by datatype or not it is called type checking/ strictly typed checking

Eg: byte a = 136 // C.E found byte

required byte

when to use byte datatype?

It is commonly used when handle data coming from stream, network.

Stream -> java.io package

byte , short , int , long - > follow base 2 format

float , double -> IEEE single precision format and double precision format

char -> follow Unicode

short :

short data type is not all used in java it is best suited for old processors like 8086.

int :

The most commonly used datatype for storing whole number is “ int “ by default

If we try to specify any number literal the compiler try to keep it in int only, but we can keep it in short and byte also

Eg: short a = 15

It shows C.E only if literal is out of range of the data type.

Compiler by default consider every whole number as int

whenever you are performing integer by integer the result will also be an integer irrespective of their result data type it is called truncated or rounding to zero

Eg:Data\_Types\_1

long:

when int is not enough to store big data long is used

when we work with large files data would come to java program in terms of gb’s

if data is too long then mark l (or) L at the end of the literal

otherwise it is treated as int and shows compile time error.

long a = 45l

float:

by default compiler will treat real number / decimal number as double you have to specify it with F(or) f similar to long

boolean / byte 1 byte , char/ short 2 bytes

To map primitive data as object in java from jdk 1.5 version wrapper classes are introduced

byte -> Byte …so on

note : String is a non-primitive data type in java .

char :

java follows Unicode format to store character .

all the data types use specific format to their data in the form of 0’s and 1’s . because computer can understand only 0’s and 1’s

utf-16 (universal transformation format) consists of 65536 which 216

16 bits = 2 bytes

So 2 bytes of memory is required to store the character in java

Note: Java doesn’t follow ascii but asci has 128 total characters which is 27 , for standardization of memory total 8 bits are taken . 8 bits = 1byte

Asci format they have given decimal , hexadecimal ,binary representation for 128 characters

Similarly they have give decimal , hexadecimal , binary representation for 65536 characters

But the representation for 128 characters in Unicode and asci is same in decimal, hexadecimal , binary

Eg: Char\_To\_Int

// In the above program char-> int conversion is done which is implicit implicit type casting .

You can give char datatype to int data type implicit type casting since char is smaller data type and int is bigger data type

char to int can be done and it is explicit type casting ,since int is bigger data type and char is smaller data type.

Double

If anything all the variables are of double data types including result data type.

zero by anything o/p is 0

anything by zero o/p is infinite

zero by zero o/p is is NaN

In java array and String is treated as object but not primitive datatype .

Java is object oriented ,but not pure object oriented because of primitive datatypes .

If you want to make any java project as pure object oriented we can make use of wrapper classes instead of primitive data types.

boolean

Eg: Boolean\_Example\_1

// you can use only boolean literals i.e true or false, but cannot use other literals other than ture (or) false

Type casting

Changing data from one data type to another data type is called type casting

Implicit type (or) automatic type casting (or) numeric type promotion (or) type promotion :

byte , short , int , long , float , double

byte b = 20 + 30;

// we can be do operations with literals and result can be put in same datatype if result is in that range .

The compiler treats 20, 30 literals as int , since they are in the the range of byte they can be stored in byte .

byte b = 20;

b = b+20;

// 2 line is note possible because , since we are using operand the compiler knows b as byte and treats 20 as int , the result would be in higher data type (int) but int can be stored in byte .

Explicit type casting :

When you do operations on some operators by default the compiler will treat them as int so you have to type cast it .

Eg: Explicit\_Typecasting

Eg: Type\_Casting\_2

// if you have given the number literal that is more than the max range of that data type , then jvm internally uses one formula to store that number

Minimum range + (result-maximum range-1)

The above formula is applied by the jvm in Eg2

Eg: Byte\_To\_Int\_Typecasting\_Expression

// byte to int implicit typecasting can be done , but in the program we are converting int to byte which is not possible then we have to type cast

Then we are adding an expression containing byte , int , byte

It is not stored in any variable and printed directly. The compiler adds all three and gives output as int .

Eg: Typecasting\_Underscore

For literals we can give \_ also but compiler will just remove it in the .class file

In the expression if you give two or more data types result will be on the higher data type , and typecasting depends on which datatype you are storing that result data type

Literals

Eg: Literals\_Underscore

Literals can be with underscore , but not in beginning or at the end .

Identifiers can start with literals but not literals

Eg: Typecasting\_Eg1

If operation is made on two different datatypes the result will always be higher data type.