CORE JAVA

- language fundamentals
- Literals Types of Literals
- Reading Runtime Values
- Accessibility Modifiers

by

Mr. Hari Krishna

DT- 05.12.2017

Language Fundamentals

(1) What is the need of a programming language?
(2) What is the need of a program?
(3) What do we need to develop and perform one operation?
(4) Give me one sample operation?
(5) What are the concepts called language fundamentals?
(6) Purpose of data types, operators, control flow statements and exceptions?
(7) Explain program execution flow with all above language fundamentals.
(8) Sample program.

(1) What is the need of a programming language?

Ans:-> Every programming language is meant for developing a program to implement an operation to be perform by a computer.

> Basically programming language are created for automating business operations to be performed by a computer.

> Automation will give several benefits over manual operations They are:,

(i) 24X7, 365 days we run our business
(ii) We can have good rel^n with customer,
(iii) Secured, accurate, efficient, fast transaction.

* An operation contains, below five activities.
  (i) reading and storing values
  (ii) performing validations and calculations,
  (iii) throwing errors of given values are wrong.
  (iv) generating results and returning results of given values are correct.
  (v) finally displaying result or error msg to end user.

* Every programming language will provide the same fundamental concepts to perform above five activities to implement logic for an operation on developing a program.

* Every language will provide below 4 concepts to perform above 5 activities they are.
  (i) Data types (Literals, variables and object).
  (ii) Operators
  (iii) Control flow statements.
  (iv) Exception handling statements.

The above 4 concepts are technically called as language fundamentals.

* The purpose of above 4 concepts is
  (i) Data types are used for creating a variable and objects for storing values in a program.
  (ii) Operators are used for performing validation and calculations to verify the given values are correct or wrong of correct for generating result.

(3) Control flow statements are used for controlling the execution flow of a program based on the given input values. If given values are correct it allows to execute calculation and if given values are wrong it allows to execute throwing exception logic.

(4) Exception handling statements are used for throwing and reporting an exception and further to catch the thrown exception.

* The concepts object and exception handling statements are exist only in OO programming languages such as C++, Java, .Net, PHP, Scala, Python, even we have them in PL/SQL We do not have object and Exception Handling statements in C language.

Summary:—

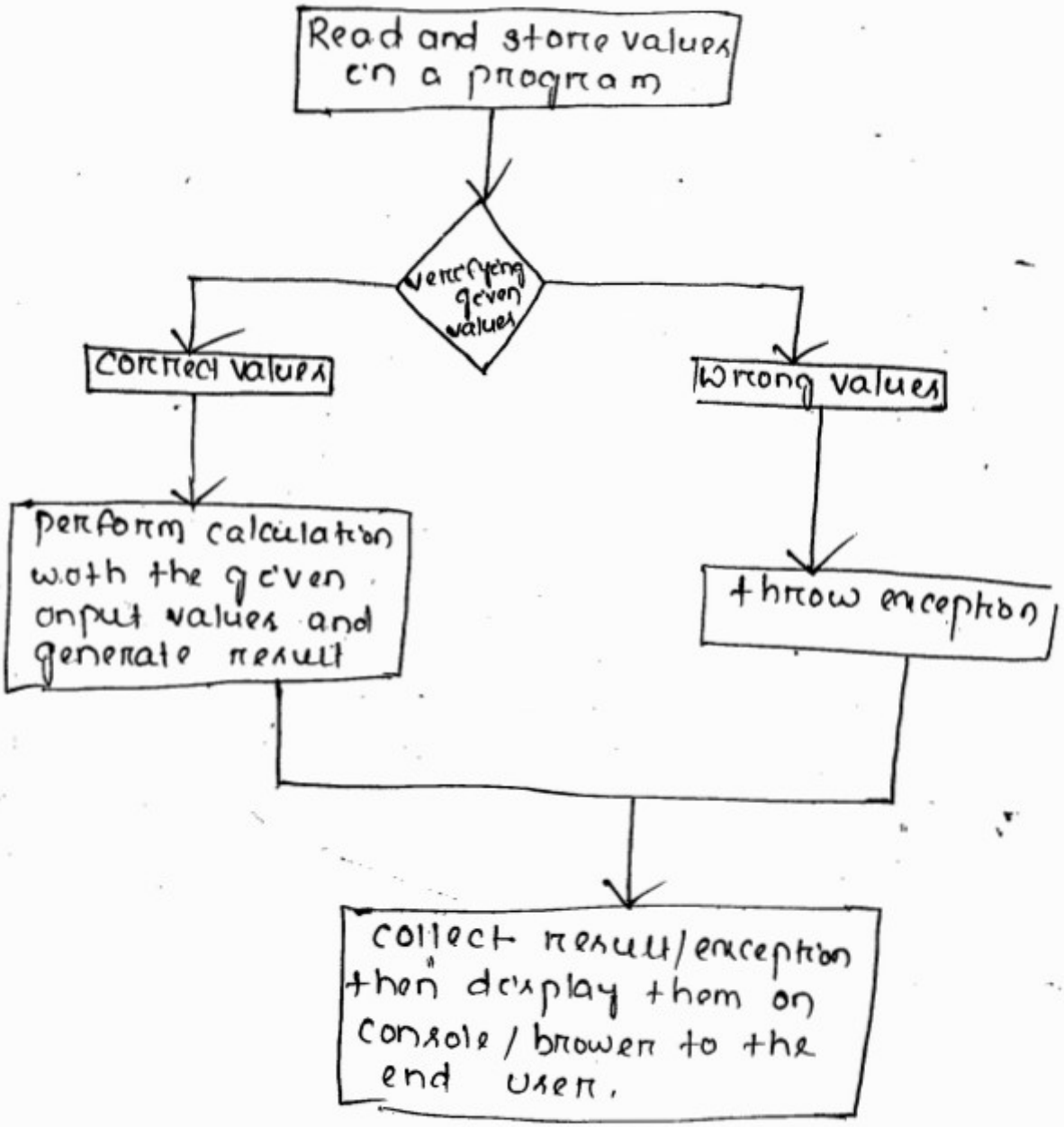
(1) We have 4 language fundamentals for developing an operation, they are:,

(i) Data Type - for storing values.

(ii) Operators - for performing validations and calculation.

(iii) Control flow statements - for controlling execution flow.

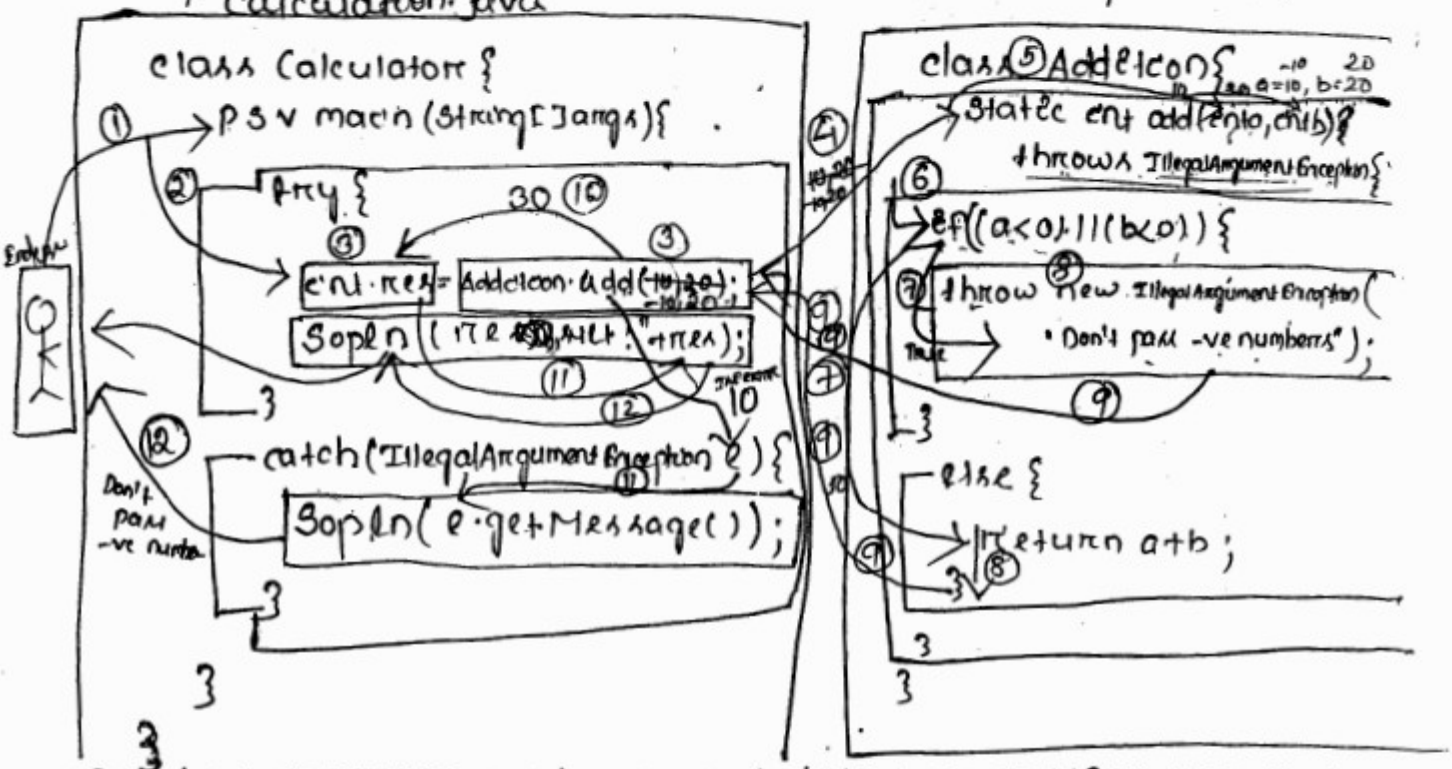
(iv) Exception handling statements - for throwing and catching and exception.


Read and store values in a program
verifying given values
correct values
wrong values
perform calculation with the given input values and generate result
throw exception
collect result/exception then display them on console / browser to the end user.

DT- 06.12.2017

Develop a program to add given two numbers. ~~by using~~. In this program you must use all above 4- language fundamentals. This program should accept only +ve numbers. If -ve numbers are passed you must throw exception.

Calculator.java



(1) In above program we have used data type int for specifying the return type of add method and for creating parameters for add method and also used for creating a variable in try block for storing result

(2) In above program we have used operators
<, || :- for performing validation to know the given values are -ve or not
+ :- for performing calculations for adding the given two numbers
= :- for assignment operation, means for storing values in the variable
new :- for creating IAE class object

(3) In above program we have used control flow statements if and else for taking decision which block of statements must execute based on condition return value.
* return for returning result to calling method main.

(4) In above program we have used EH statements.
* throw :- for throwing an exception with error msg if given inputs/arguments are wrong.
* throws :- for reporting/informing the exceptions throwing out from this method
* try :- for placing exception causing statements and further try to catch this exception in this current method itself.
* catch :- for catching the exception i.e. tracked in try block and further to place this caught exception handling code to take necessary actions on this exception either display error msg or doing some other work.

→ javac Calculator.java

compilation and execution

> We need to compile only Calculator.java,

> Compiler will compile Addition.java file automatically because we used class Addition in class Calculator.

> javac Calculator.java

|→ Calculator.class

→ Addition.class

Execution Case #1: input values are 10 and 20

> java Calculator

Result: 30

Execution case #2: Input values are -10 and 20

> javac Calculator.java

> java Calculator

Do not pass -ve number.

Execution flow of above program with case#1 : val 10,20

(1) End user executes the class calculator by passing values 10 and 20.

(2) Inside main method, inside try block res

(3) variable memory is created and then add() method is called by passive values 10 and 20

(4) Then the control goes to Addition class add() method.

(5) Inside add() method the input values
10 will be stored in parameter a
20 will be stored in parameter b.

(6) If condition validation logic is executed it generates false because given numbers aren't -ve.

(7) Then control goes to else block.

(8) else block logic is executed result 30 is generated.

(9) with this result 30 control returns to main method.

(10) this result 30 is stored in the variable res

(11) next this result is passed and substituted in Sopln()

(12) finally the result 30 is displayed on console to end user.

Execution flow above program with case#2 vals - 10,20

Open Calculator.java file and change values 10,20 to -10,20 then save program, compile and then execute it.

The steps 1,2&3 are same as in case #1.

(4) Control goes to Addition class add method with the values -10 and 20

(5) These values are stored in parameters a and b respectively.

(6) Further if() condition is executed with these values and generated result true, because given values are -ve.

(7) then control goes into if block, executes if block code.

(8) creates exceptions IAE object with given error message.

(9) then throws this IAE to main method.

(10) then main method passes this IAE to catch block, catch block with catch this IAE and stores in var e.

(11) then catch logic is executed, e.getMessage() method will retrieve the error message from IAE object.

(12) finally error message Donot pass -ve number is displayed on console to the end-user.

Run below cases to understand the functionality of throw, throws, try and catch.

Case1:- Remove throws IAE, then compile and execute Calculator class by passing -10,20.

Result:-

Case#2:- Place throws IAE, now remove, now remove throw new IAE() then compile and execute Calculator class.

Result:-

Case#3:- Place throw new IAE(), now remove try & catch block in Calculator class, then compile and executed.

Result:-

Data type, Variables, Arrays, Objects

(1) What is datatype?
(2) Why data type?
(3) When should we use datatype?
(4) Types of data types?
(5) Explanations on each datatype?
(6) What is a variable and use of variable?
(7) Limitation of a variable, its solution?
(8) What is an array, how to create an array in Java?
(9) Limitation of an array, its solution?
(10) What is a user defined data type/custom data type?
(11) What is a class and why a class is called UDD?
(12) One example creating a user defined data type?
(13) What is an object, class and instance?
(14) Steps to create real world object in java world
(15) Sample projects development
(16) Interview questions and OCJP Bits.

Need of this chapter

→ In this chapter we will learn creating variable and object type memory for storing single and multiple values in a program as part of performing one operation.

→ To perform one operation we must store values in program. To store value we must allocate memory. Here we need to provide information above the number of bytes and type of memory must be allocated to store a value

→ To specify number of bytes and memory type to the compiler and JVM we must use some set of keywords. Those set of keywords are collectively called as datatypes.

→ Hence we can say data type is used for allocating memory for storing value in a program by specifying the required numbers of bytes and memory type.

* What is a datatype?. What is the use of datatype?

09/10/17

* A keyword that is used for creating variables and objects for storing single or multiple values is called datatype.

* A variable is a named memory location that is used for storing one value or one object reference.

* An object is also a memory location but it is used for storing multiple values/object.

* A datatype keyword provides below information

(i.) Type of the memory.
(ii.) Size of the memory (how many bytes).
(iii.) type of value and Range of the value.
(iv.) Diff. operators allowed to apply.
(v.) Result type Comes out from an expression

which ever the keyword provides above information that keyword is called data type.

For example:-

int, long, double, char, boolean -------- keywords are called datatypes.

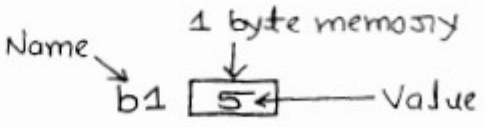
For example:-

If we use the datatypes bytes for creating memory

(i.) It will create integer type memory.
(ii.) It will create 1 byte memory.
(iii.) It allows to store integer value in the range -128 to 127.
(iv.) It allows AO, RO, EO, BWO to apply but it does not allowed logical operator.
(v.) If we use byte memory in an expression, it returns result as int type value, but not as byte type value.

For example:-

```
byte b1 = 5; ✓
byte b2 = 5.0; X
byte b3 = "ab"; X
byte b4 = true; X
```



b1, b2, b3, b4 are called as variable

* Different types of datatypes

Java supports two types of datatypes

(1) primetive data types (8)

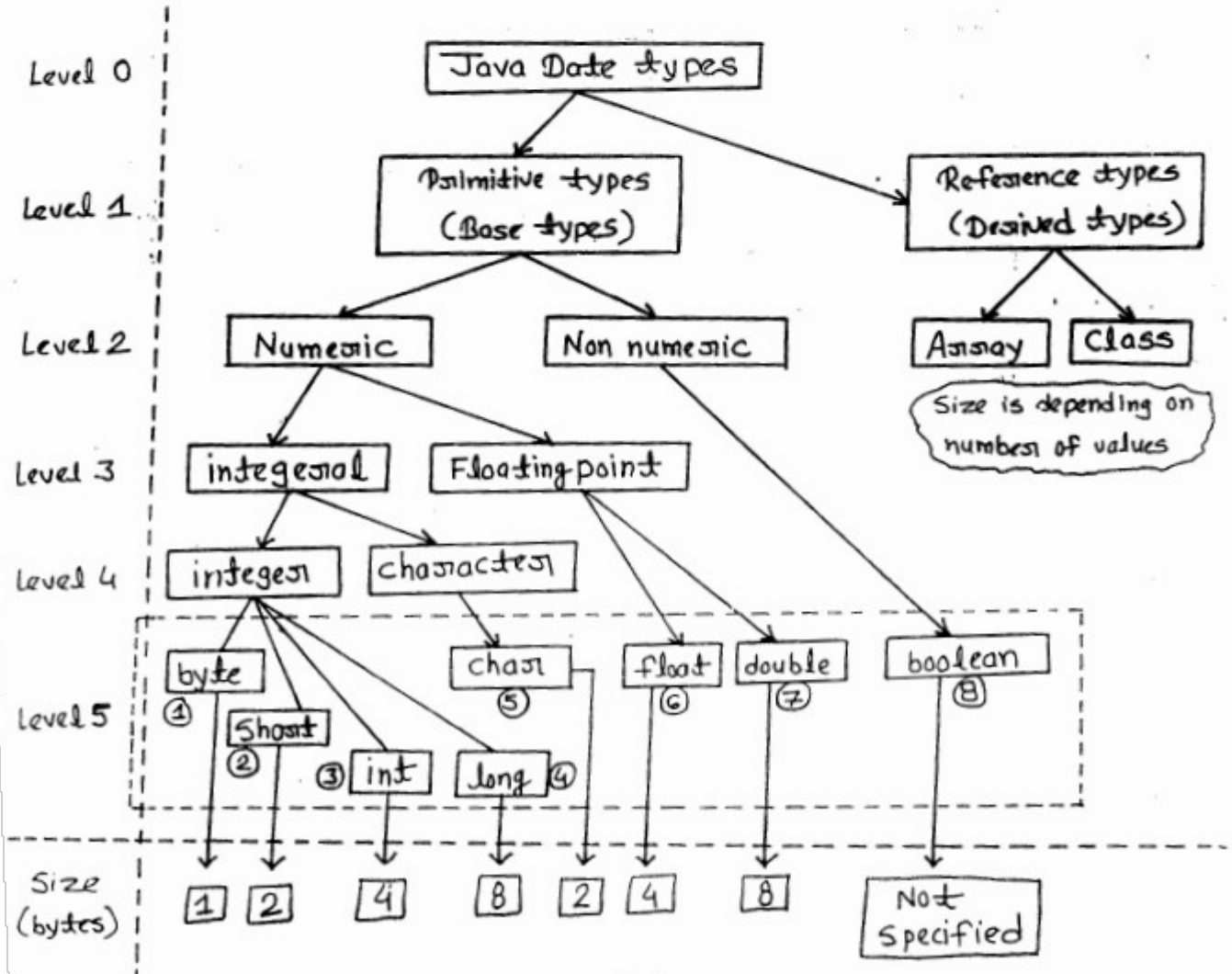
(2) referenced data type (5)

→ PDTs are used for creating a variable (one value memory). for storing one mathematical value.

→ RDTs are used for creating an object (multiple values memory) for storing multiple values on objects of same type or diff. type.

* Java Datatype Hierarchy :-

10/10/2017



DT-10.12.2017

Q. Why do we need 8 primitive datatypes and 5 reference datatype?

Ans:- * Based on the type of the value and the range of the value.

* Java supports 8 PDTs.

The data types int, double, char, and boolean are given based on ~~the~~ type of the value.

- int → for storing integer values
- double → for storing floating-point values
- char → for storing character values.
- boolean → for storing logical values true and false.

The additional datatypes byte, short, long and float are given based on range of the data.

- → for storing lesser range value we must use byte, short, float.
- → for storing larger range value we must use long.

Below chart will show you maths type and their matching Java types:-

Math types	Sample values	Java types
Integers	10, 20, 30, ...	byte, short, int, long
Floating-point	10.0, 20.4, 30.7, ...	float, double
characters	'a', 'b', 'c', '#', '1', ..	char
Boolean	true, false	boolean
String	"a", "b", "Hari"	Java.lang.String

Q. Why do we have 5 Referenced datatypes?

Ans:- Java supports 5 referenced data types, they are;

(i) array
(ii) class
(iii) interface
(iv) enum
(v) annotation

→ For storing multiple as one group by using single name.

→ By default a PDT variable can store only one value at a time, But in project we must store multiple values at a time. In this we want to. RDTS

→ For storing multiple values of same type we must use array.

→ For storing multiple values of. diff. type we must use class.

→ For storing multiple constant values with fixed names we must use enum

→ interface and annotation are RDTs but they are not used for storing values, rather they are used for declaring multiple methods of an object to be implemented by sub type programmers.

Understanding all points related to Java data type in comparision with C-language:—

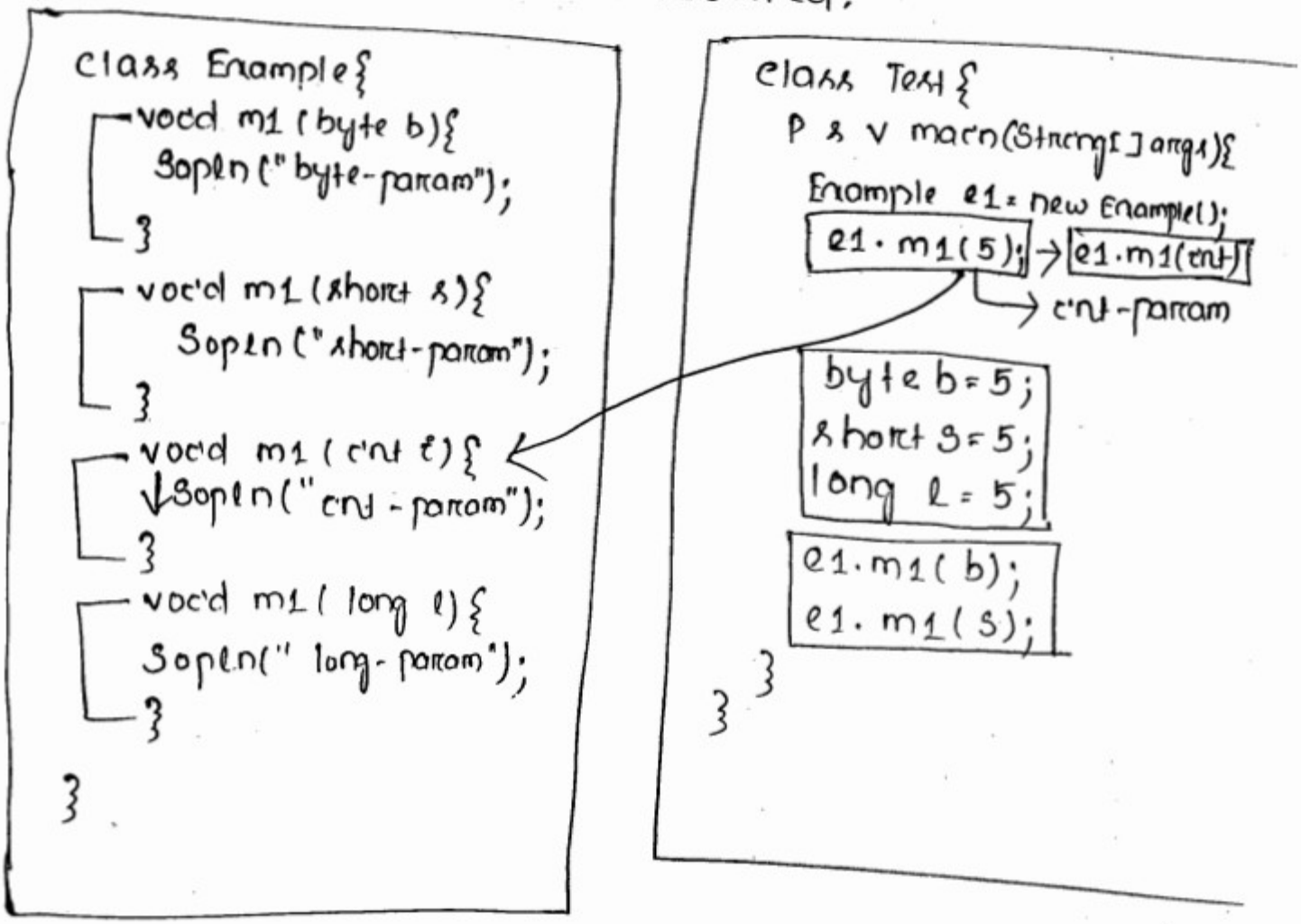
(1) Integer Data type (byte, short, int, long) points:—

Ans:-

(i) All integer values represented by 4-data types, (byte, short, int, long).

(ii) By default every integer number is considered as int type, it means when we use any integer no. in a program, compiler and JVM will consider this number as int type number.

(iii) Ex, Consider below program, identify which parameter method is executed.



→ In above program inside main method we called m1 method by passing the argument 5. Compiler and JVM will consider this value as integer type value, then compiler and JVM will execute m1(int) hence we got O/P int-param.

(iv) To represent a integer number as byte, short, long type we must prefix / suffix cast operator or a special alphabet.

Ex:-

* to represent 50 as byte we must explicitly write below code.

```
byte b1 = 50; ← variable assignment
(byte) 50     ← casting
 ↑
cast operator
```

* to repren 50 as short type, we must write below code.

```
short s1 = 50; ← variable assignment
(short) 50     ← casting
 ↑
cast operator
```

* to represent integer number 50 as long type, we must explicitly write below code.

```
long l1 = 50; ← variable assignment
(long) 50 ;
 ↑
 ——— cast operator

50L / 50l
 ↑   ↑
 ——— ——— suffix character
```

Rule:- There is no suffix character to represent integer number as byte & short. we must use as cast operator on variable assignment as shown in above lines.

NOTE:- To call and execute byte, short and long type methods in above program in previous page, we must pass value 5 as below;

```
byte b1=5;
short s1=5;
long l1=5;
```

```
m1(b1); ⟶ byte - param ✓
m1(s1); ⟶ short - param ✓
m1(l1); ⟶ long - param ✓
```

`m1((byte)5);` ✓	`m1(5B);` ∝ (CE)
`m1((short)5);` ✓	`m1(5S);` ∝ (CE)
`m1((long)5);` ✓	`m1(5L);` ✓
	`m1(5l);` ✓

(V) In variable assignment, we must follow one rule i.e value type and range must be lesser than variable type range.

For ex:-

```
int i1=5;   (5 is a int type, so allowed to store in int variable)
int i2=5L;  (5L is a long type, so not allowed to store in int variable)
```

but we can assign int type value in lesser range data types byte and short variables, because these data types don't have their own type of special values.

For ex:-

```
byte b1=5; ✓
short s1=5; ✓
```

Rule:- We cann't store all integer numbers in byte and short variables. We can store only the integers which are in the range of byte and short.

byte range is "−128 to 127"

short range is "-32,768 to 32,767

hence we can store int values in byte and short variables in this range.

```
byte  b2 = 560;      X
short s2 = 56000;    X
```

* Special point on large integer number :——

Large integer numbers are also consider by compiler and JVM as int type numbers. But larger integers are not in the range of int data type. Hence ~~compiler will throw error~~. Hence when we use large integer number directly, compiler will throw error.

The compiler throwing error is "integer number is too large".

Ex:-

```
Sopln(12345678909);
```
CE : integer number too large.

Closely observe we have used any variable assignment, but still we got error because this number range is exceeds the range of int data type hence we got error.

To ~~use~~ use large ~~integer~~ integer number, we must suffix this number with L. Then compiler will treat this large number as long type, because this numbers is in the range of long data type. We don't get error

Ex:- Sopln(12345678909L); ✓

Identify ~~CE in~~ value type and CE in below statements.

5B
5S

Don't show your talent
5B is not byte
5S is not short

5 is int type
5L is long type

```
int i1 = 5; ✓
long i1 = 5; ✓

int i2 = 5L; X
long l2 = 5L; ✓

int i3 = 12345678909; X
long l3 = 12345678909; X
long l4 = 12345678909L; ✓

(byte)5;
(short)5;
```

→ In above two statement i3 & l3 we got CE not bcz of assignment, we got error beacuse value not in the range of int, value representation is wrong. We must suffix L to this number to represent it as long to solve this error.

8* Compared to 'C'-language, in java int data type size can't changing from one processor to another processor. In Java int data type size is fixed 4 bytes. Hence we will get always same o/p. across all systems even though process is change

Hence java is robust.

In 'C'-language, int data type size is changing

> for 16 bit processor its size is 2 bytes
> for 32 bit processor Its size is 4 bytes

because size is changing, we can not get same result across all systems. Hence 'C' isn't robust.

DT-11.12.2017

* In project to store integer we always use int type variable.

> If we want to store large integer number we will use long.

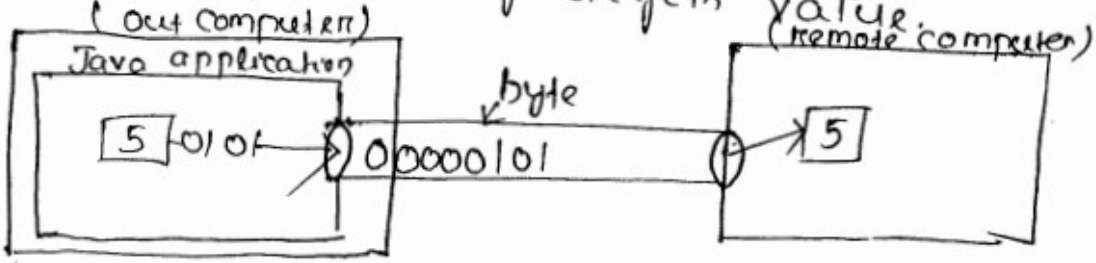
Ex:- for storing acc.Number, debit and credit card number aadhaar we must choose long data type variable

> In regular program calculation we never use byte and short types.

> short data type is used for storing values for processor level calculation.

> byte data type is used for sending data to storefiles to or send data through network in terms of binary digits value.

Ex:-



19) Compiler code optimization

If we use int for storing a number for lesser range numbers we loss memory.

Q. How will we use to solve this problem?

We no need to worry about memory loss, compiler s/w will optimize code to get best performance by consuming less memory. When we compile program, compiler s/w internally verifies how much memory it's required for storing the assigned value based on the type, it will prepare byte code instruction in .class file for JVM

(10) Size, Range and default values of integer data types.

Data type Name	Size(byte)	Range	Default value
byte	1	-128 to 127 (-2^7 to 2^7-1)	0
short	2	-32,768 to 32767 (-2^{15} to $2^{15}-1$)	0
int	4	-2147483648 to 2147483647 (-2^{31} to $2^{31}-1$)	0
long	8	-9223372036854775808 to 9223372036854775807 (-2^{63} to $2^{63}-1$)	0L

Floating-point datatypes points

(i) To store floating point number Java supports two data types float and double.

(ii) by default every floating point number is of double type. it means when we use floating point number in program compiler and JVM will treat this number as double type.

(iii) hence we can't assign floating point number directly to float variable, compiler will throw error.

Ex:-
```
float f1 = 10.0;   ✗ CE
double d1 = 10.0;  ✓
```
(double)

(iv) to represent floating-point number as float type and to store floating-point number in float variable,
We must suffix f or F to this number

Ex:-
```
float f2 = 10.0F
```

(v) Compared to C-language, in C-language we can store floating point number in either float or double variable even we can store in int type variable, C compiler won't throw error, it will do required conversions

	In C	In Java
`int i1 = 10.0;`	✓	✗
`float f1 = 10.0;`	✓	✗
`double d1 = 10.0;`	✓	✗

In Java,
If we want to assign floating point number to;
(1) float variable, we must suffix f or F
(2) int variable, we must prefix cast operator (int)

```
int   ia = (int) 10.0;  ——  ia [10]
float fa = 10.0F;       ——  fa [10.0]
double da = 10.0;       ——  da [10.0]
```

(V0) In regular programming calculations we don't use float, always we will use double, because of two reasons.

(1) by default every floating-point number is treated as double. If this floating-point number want to be used as float method caller must explicitly suffix (f/F) It more burden to programmer. If we choose double we no need suffix any character.

(2) float is single precision datatype means it occupies 4 bytes hence in large comutations we will not get complete result. If we choose double, because it is double precision data type means it occupies 8 bytes, we will get complete result.

NOTE:- If you ensure that the result range ~~will be~~ is with in the floating range you can choose float data type for saving memory.

(NCC)

Data type Name	Size	Range	Default value
float	4	1.4012984643248 1707 to 3.4028234663852886 0e+38	0.0F
double	8	4.94065645841	0.0

Character datatype points

(i) Any single alphabet or digit or special character placed inside single quote is called character.

Ex:- 'a', '1', '@' are characters

a 1 @ aren't characters.

(ii) Every character by default is treated as char type to store a single character we must create char variable.

Ex:-
```
char ch1 = 'a';
char ch2 = '1';
char ch3 = '@';
```

(iii) In a single quote.

> only one character is allowed
> only one space allowed and
> empty character is not allowed.

(iv)
```
char ch4 = 'ab'; ✗
char ch5 = '10'; ✗
char ch6 = '@9'; ✗
char ch7 = ''; ✗ (no space)
char ch8 = ' '; ✓ (single space)
char ch9 = '  '; ✓ (double space)
```

(iv) > Compared to c-language, in java language char data type is 2 bytes, because java char data type is used to represent unicode character set. Unicode character set range is 0-65535, which occupies 2 bytes memory.

> C language char data type is 1 byte, because c language supports only ASCII character set which contains 0-255 characters. Hence 1 byte memory is sufficient.

> Java supports unicode character set for developing Internationalization (I18N) application

> I18N application means, it should display content in the ~~country~~ current country native language in which it is running.

(v) Default, Size, Range of char data type.

Data Type Name	Size	Range	Default value
char	2	0 to 65,535	one whitespace

DT-12/12/2017

About boolean value

(1) Java support speccially a data type called boolean for representing conditional/logical values true and false.

(2) In a java program to store true and false we must use boolean variable as shown below

```
boolean bo1 = true;
boolean bo2 = false;
```

(3) In general we we'll use boolean value for checking condition. It is mainly used in if(), while(), for() for executing a block of statements either 0 or 1 time or 0 or n times.

```
if(true){          | while(true){       | for(;true;){
  -----            |   ------           |   ------
}                  | }                  | }
```

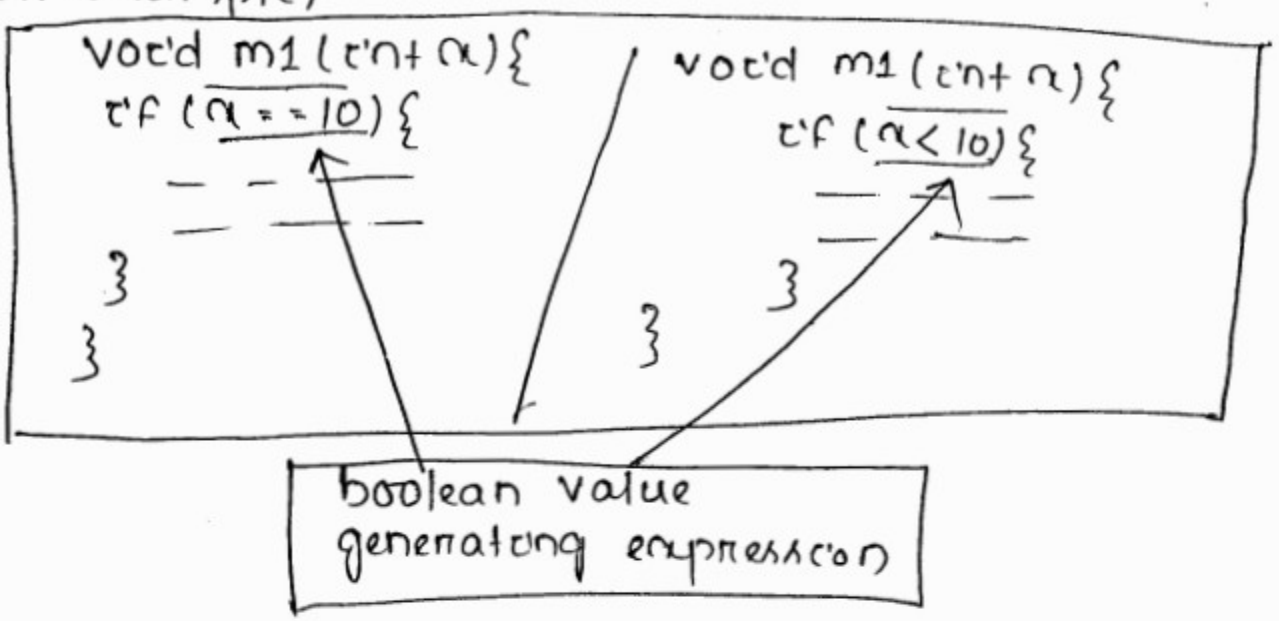
if(false){}/while(false){}/for(;false;){}

(4) → In projects we don't directly place boolean values true/false in if()/while()/for() condition as shown in above code rather we we'll place an expression that generates boolean value true/false dynamically with the given input values.

→ To generate boolean values dynamically, inside an expression we must use boolean value generating operators such as;

* relational operators
* equality operators
* logical operators

For example,



(5) boolean data type is used in validations / verifications / checking but not in calculations. It means to confirm the given input value is correct or wrong, we will use boolean data type value where as byte, short, int, long, float, double, char data types are used in calculations to generated o/p value from the given inputs.

Ex:-

```
class Addition{
    static void add(int a){
        if(a<0){
            System.out.println("Do not pass -ve num");
        }
        else{
            Sop(a+20);
        }
    }
}
```

It is a validation, RO is used, boolean value is generated

It is a calculation, AO is used int number is generated.

(6) Compared to C-language

* In C-language we don't have boolean data type and also we don't have true/false.
* In C-language we use,
  → 0 as false and
  → any -ve or +ve integer as true.

Hence in if()/while()/for() conditions we will use int number or any expression that generate int number to execute the block of statement.

Ex:-

	in 'C'	in java
if(0){ }	✓	✗
if(1){ }	✓	✗

What is the O/P from below program?

```
class Example{
    static void m1(int a){
        if(a = 10){ //CE:
            Sopln("Hi");
        }
        else{
            Sopln("Hello");
        }
    }
    P S V main(SL Jargs){
        m1(10);
        m1(20);
    }
}
```

(assignment operator, comparision operator)

To solve above CE we must use '==' operator which is equality operator & it generates boolean value based on given a value.

if a=10, it generate true, we will get o/p 'Hi'
if a≠10, it generate false, we will get o/p 'Hello'.

(7) boolean data type size, range and default value

Datatype	size	Range	Default value
boolean	1	false/true	false

About String data type

(1) String is a referenced data type. It is a class. It is a predefined class created by SUN. It is available in java.lang package in java.base module.

(2) It is used for representing sequence of characters placed in " ". It means for storing a sequence characters as one group we must create String class type ref var as shown below

```
String s1 = "Hari krishna";
```
It is a String

```
Strong s1 = Hari krishna;
```
CE: It is n't a string

(3) Rule:- If we want, compiler and JVM treat a sequence of characters as String type, we must place them inside " ". If we use a sequence of characters directly in a program compiler will not treat it as a String, throws error.

```
String s2 = Hari krishna;
```
not string
CE

(4) When we want to store a name, course, dept, alpha-numeric number like PAN number, engNum, bikeNumber. We must use String data type variable.

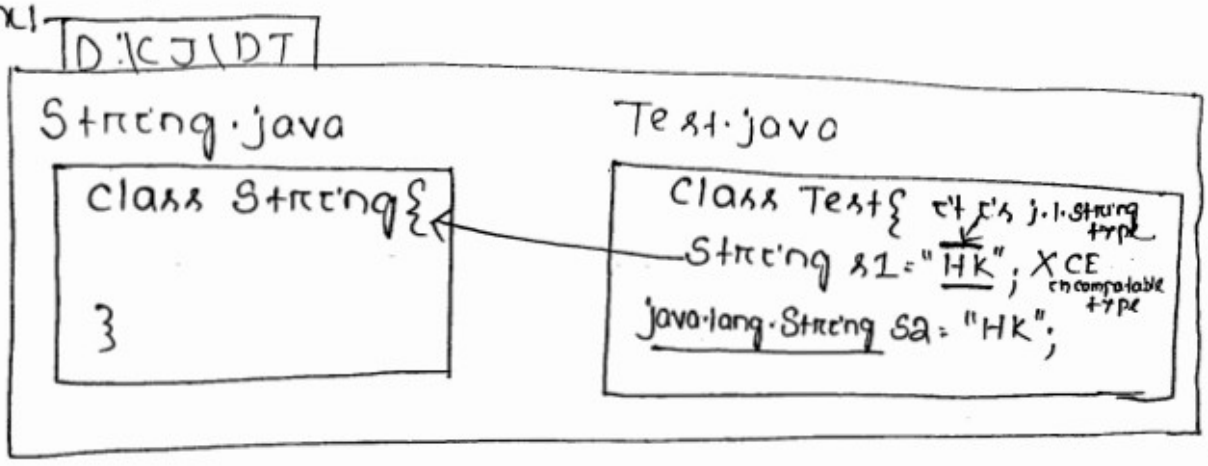
```
String name = "HK";
String course = "Java";
```

```
String bikeCName = "Bajaj";
String engNum = "ABC123PQ";
String bikeNum = "TS09BJ9999";
```

(5) Inside " " we can 0 or n number of characters. It means an empty String is allowed.

```
String s1 = "";
String s2 = " ";
String s3 = "   ";
```

(6) Note :- We are allowed to create our own class with name String. If we create our own class with name String. We must access pre-defined String class with its fully qualified name. It means we must access it by using its package name.



(7) String data type size, range and default value

→ String data type,

size is (number characters *2) and no limit in range, you can store any number of characters

→ default value is null

DT-13·12·2017

* When should we use datatype in program?

Ans:- We must use datatype, in the below 3 situations.

(1) For creating variable.
(2) For creating method parameter.
(3) For creating return type.

If we not required of creating variable/parameter or return type then no need to use return type.

Q. What is the output from the below code ?

```
class Example {
    void m1(int x) {
        if (x=10) {
            Sopln("Hi");
        }
        else {
            Sopln("Hello");
        }
    }
}
```

```
class Test {
    p s v main(string [] args)
    {
    Example e1 = new Example();
    e1.m1(5);
    }
}
```

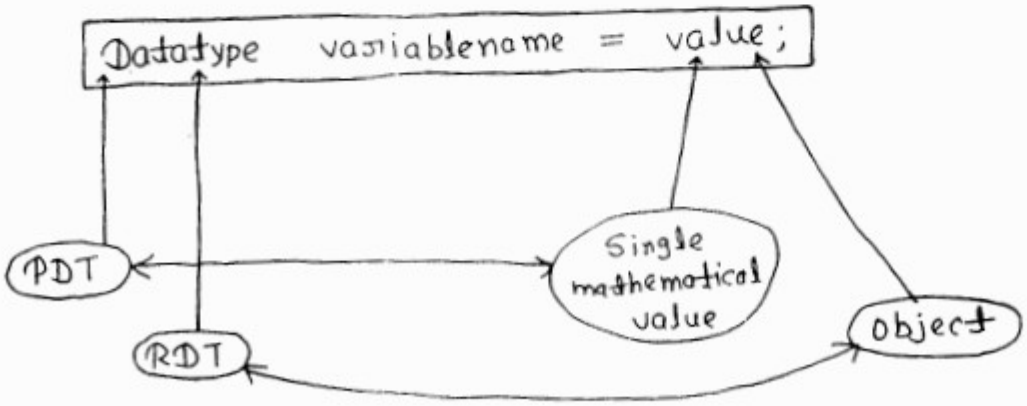
Hello chitti, closely look at condition expression (x = 10), the operator is not equality operator (==), It is assignment operator hence int value is generated, compiler will throw error.

To compile this program we must use (==) operator, then no compiletime error, we will get output "false".

Q. What is a variable, Syntax to create variable ?

⇒ * A variable is a named memory location, it is used for storing single mathematical value or one object reference.

* A variable is created by using datatype (either PDT or RDT).



Q. What is the difference between Primitive variable and referenced variable ?

⇒ Primitive variable	referened variable.
(i) The variable that is created using primitive data type is called primitive variable.	(i) The variable that is created using referenced datatype is called referenced variable.
(ii) In PDT variables, we can store one mathematical value directly.	(ii) In RDT variables, we can store one object reference but not mathematical value.
(iii) We can use PDT variables in calculations, because it generates new number which is meaning-ful. e.g. int a = 50; a [50] int b = 60; b [60] int c = a-b; c [-10] Sopln(c);	(iii) We can not use RDT var. in calculations, compiler will throw error. We can not use RDT variable in calculation, because it generates new reference number, in which object is not exist. example :- String s1 = new String("H"); String s2 = new String("K"); String s3 = s1-s2; X CE Sopln(s3);

12/10/17

Q. What Compiler and JVM will do when we create variable in the program ?

For example :- `int a = 5;`.

⇒ Compiler software will only verifying syntax of the var. creation statement. It will verify below 4 points :-

(i) Data type is valid or not and it is existed or not.

(ii) All 6 identifier's rule followed or not, in variable name creation.

(iii) The assigned value compatible with this variable datatype or not.

If all above 3 rules are satisfied this statement is compiled fine.

(iv.) It also verifies value range <= Datatype range.

Example:-

```
class test {
    public static void main (String [] args) {
      int a=5; ✓
      int1 b=6; X CE: Can not find symbol class int1.
      X ; X Datatype missing.
      X = 10; X Symbol missing.
      byte b1 ; ✓
      byte b1 = 20; ✓
      byte b1 = 1000 ; X CE: Possible loss of precision.
      byte b2 = 'R'; ✓
      byte b7 = true; X CE: Incompatible type.
      byte 1a = 10; X CE: Not a statement bcoz starts with 1.
      byte b1 ; ✓
      byte static = 10; X CE
      byte String= 10; ✓
      String s1 = "ab"; ✓
      String s2 = true; X CE: Incompaitible type.
      String s3 = 10; Warning:- Not allowed on Java9 onwards.
      byte _ = 10;
      byte $ = 20; ✓
      byte 1 = 10; X CE: Not a statement.
    }
}
```

byte 1a = 10; X CE: Not a statement.

We can not start identifiers with a digit, because, it is not allowed to start identifiers with a digit.

* If we create identifiers with single characters i.e. just with digit like int 5 = 10;
then, in the next lines if we write or use this 5 as Sopln(5); Compiler can't decide whether this 5 is value or variable 5.

Hence, we are not allowed to create an identifiers starts with a digit.

We can start identifiers with a alphabet because in a prog. we will use alphabet as value either in ' single quote (') or in double qote.("). Hence No confusion to compiler.

```
Sopln(a);  <---- Here a is variable
Sopln('a'); <---- Here a is value of type char
Sopln("a"); <---- Here a is value of type string.
```

*. Rules and Summary :-

1) If we want to create variable, we must use datatype, without using datatype if you just specify variable name or if we just specify variables with value then we will get Compile Time Error

b1; XCE: Not a statement (Datatype is missing)

b1 = 10; XCE can not find variable b1, because Datatype is not used and It is not declared in previous line.

byte b1; // Declaring means Creating a variable.

b1 = 10 // storing values in the above variable b1.

byte b2 = 30; // Declaring and storing.

2.) If we use some word in datatype place others than 8 datatype keywords b, s, i, l, f, d, c, bo then compiler will treat this word as class name, then if there is no class found with this word compiler will throw error, Can not find symbol class.

```
int i1 = 5; ✓
Int1 i2 = 6; X
Ex i3 = 7; X
```

> CE: Can not find symbol class Int1, class Ex

3.) Compiler never create memory for the variable, compiler always only checks variable creation syntax by considering above 4 points specify in previous page.

4.) Only JVM will allocate memory for the variable. Hence, JVM executed variable creation statement by allocating memory by no. of bytes based on given datatypes. Named this memory by given name. Store given value in the memory.

For e.g. :- int a = 5; ⟶ a [5] (4 bytes)

5.) Will JVM store the given decimal value as it is as shown above?

⇒. No, it converts given decimal value into its equivalent binary numbers 0's and 1's. as shown below.

int a = 5;

	1	2	3	4
a	00000000	00000000	00000000	00000000

It is not array memory, it is variable memory with 4 bytes.

These fields are filled from Right to left.

14/10/17

*. Real time project based code on datatypes.

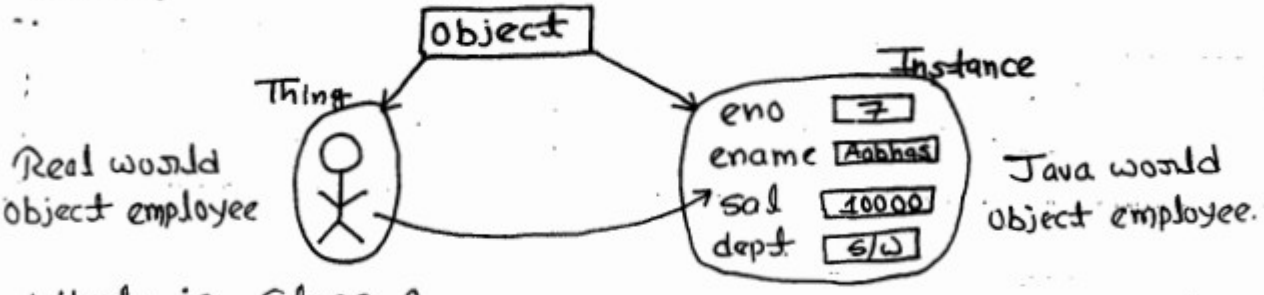
*. We will develop project for performing real world business operations by using computers.
Every business operations will be performed for one particular object. For example:- NaveshIT teaching operation is performed by an object employee for an object student hence, In the project developement we must create real world objects in the programming world. Then we must implement operations and should perform operations for those objects.

*. Hence, the primary task in developing project is creating real world objects in programming world, for this purpose, we must use class and instance.

*. We must use class for creating real world objects in programming world by storing its value which one of different types.

Q. What is an object ?

⇒ Object is a real world thing which is an instance of a class.



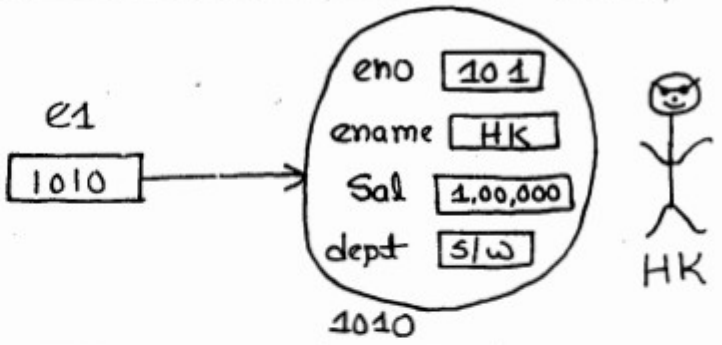
Q. What is Class ?

⇒ • A class is a user defined datatype. It is a blueprint of an object from which we create multiple object of same type.
• Class is also called as Specification or template, because it specifies what are the operations and properties of an object by creating variables and methods inside the class.

- Class creates only type of objects but not actual object, it means, class creates the type of object Employee, but not the particular object employee HK. Hence, class is also called as Logical construct of an object.
- To create actual object we must create memory from this class and in this memory we must store particular object values, the object which we want to represent in the program.
- Instance is a memory created from a class for representing a particular object physically by storing this object value.
- Below diagram will show you a class creation to represent real world object Employee and instances creation for representing a particular Employee objects by storing their values.

```
// Employee.java
class Employee {
    int eno;
    String ename;
    double Sal;
    String dept;
}
```

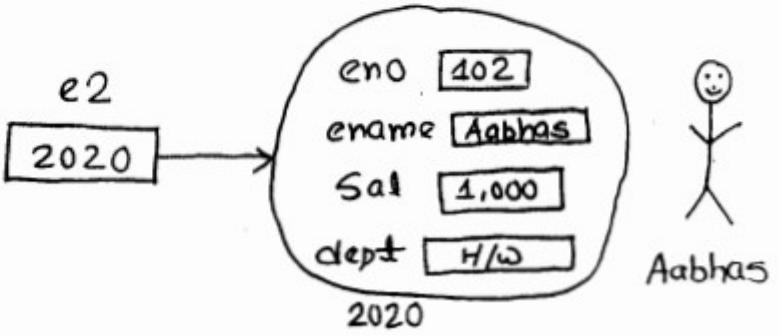
```
Employee e1 = new Employee();
```



We must treat, Employee as a datatype name like byte, int, long.

So using the name Employee we can create variables for storing Employee object references.

```
Employee e2 = new Employee();
```



* In above diagram the keyword class creates datatype to represent real world object of type Employee.
* The keyword new one of the object of type Employee by providing instance.
* Java is called dynamic memory programming language because once we declare a variable inside a class, we can create multiple ~~object~~ copies of memory from the same variable ~~sp~~ declaration by using new keywords as shown ~~below~~ above.
* C is called static memory programming language because from one variable declaration we can create one copy memory.
  If we want another copies of memory then we must declare one more variable.

———————X———————X———————X———————

Projects

Project 1 :-

Below is the complete program for compiling and executing Employee object.

//Employee.java (Data type class)

```
class Employee {
    int sno;
    string ename;
    string department;
    double salary;
}
```

// Company.java (Executable class)

```
class Company {
    public static void main(String[] args) {
    // Creating 1st instance
    Employee e1 = new Employee();
    e1.eno = 101;
    e1.ename = "HK";
    e1.salary = 100000;
    e1.department = "JAVA";
    // Creating 2nd instance
    Employee e2 = new Employee();
    e2.eno = 102;
    e2.ename = "Aabhas";
    e2.salary = 10000;
    e2.department = "HTML";
    // Displaying e1 & e2 object's values
    Sopln("e1's pointing object values");
    Sopln("e1.eno" + e1.eno);
    Sopln("e1.ename" + e1.ename);
    Sopln("e1.salary:" + e1.salary);
    Sopln("e1.department:" + e1.department);
    Sopln();
    Sopln("e2's pointing object values");
    Sopln("e2.eno" + e2.eno);
    Sopln("e2.ename" + e2.ename);
    Sopln("e2.salary" + e2.salary);
    Sopln("e2.department" + e2.department);
    } // main method close.
} // class close.
```

← Here, it is only Instance.

← storing HK values in e1 instance.

→ Here onwards, e1 instance is called Object.

← storing Aabhas value in e2 interface.

Project 2:-

Create a user defined datatype for representing real world object student In java world, Create 2 students Object HK, Balayya with their values and display values.

```
// Student.java
class Student {
    String Name;
    int rollnum;
    String course;
    int fees;
}
```

```
// Institute.java
class Institute {
    public static void main(String [] args) {
    Sopln("Creating 2 student objects");
    Student s1 = new Student();
        s1.Name = "Hk";
        s1.rollnum = 10234;
        s1.course = "Core Java";
        s1.fees = 1500;

    Student s2 = new student();
        s1.Name = "Balayya";
        s1.rollnum = 10235;
        s1.course = "Acting";
        s1.fees = 5000;
```

```
        Sopln("1st Object value");
        Sopln("Name:" + s1.Name);
        Sopln("Roll No.:" + s1.rollnum);
        Sopln("Course:" + s1.course);
        Sopln("Fees:" + s1.fees);

        Sopln();

        Sopln("2nd Object values");
        Sopln("Name:" + s2.Name);
        Sopln("Roll No.:" + s2.rollno);
        Sopln("Course:" + s2.course);
        Sopln("Fees:" + s2.fees);
    } // main method.
} // class.
```

Project #3:-

Create a user-define datatype to represent real word object Bank account in java world and create 2 bank accounts for 2 customers HK and Balayya with their values and display those values.

// Account.java

```
class Account {

    String Accname;
    long Accnum;
    String Address;
    double Balance;
}
```

// Bank.java

```
class Bank {
    Public static void main (string [] args) {
    Sopln("Creating 2 Accounts Holdess Objects");
    Account A1 = new Account();
     A1. Accname = "Hk";
     A1. Accnum = 3205406153 2 ;
     A1. Address = "Hydesabad";
     A1. Balance = 1000000.47;
    Account A2 = new Account();
     A2. Accname = "Balayya";
     A2. Accnum = 3254006152 3;
     A2. Address = "Chennai";
     A2. Balance = 999999999.99;
     Sopln("1st Account holder values");
    Sopln("Name : " + A1. Accname);
    Sopln("Acc. No.: " + A1. Accnum);
    Sopln("Address:" + A1. Address);
    Sopln("Balance:" + A1. Balance);
     Sopln();
     Sopln("2nd Account Holder values");
     Sopln("Name : " + A1. Accname);
     Sopln("Acc. No. : " + A2. Accnum);
     Sopln("Address : " + A2. Address);
     Sopln("Balance : " + A2. Balance);
  } // main method.
} // class.
```

Litecals, Types of literals & Rules & Java 7 new features

1. What is literal and What is use of a literal?
2. Types of literals
3. Explation on :-
   Integer literals, Float-point literal
   characteee litecals, Boolean literals.
   String literals, class literals and
   null literals
4. Identify valid litecals from given list!
5. Types of Integee literals (B, O, D, HD)
6. Identify valid Integee litecals
7. Unicode character set, and special programs
8. Java 7 new featuees in litecals
   1. Binary litecals
   2. Undeescoee in number litecals.
9. Interview questions and OCJP bits

① What is a literal ?

Ex : 10, 10.0, 'a', " a ", true, null etc ... are literals

② How many diff types of literals Java Supports

A) Java Supports 7 types of literals

Integer literals ⟶ Ex : 10, 20, 30 .....

Floating literals ⟶ ex : 10.0, 20.4, 30.5

character literals ⟶ ex: 'a', '1', '@'

conditional literals ⟶ ex : true, false

String literals ⟶ "a", "ab", "1", "123"

Null literals ⟶ null

class literals ⟶ Student.class, Emp.class, String.class, Integer.class System.class

Evecy litecal has default type in Java

① Evecy Integec will be teeated as int type.

Ex: - In a pogeam if we use the value 10, compiler and JVM considec 10 as of int type.

- If we want to treat any Integec number as long, we must suffix L/l to this Number

  Ex: 10 is of int type
  10L is of long type.

- if we want to treat any Integer Number as byte/short we must prefix cost operator

  Ex: 10 is of int type
  10L is of long type
  (byte) 10 is of byte type
  (shoct) 10 is of short type

② Evecy floating-point numbec is teated as double

Ex: 10.0 is of double type

- if we want to treate floating point Number as bloat we must suffix F/f.

Ex : 10.0 ⟶ is of double type
10.0f ⟶ is of float type

float f1 = 10.0 ⟶ CE : possible loss of precision
float f2 = 10.0F ⟶ correct.

③ If we want treat Something as a character, we must place it inside single quote (' ')

- Any Single letter / digit / special character placed inside single quote is treated as a character.
- Every character by default is treated as char type

Ex: 'a' is a char type
'#' is a char type
'1' is a char type

a is not a character, it is just an alphabet considec as a vaciable.

④ - In java we have a special data type called boolean to represent conditional values true and false.

- Like in c, in java we can not use 0 (zero) as false, 1 as true, java compiler will throw error. we must use false as false and true as true.

Ex: boolean b1 = true; ✓ if(true){} ✓
boolean b2 = false; ✓ if(false){} ✓

boolean b3 = 0; } X CE if(0){} } X CE
boolean b4 = 1; } CE if(1){} }

⑤ If we want to treat some value as String type, we must place that value inside double quote (" ").

- Any zero or multiple characters placed inside " " will be treated as String.

- In java every String is treated as java.lang.String type, in java String is a referenced type it is class type.

- Ex: "" } is a String type
" " }
"a" }
"al" }
"Hari" }

Harikrishna → is not a string, it is consider as variable. ~~This class life name, it means classname.class is a literal in java program.~~

⑥ Every class will have .class file after compilation
This .class filename, it means classname.class is a literal in java program.

- It means classname.class is treated as java.lang.ClassType

- For example :-

String.class
Integer.class
Student.class
Employee.class
} is a java.lang.ClassType

⑦ Null :-

- null is a litecal of any refecenced type
- it means it does not have any specific type
- null is any referenced type litral means it may be of array / class / interface / enum / annotation type. It means we can store that literal null in any type of referenced variable.

For example :-

null ⟶ null type can not say

String s = null; ⟶ String type
Integer i = null; ⟶ Integer type
Student s = null; ⟶ Student type
} class

int[] ia = null; ⟶ int array
Employee[] ie = null ⟶ employee accay } accay

Math Type	JavaType	Simple literals
Integer (default → int)	byte	(byte)5, (byte)6, (byte)7
	short	(short)5, (short)6, (short)7.
	int	5, 6, 7, 8 ....
	long	5L, 6L, 7L, 8L ....
Floating-point (default → double)	float	5.0F, 6.0F, 7.0F ....
	double	5.0, 6.3, 7.2, 8.5 .....
characters	char	'a', 'b', '@', '%', '1', '2',
Conditional	boolean	true false
String	java.lang.String	"a", "b", "Hari", "123" "2#2"
	java.lang.Class	Emp.class, Stud.class
	Null type	null

Q) Identify literals Type in below list :-

50	int
50L	long
50F	float
50D	double
(char) 50	char
(byte) 50	byte
(short) 50	short

(boolean) 50	CE

Q) Identify valid literals in the below list ?

A) → Refer volume 1A page No 70

* Write a program to varify and confirm literal type by compiler as shown in above table.

To cheak the literal type we have two ways.

① Assign a literal to boolean Type variable or some other type of variable like String then compiler will throw error by specifying literal type.

② Overload the method with all premitive data types as parameters, call this method by passing all 7 types of literals

progcam :-

```
class Test01_LitralTypeCheaking {
    p.s.v.m.(String[] args) {

// boolean bo = 5; CE: int cant convected into boolean
// boolean bo = 5L; long ——— " ———————
// boolean bo = 5.0; double ——— " ———————
// boolean bo = 5.0F; float ——— " ———————
// boolean bo = 'a'; char ——— " ———————
// String s = true; String ——— " ———————
// boolean bo = String.class; class ——— " ———————
// int i1 = null; <null> is not converted into int.
// int i2 = (String)null; String can't convected into int

  m1(5); m1(5L);
  m1(5.0); m1(5.0F)
  m1(5F); m1(5D);
  m1('a'); m1(true);
  m1("a".); m1(String.class);
  m1((true)5); m1((short)5);

// m1(null); // CE: Ambigious error

  m1.((String)null); m1.((class)null);
/* ① Null do not have any specific type It is
      of any referenced Type.
```

② So m1(null) call is matched with both m1(S) and m1(C) parameter method

③ Hence compiler unable to take decision which parameter method must be executed, therfore we got CE: ambigious error

④ We are specifying type to null to resolve CE: AE i.e. m1.((String) null);

*/

}

```
static void m1(byte b1){
    sopln(" byte-param"); }

static void m1(short s1){
    sopln(" short-param"); }

static void m1(int i1){
    sopln(" int-param"); }

static void m1(float f1){
    sopln(" float-param"); }

static void m1(long l1){
    sopln(" long-pacam");}

static void m1(double d1){
    sopln(" double-param")};

static void m1(char c1){
    sopln(" char-param"); }

static void m1(String s2){
    sopln(" String-param"); }
```

```
static void m1(class c1){
    sopln("class-param");}
```

Rules Using litrals :-

- We can not use litral as a single stmt.
- we can not litecal left side of assignment opecatoc
- We can use litecal only in
    1. variable assignment
    2. in an expression
    3. as method arg
- class Test02_LitealsRules {

```
p.s.v.m.(String[] args){
// 10
// 10;
// 10 = 10;
int m = 10;
int n = m+20;
sopln(10);
}
}
```

⊕) Displaying literals by Using print or println methods:

① When we display Integer, floating point, character, boolean and String literals we dont get any CE, the literal value is directly displayed on console.

② For displaying passed literal, The literal type parameter println method is executed.

③ println method is a overloaded method it has 10 overloaded forms with parameter type - int, long, float, double, char, boolean, String, char[], Object and No parameter

④ Hence based on above point we can conclude we do not have byte and short parameter println method and also we do not have our own class type and ClassType method parameter method, then —

ⓘ if we pass byte or short Type value as arg to println method, int parameter method is executed

ⓘⓘ if we pass our own class object like new A(), new Student(), new Emp(), ... and class Type literals A.class, Student.class, String.class etc. , then Object Parameter println method is executed.

⑤ if we pass null as argument directly to println method we will get Compile time error, Ambigious error, bcoz null is matched with char[] and String param. , hence we can say we can not call println method by passing null directly we will get compile time error.

⑥ To pass null as arg to println method we must specify ~~type to null~~ specific type to null by using either variable assignment or by using cast operator

⑦ if we pass null as char[] type we will get NullPointerException.

⑧ if we want to pass class literal (classname.class) , with this className we must create either class or interface or enum or annotation or we must have pei-defined class / interface / enum / annotation otherwise compiler will ~~throug~~ throw error.

⑨ When we want to use large Integer we must suffix with L/l otherwise compiler will throw error.

below peogeam will show you all above point, in this program identify which parameter println method executed, and also identify CE & RE.

```
class Test03_DisplayingLitrals {
    p.s.v.m (String[] args) {
sopln (10); ← Integer → println (int) → 10
sopln (10L); → long → println (long) → 10
sopln (10.0); → println (double) → 10.0
sopln (10.0F); → println (float) → 10.0
sopln (true); } → println (boolean) → true
sopln (false); }                    → false
sopln ('a'); → println (char) → a
sopln ('1'); ——"—— → 1
sopln ('10'); error
sopln ('ab'); error
sopln (''); error
sopln (' '); → println (char) →
sopln ('  '); error
sopln (" "); }
sopln ("  "); } println (String)
sopln ("a"); → a
sopln ("1"); → 1
```

sopln("ab"); ⟶ ab

sopln("10"); 10

sopln("abc123@bbc") ⟶ abc123@bbc

Sopln(a); → error → a is not a char

Sopln(Hi); → Hi is not a String ⟶ error

int a=5;

sopln(a); ⟶ o/p = 5

String Hi = "Hi";

sopln(Hi); ⟶ o/p → Hi

sopln(5); → println(int) ⟶ 5

sopln('5') → println(char) ⟶ 5

sopln("5") ⟶ println(String) → 5

sopln(A.class) → println(Object) → op ⟶ class A

sopln(B.class) ⟶ —"— op ⟶ interface B

sopln(E.class) → —"— op ⟶ enum E

sopln(Ann.class) → —"— o/p →

sopln(P.class); // error CE We dont have created class with nam P & their is No class predefined like P

sol sopln(String.class) → predefined class
→ op → java.lang.String.

Sopln(null); // error

String s = null;
Sopln(s); → op → null
Sopln((String)null); → op → null

char[] ca = null; → No error
Sopln(ca); → error NPE
Sopln((char[] null); → error → NPE
// NPE → NullPointerException

~~sopln(99889988997707);~~

sopln((int)null);
sopln((String)null);
sopln((A) null);
} println(Object) op → null

sopln(9988998899770707); → error
no. to large
sopln(9988998899770707L); → O/K

int i1 = 9988998899770707;
int i2 = 9988998899770707L;
} CE

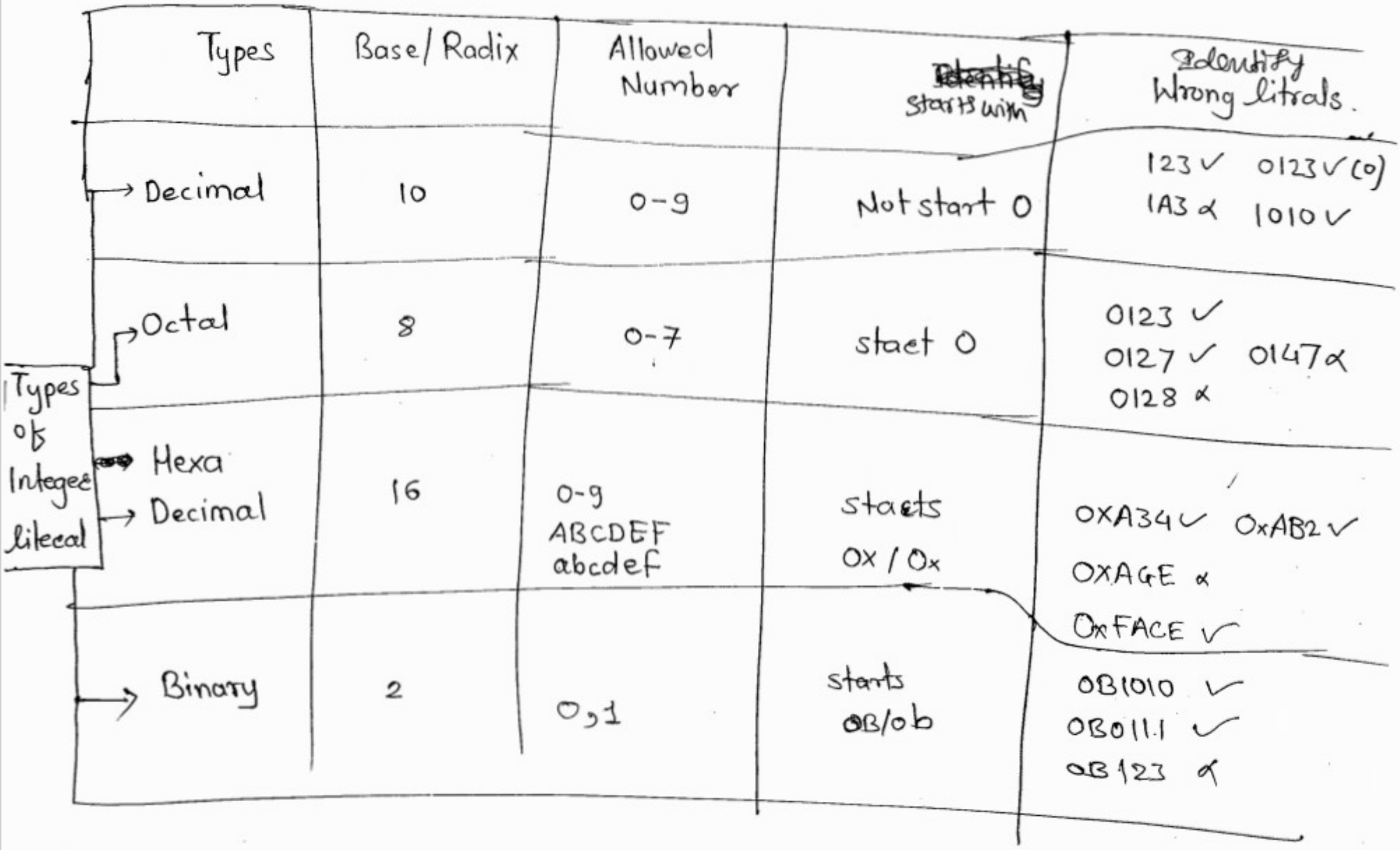
```
long l1 = 998899889988559; → error, L missing
long l2 = ----//---------- L; → ok.

println(int) { system.out.println(0123); → octal Number → 83
               {sopln(1010); → Not binary → 1010
}
}
class A{}
interface B{}
enum E{}
@interface Ann{}
```

int type litrals :-

- As per number system we have four types of Integers (int literals)
- They are
  1. Binary litrals
  2. Octal litrals
  3. Decimal literals
  4. Hexa decimal literals.

Follow below table to find the literal type and to answer interview Que on above int type litrals.


Types of Integer literal
Types
Base/Radix
Allowed Number
Identify starts with
Identify Wrong literals.
Decimal
10
0-9
Not start 0
123 ✓ 0123 ✓ (0)
1A3 ✗ 1010 ✓
Octal
8
0-7
start 0
0123 ✓
0127 ✓ 0147 ✗
0128 ✗
Hexa Decimal
16
0-9 ABCDEF abcdef
starts 0X / 0x
0XA34 ✓ 0xAB2 ✓
0XAGE ✗
0xFACE ✓
Binary
2
0,1
starts 0B/0b
0B1010 ✓
0B0111 ✓
0B123 ✗

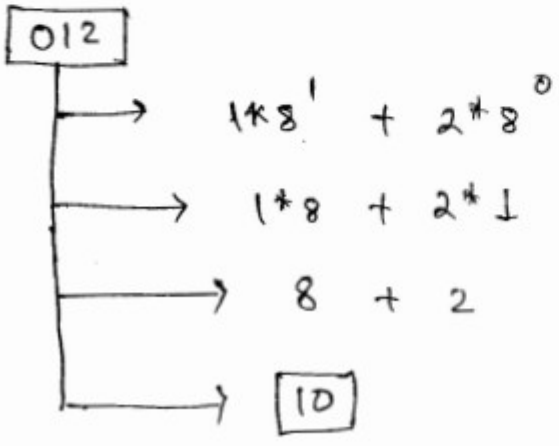
Binary, octal, Hexadecimal number Convecsions

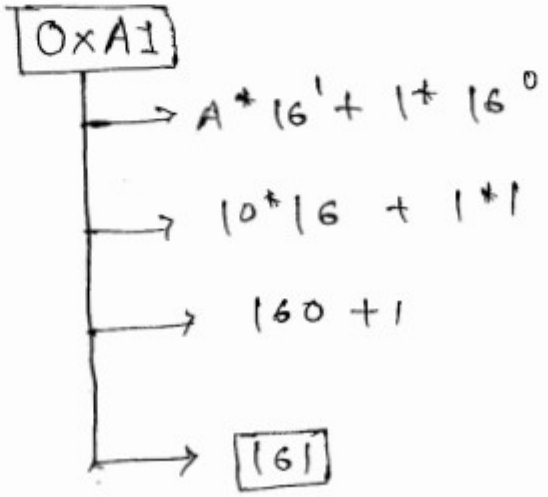
- When we use binacy, octal & Hexa decimal No's in a peogcam compilee will converts all these three Numbees to decimal then this converted decimal Numbee will be use in the program calculation and foc displaying.

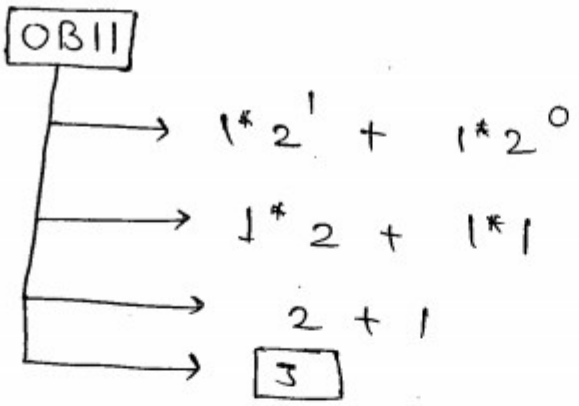
- ex :-

```
sopln(12); ——→ 12
sopln(012); ——→ conveeted ——→ 10
sopln(0XA1); ——converted——→ 161
sopln(0B1010); ——converted——→ 10
sopln(1010); ——————→ 1010
Sopln(0101); ——converted——→ 65
```

- below diageam shows O, HD, B Numbees conversions to decimal.







Case 3 : [redacted], we can not create binary, hexadecimal number as floating-point no. if we try compiler will throw error.

*Identify Compile time error in below literals :-

Decimal	Binary	HD	Octal [Imp]
10 ✓	0B1000	0x123 ✓	0123 ✓
10L ✓	0B1010L ✓	0x123L ✓	0123L ✓
10F ✓	0B1010F ✗	0x123F ✓	0123F ✓
10D ✓	0B1010D ✗	0x123D ✓	0123D ✓
10.0 ✓	0B1010.0 ✗	0x123.00 ✗	0123.0 ✗

✓ 012389F
✗ 012389L

① We can represent decimal Number as int, float, long, double.

② We can represent binary Number as only int and long type

③ If we try to represent it float or double by suffixing F, D or .0 then compiler throw error.

④ We can also represent Hexa decimal number as only int and long type.

→ if we try to represent float or double with suffix .0 or .0F or .0D compiler will throw error.

→ but we dont get CE if we suffix F or D Here F and D are not act as suffix chars. bcoz they are in range of Hexa Decimal Number, they are replaced with 15, 13 respectively.

⑤ Octal literal can be represented as int and long

→ To the octal litral we can suffix F, D and 0.0 we doent get CE but here twist is the octal number is not converted into decimal the octal number is as it is used

as floating point Number means 0123F is converted to 123.0

⑥ if we suffix F, D, .0 to a octal Number, it is not consider as octal Number rather it is consider as floating-pt Number hence with suffix character F, D and .0 in a octal Number we can use the digits 8,9

→ It means 012389F, 012389D, 012389.0 are correct. the result of these Number 12389.0 .

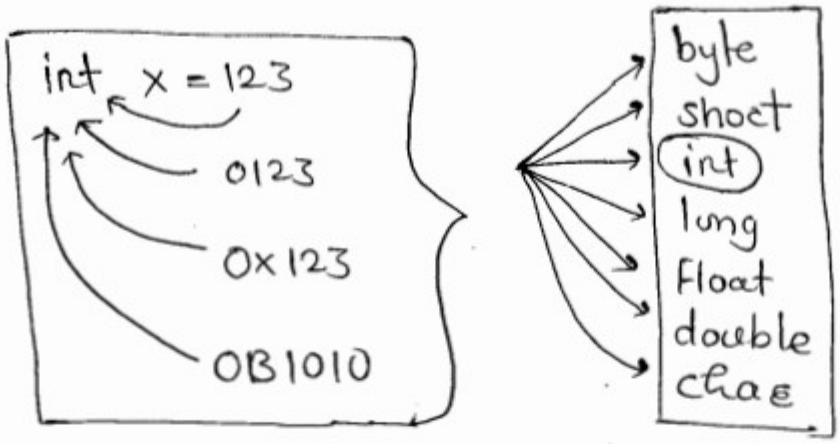
→ But suffix character L we can't use 8,9 in a Octal Number compile throw error bcoz it is consider as octal Numbers.

→ We cannot use L to a floating point Number compiler throw error, but we can convert Floating-point Number to long cast operator.

```
long l1 = 123.01; ✗ CE
long l2 = (long) 123.0 ; ✓
          ↑ cast operator
```

⑦ Binary, Octal and HD Numbers or int type literals then we can assign and store these 3 type of literals to all primitive data type literals except boolean.

→ The default datatyp of these three litrals is int.



→ Here, the rule is the litral assigned to variable must be in the range of variable hence CE.

byte b1 = 0xA; ✓ ⟶ 10 → b1 [10]

byte b2 = 0xA1; x ⟶ 161 →

short s2 = 0xA1; ✓ s2 [161]

int i2 = 0xA1; ✓ i2 [161]

long i2 = 0xA1; ✓ i2 [161]

float f1 = 0xA1; ✓ f1 [161.0]

double d1 = 0xA1; ✓ d2 [161.0]

char c2 = 0xA1; ✓ c2 [161 matching char]

boolean bo2 = 0xA1; ✗

String str2 = 0xA1; ✗

Java 7 New Features in literals :-

- In Java 7 version two new features are added with respective 2 literal

  ① Binary literal.
  ② Underscore in number literal.

① Binary literal :-

① upto Java 1.6 version java doesn't support binary literals.
→ java supports only Octal, Decimal and hexadeci.
→ from java 7 onwards, java supports binary litrals with below two rules
  ⓐ binary number should start with 0B/0b
  ⓑ It should contain digits only 0, 1

Example :- int i1 = 0B1010; ✓ → i1 [10]
int i2 = 0b1012; ∝ CE
int i3 = 1010; ✓ (decimal num) i3 [1010]

② Underscore in number literal :-

① From Java 7 onwards we are allowed to place '_' in a number in between digits. this feature will make a number easily understandable and readble.

② In Real life for seprating digits in Number we use ',' but in java we are not allowed to use ',' CE occured

③ As a Replacement of comma we must use '_'

Ex:
```
double amt1 = 1,00,000; ✗ CE
double amt2 = 100000;   ✓
double amt3 = 1_00_000; ✓
```

Note: When we display a Number with '_', underscore is not displayed directly Number displayed

```
sopln(amt3);  → 1_00_000.0 ✗
              → 1,00,000.0 ✗
              → 100000.0   ✓
```

Rules :-

① Underscore must be use in between digit. We should not use at begin or end of number
CE → illigal Underscore.

```
double d1 = 1_00_000;  ✓
double d2 = -100000;   ✗ CE
double d3 = 100000_;   ✗ CE
double d4 = 10_____000; ✓
```

② We can use multiple underscroe in betⁿ digit.

(2) Underscore is not allowed immediaty before and after dot in floating point number.

double d5 = 1_00_000.65 ✓

double d6 = 1_00_000._65; ✓

double d7 = 1_000_00._65; ✗

double d8 = 1_00_000_.65; ✗

(3) Undersocre is not allowed immediatly before and after suffix character L, F, and D

double dg = 100000_L; ✗

double d10 = 100000L_; ✗

double d11 = 100000_F; ✗

double d12 = 1_00_000L; ✓

(4) Underscore is not allowed immediatly before, after and betn prefix characters \0, 0x, 0B,

double d13 = -0xA1; ✗

double d14 = 0_xA1; ✗

double d15 = 0x_A1; ✗

double d16 = 0xA_1; ✓

double d17 = 0xA_B; ✓

double d18 = 0_B1010; ✗

double d19 = 0B10_10; ✓

double d20 = _0123 ✗

double d21 = 0_123 ✓ → 0123; → 83

Unicode characters set :-

① Unicode is stands for universal characters code.

② It means unicode character set will contains all countries speaking languages character codes

③ Unicode characetes set will support 65536 characters (i.e. 0- 65535).

④ So to store unicode characters 2 bytes of memory should be allocated.

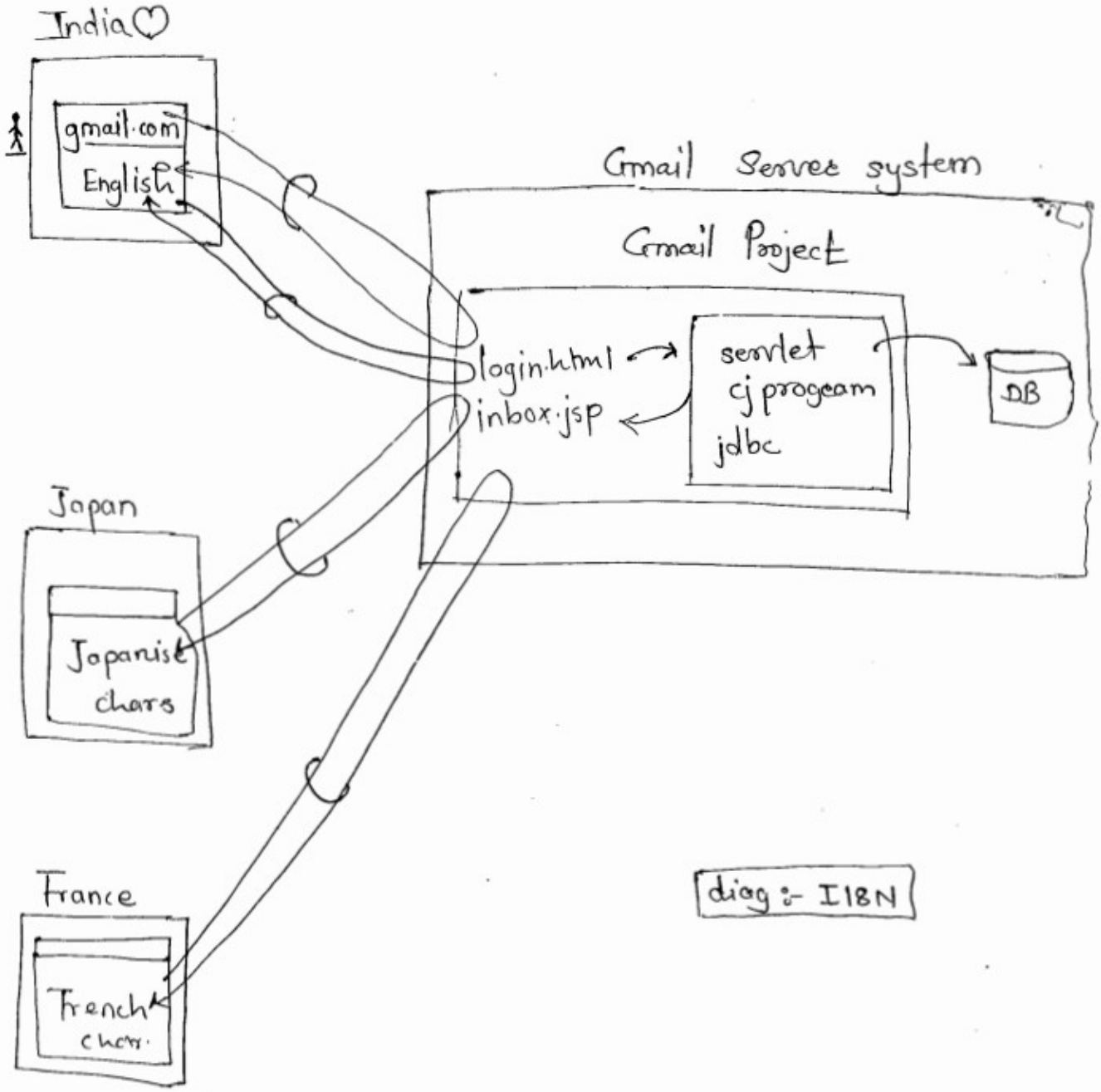
⑤ Java takes 2 bytes for char type data variable. because java wants to support unicode character set

⑥ Unicode character set is used for developing (I18N) Internatization application.

⑦ I18N application means it will display content characters on browser to end-user in that country specific language

⑧ The languages those support web applications development must support unicode character set, because web appl^{n} is an I18N application.

- below diagram shows The meaning of I18N.



diag :- I18N

(9) Unicode is a hexa decimal int type number. so in a unicode number the allowed digits are 0-9A-F

(10) Unicode number has a special format that is it starts with \u and ends four digits.

ie. :→ \uxxxx ← 4 digits in the range 0-9AF

(10) A unicode number can be represented as

(1) as number → use directly as \uxxxx.

(2) as character → use it in ' ' as '\uxxxx'

(3) as string ⟶ use it in " " as "\uxxxx"

(11) Like binary, octal, hexa decimal numbers, unicode number will also converted into its equivalent decimal number.

(1) For example :- int i1 = \u0031;

⇒ int i1 = 1;

(2) Unicode literal as character :-

int i1 = '\u0031';

⇒ int i1 = '1';

⇒ int i1 = 49;

i1 | 49 |

char ch1 = '\u0031'; ch | '1' |

(3) Unicode literal as String :-

int i1 = "\u0031"; ✗ CE

String s1 = "\u0031"; s1 | "1" |

(12) Below table shows unicode Number and its corresponding value in a progcam :-

Unicode Num	Corresponding value in program.
\u0030	0
\u0031	1
\u0041	A
\u0042	B
\u0061	a
\u0062	b

below table shows unicode literal with its coumrosponding ASCII char and ascci Number.

Unicode	ASCII char	ASCII Numb.
'\u0030'	'0'	48
~~'u0031'~~ '\u0031'	'1'	49
~~'u0041'~~ '\u0041'	'A'	65
'\u0042'	'B'	66
\u0061'	'a'	97
'\u0062'	'b'	98

Q) A unicode literal can only form one digit, then how can we prepare a number using unicode literal?

Ans) place equivalent unicode literals on after one to form a Number.

ex:- prepare a Number 123 we must place unicode literals like below.

```
int i1 = \u0031\u0032\u0033;     i2 [123]
⇒ int i2 = 123;
```

Q) prepare a string "abc" by using unicode literals

```
String s1 = "\u0061\u0062\u0063";
s1 = "abc";
```

Rules :-

① A unicode Number that generates a letter 'A' or 'a', 'B' or 'b' .... We must use this unicode Number either in single quote or in double quote else compiler throw error saying variable not found.

ex: sopln (\u0031); ⟶ sopln(1); ⟶ 1 ✓

sopln (\u0041); ⟶ sopln(A); ⟶ CE: cfs var A

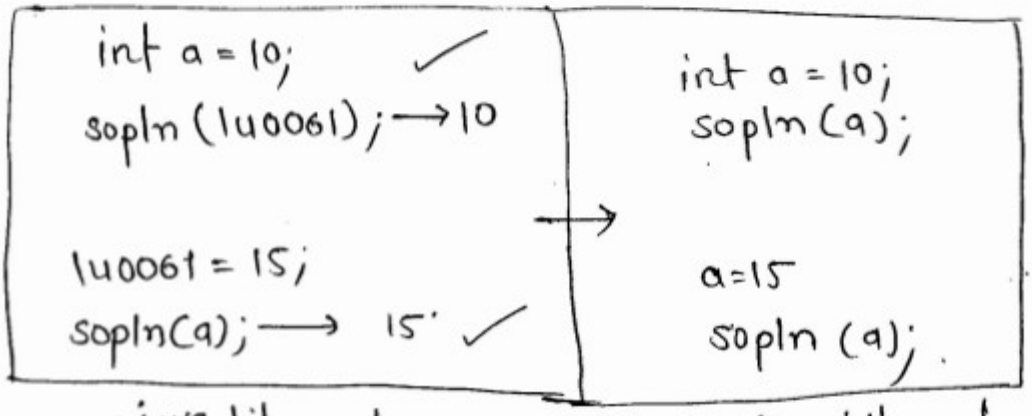
sopln('\u0041'); ⟶ sopln('A'); ⟶ A

sopln(\u0041\u0041); ⟶ sopln(AA); ⟶ CE ✗

sopln("\u0041\u0041"); ⟶ sopln("AA") ⟶ AA ✓

② If we want to use a as a variable Name, we can now use \u0061 directly without sigle Quote or double quote.

ex:- sopln(\u0061); ⟶ CE



· java file code

· class file code changed by compiler.

ch\u0061r\u0063 = '\u0062';

char c = 'b';

- As you can see in above statement you can use unicode literal inside a program anywhere, but you must carefully with correctly, if you are talented you can complete java program using unicode litral.

ex:-

A.java

```
\u0063\u0061ss \u0041{
    public static void m\u0061in (String[]\u0061rgs) {
        Syst\u0065m .out.println ("H\u0061i");
    }
}
```

A.class

```
class A {
    public s. v. main (string[] args) {
        s.o.p.ln ("Hai"); } }.
```

③ The unicode litecal \u0041 to [redacted] is A and the next unicode litecal \u0061 is a, then the Only 20 Numbes diff is there, How 26 letters are represented.

→ beause unicode follows Hexadecimal digits it uses 0-9 A-F digits. to represent missing six digits betn 41-61 unicode litral useuse A-F as after 49 it uses 4A to 4F and then 50-5A onWords [redacted]

Q) What is the Rule on Unicode literal ?

Ans → It should starts with \u and followed by 4 digits 0-9 and A-F

int p = \u00[redacted]; illegal Unicode escapse

Rule :- you can't place wrong unicode literal ~~in~~, even inside a comment. compiler will keep throwing error until we remove this wrong literal or correct this literal.

- comment above literal to test this point
- What is the o/p from this program.

```
- class Example {
        public s.v.m. (String[] args) {
    // \u000d System.out.println ("Hi");
  }}
```

ophions :- ① No output
② CE
③ Hi
④ RE

\u000d ⟶ Next line

// is scope is single line.

Explanation :- Here \u000d is unicode literal represents new line character Hence the scope of single line ~~character~~ comment is an end. then sopln stmt is available in nextline hence it is compiled & executed Hi display

- We can use unicode literal in preparing octal, Hexa Decimal, Decimal and binary Numbers.

Q→ Identify the type of the litral in below stmts :-

op

int i3 = \u0031\u0032; sopln(i3); ⟶ 12

int i4 = \u0030\u0031; sopln(i4); ⟶ 01 ⟶ 1

int i5 = \u0030\u0038; sopln(i5); ⟶ 0A ⟶ CE

int i6 = \u0030\u0031\u0032 sopln(i6); ⟶ 012 ⟶ 10

int i7 = ~~\u00~~ 0x\u0031\u0032; sopln(i7); ⟶ 18

int i8 = 0x\u0041\u0031; sopln(i8);

int i9 = 0B\u0031\u0032; sopln(i9); ⟶ CE

int i10 = 0B\u0031\u0030; sopln(i10); ⟶ 2

→ We can suffix char to unicode literal.

sopln(\u0031L); 1

sopln(\u0031D); 1.0

sopln(\u0031F); 1.0

sopln(\u0031.0); 1.0

but we cannot suffix the above characters to o unicode litrals that represents A letter or special characters.

```
sopln (1400 41L);  ──→ AL
sopln (1400 41D);  ──→ AF
sopln (1400 41f);  ──→ AD
suplⁿ (1400 41.0); ──→ ')' expected.
```
(CE)

→ place " " and place char inside " " then no error.

* Expression evaluation procedure by compiler s/w :-

① compiler evaluate expr sometimes with type and sometimes with values based on the type of the expr.

② we have two types of expressions:

ⓐ variables expression:

ⓑ Constants expression.

③ if an expression contain at list one variable as oprand then that expr. is called variables expr.

→ compiler evaluates variables expr. by subtituing variables type, generates result type, if the result type range lesser than or equals to destination type variable range, program is compiled else compiler throw error.

i.e →
```
byte b1 = 10;   ✓
byte b2 = 20;   ✓
byte b3 = b1 + b2;
```
byte + byte → int

④ An expr. that contains only literals or final variables with assigenments or both then that expr is called constant expr.

→ compiler evaluates constant expr. directly using values used in the expr and by replacing final variables with their value, generates result, then it verifys is this result range within the range of destination variable type or not, if no compiler throw error else compiles fine.

i.e ① byte b3 = 10 + 20 ✓

= 30

② byte b3 = 110 + 20

= 130 ← Not in the range.

* final variable's evaluation has two cases :-

① final variables declaration with initialization :-

② final variables declaration without initialization :-

① final byte b1 = 10;
final byte b2 = 20;
byte b3 = b1 + b2
10 20
30

② final byte b1;
final byte b2;

b1 = 10;
b2 = 20;
byte b3 = b1 + b2
byte byte
int CE

imp

① if final variable is declared with value, inside expr variable name is replaced by value directly, expr is evaluated with values

imp

② if final variables are not initialized in declaration, in expr variable Names are replaced by their type, expr is evaluated with types as shown above.

```
byte b4 = 10L+20      ✗
byte b4 = (byte) 10L+20;   ✓
byte b4 = (int) 10L+20;    ✓
byte b4 = (int) (10L+20);  ✓

Long l1 = 10;          ✓
byte b5 = l1+20;       ✗
byte b5 = (byte) l1+20;  ✗
byte b5 = (int) l1+20;   ✗
byte b5 = (byte) (l1+20);  ✓
```

Note :- The above ~~exp~~ compile time errors in expr we will get only when we perform assingments operations. if we pass above expr to sopln no ce, Result is displayed.

i.e:

```
sopln(10+20);  → 30  pln(int);
Sopln(10L+20); → 30  pln(long);
byte b1=10;
byte b2=20;    soplu(b1+b2);
```

Reading RunTime Values from Keyboard :-

→ In this chap. we will see reading input values from keyboard.
→ If we specify values directly in the program, this program will execute only with those specified values.
→ If we want to change values and want to execute this program with new values, it is not possible automatically without modifying code.

for example :-

Addition.java

```
class Addition {
public static void main(String[] args) {
② → int a = 10;        a [10] → ③
⑨ → int b = 20;        b [20] → ⑤
⑥ → int c = a+b;       c [30] → ⑦
⑧ System.Out.Print("Result" + c);
 }
}
```

Compilation & Execution

> javac Addition.java
> java Addition ①

Result 30 → Execution ⑩

→ If we run above program any number times it will execute only with specified values '10 & 20', if we want to work with new values we have to do changes inside source code. then compile again and then should execute again with modified values.

→ As a programmer we can modify source code but as a customer cannot do changes in source code coz he/she dont know java, even if he know java it is not good to modifying source code everytime for new values.

→ The good programming practice we must develop program only once, it should accept values at execution times with the user given input values, its logic should be executed.

Q What is the diff between hard coded (static) & dynamic application?

Ans If we develope a program by specifying values in it, it is called hard coded or static application.
for ex:- above addition class is a hard coded application, coz values 10 & 20 are directly specified in it.

If we develope an application without specifying values in it, rather we are reading the input values from keyboard at execution time is called dynamic application.

In projects we must develope dynamic application for making program to be executed for any given input. for ex:- Gmail.com, paytm.com, hdfc bank.com are dynamic application coz they will work with user given inputs.

In this chapter we will learn, reading input values from keyboard using java language concepts.
In java we can read input values in 6 ways :-

1. From command line.
2. Using BufferedReader class
3. Using scanner class
4. Using Console class.
5. Using AWT, String GUI
6. Using System properties.

By using above 6 approaches we must read value from keyboard, should substitute those values to variable assignment place as shown below.

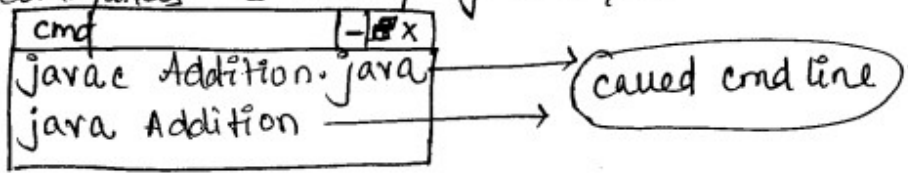
```
Class Addition {
   public static void main(String[]args) {
      int a = ----
      int b = ----
      int c = a+b;
   System.out.println("Result" +c);
   }
}
```

```
Java Addition
Enter F.NO = 10
Enter S.NO = 20
Result : 30
```

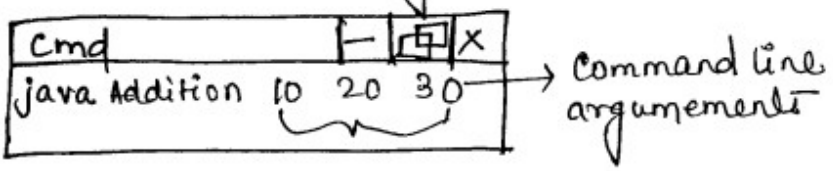
All above six approaches are having their own advantages and limitation, we must choose the particular approach as per our project needs.

Reading Values from Command line :-

(1) What is command line :- In command prompt the line at which we are running commands javac and java commands is called command line.

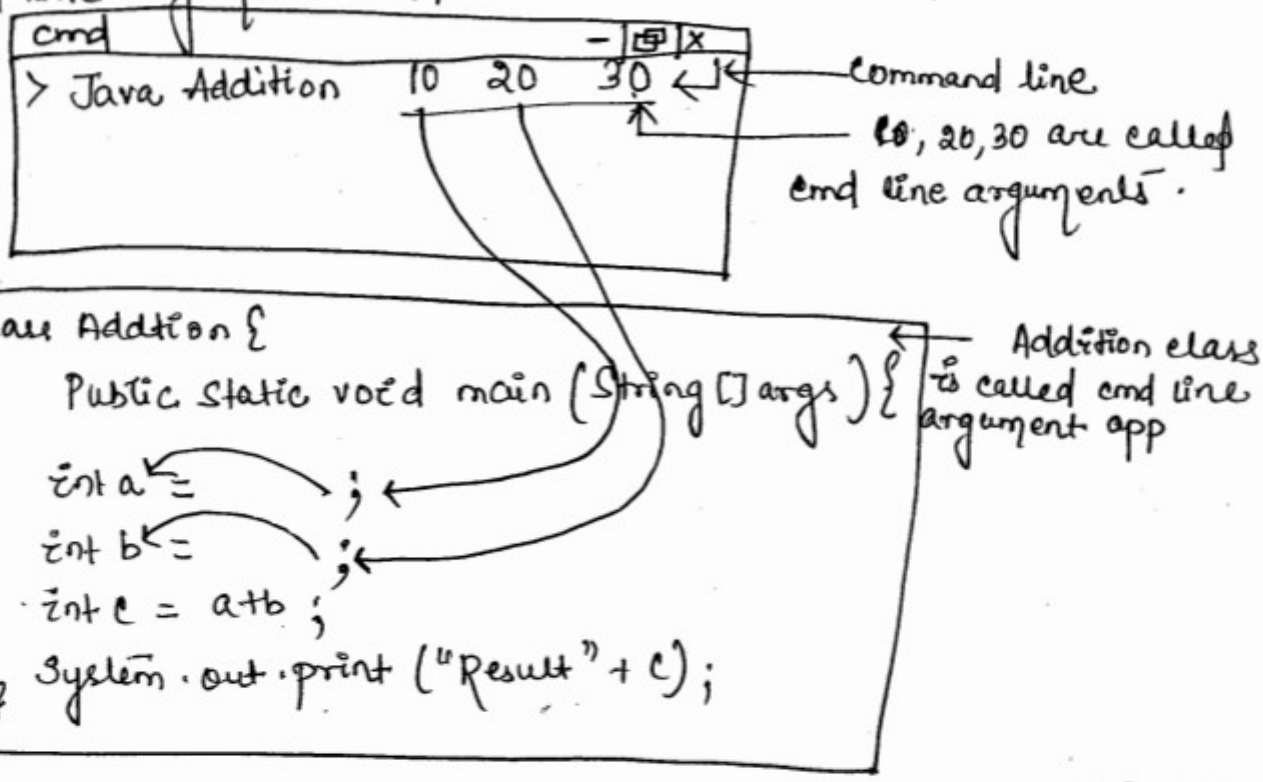


(2) What is cmd line arguments :- The input values we are sending from cmd line at program execution time are called cmd line arguments.



Rule :- We must separate arguments with space

Q What is cmd line argument appl° :- A program that reads input values from command line is called cmd line arguments app.



Q How JVM will pass the command line arguments into Java program?

A At String [] object as argument to main method. With the specified values JVM internally creates String [] object, passes this object represnce as argument to the main method then this object reference is stored in main method parameter args variable.

Q How can we read cmd line arguments inside java program?

A Using main method parameter name "args" as args [0], args [1], args [2] ....

args [0] returns first values.
args [1] returns second values.
arg [2] returns third values.

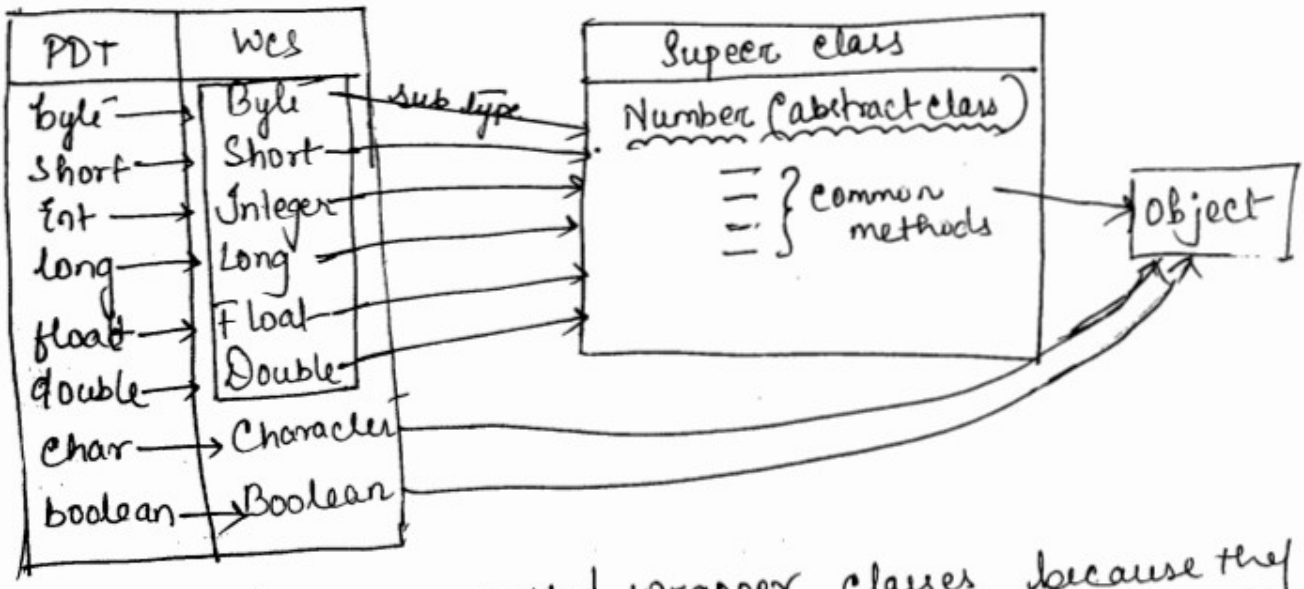
Q The number we are sending from keyboard will passed into program as String.
Ex:- "10". 10 will be passed as "10" which is string.
then how can we convert- string number "10" to its actual type no 10 ?

A We must use wrapper class method parseInt(-), parseDouble(-), parseBoolean(-)... methods

Q What is a wrapper class?
A? A class given by Sun for representing ~~primitive~~ primitive type value as object (reference type) and performs different operation on this primitive value on this object is called wrapper class.

For defining set of methods and to operate each primitive value seperatly 'sun' has defined 8 wrapper classes 1 for each primitive type seperatly

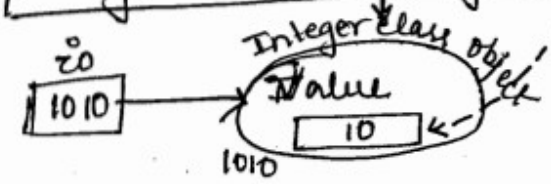
Below table shows 8 primitive types & their wrapper classes:-



→ Above 8 classes are called wrapper classes because they will wrap primitive value as object, allows us to perform diff operations on this data.

Wrapping means storing single type primitive value in its object memory and performing diff. operations on it.

for ex :- `Integer io = new Integer(10);`



→ In above diagram the primitive value 10 is wrapped inside integer class object.

→ There is no serious reason behind the name "wrapper", since all these 8 classes must refer as a group by one name, so the name wrapper choosen because these classes covering / wrapping single primitive value.

→ Except in Character class remaining all seven wrapper classes, we have a method for converting a number from string to primitive data type value.

Method is `public static xxx parseXxx(String value)`

(PDT name)

for ex :- In Integer class it is defined :-

`public static int parseInt(String value)`

In Double class :- `public static double parseDouble(String value)`

In Character class :- `public static char parseChar(String value)`

Hello Gajni, character class doesnot have parseXxx() method.

In Boolean class :- public static Boolean parseBoolean(String value)

Coding:- Converting "5" to ints

```
int x = Integer.parseInt("5");
```
(5 → Integer.parseInt, "5" → Integer.parseInt) → x [5]

Converting "5.4" to double 5.4

```
double a = Double.parseDouble("5.4");
```
("5.4" → Double.parseDouble, 5.4 → a) → x [5.4]

Q How can we convert a character from String to char form. as we donot have parse char method?

A We must use string class method charAt() as shown below:

```
String s1 = "a";
char ch = s1.charAt(0);
```
("a" → s1, "a" → ch)

"a" → String object

'a' → char variable

Q How can we convert cmd line argument value from string form to its actual primitive type?

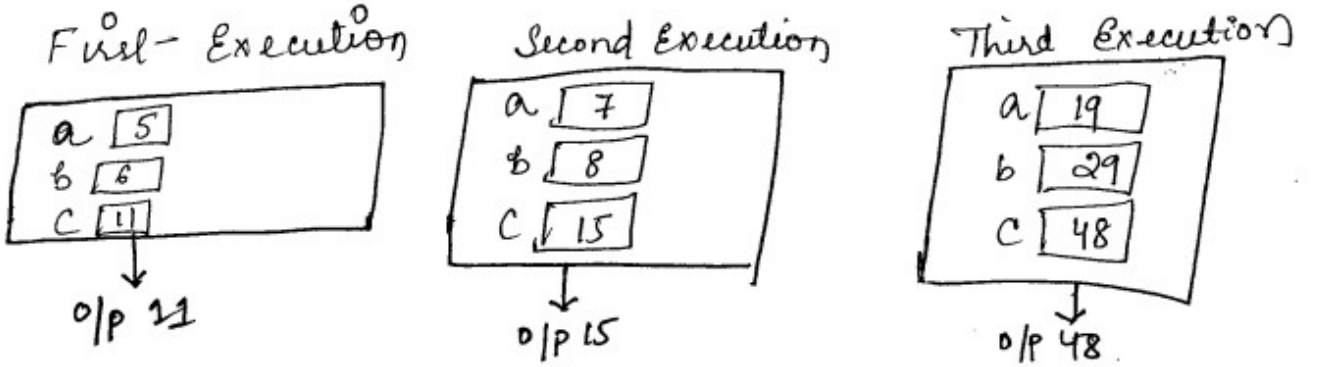
A We must use wrapper class parseXXX method as shown below.

~~> java Addition 10 20.0 a # 7279 HK 10,000 M true~~

eg By using wrapper class.

```
int i = Int.parseInt(args[0]);
double d = Double.parseDouble(args[1]);
...
```

1st Box :- Every time when we are running addition class internally, new memory is created for variables a, b, c with given values and generated result-



After execution completed in every turn, memory created for 3 variables is destroyed.

Rules in executing cmd line program :- [Self or Written type]

~~Java Addition~~

Rule 1 :- ① While must input values to cmd lines program minimum equal to number of values we are reading inside program.

② If we pass less values we will get exception AIOBE

③ If we pass equal to or more than number values we are reading, we won't get any exception, program will read required value, complete execution. The more value you are passing are not used.

④ When we run cmd line program without giving input values, internally jvm creates empty string [] objects passes its reference to main method

⑤ If we run cmd line program by passing input values, internally jvm creates string [] object with given values & passes its reference to main method.

Write a program to read two integer values from command line, add those two int numbers, display output.

Execution pattern of this program

```
>javac Addition.java
>java Addition 10 20
       (class name)  ↓  ↓
                  args[0] args[1]
```

Addition program working with command line arguments.

```
Class Addition {
    public static void main(String[] args){
    int a = Integer.parseInt(args[0]);
    int b = Integer.parseInt(args[1]);
    int c = a+b;
    System.out.println("Result:"+c);
}
}
```

a [5]
b [6]
c [11]

Compilation and execution :-

```
>javac c.Addition.java
>java Addition 5 6
Result: 11
```

"5"
"6"

Execution cases : ① Above program is dynamic application coz it is executing with the values given at execution time

② Without modifying its code, recompiling we can change values, will work with newly given values automatically.

③ Run above program multiple times by given diff. values

```
java Addition 7 8
Result: 15
```

,

```
java Addition 19 29
Result: 48
```

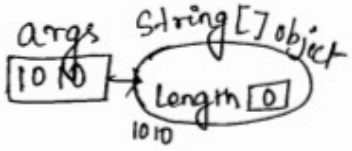
If we run cmd line program without passing inputs, what is the output in below two lines?

	Sopln(args)	Sopln(args[0])
① AIOBE		✓ (No value
② Null		
③ String[] obj reference	✓ (empty obj)	
④ Non of above		

Case-1

java Addition ↵

Exception in thread "main"
AIOBE: 0

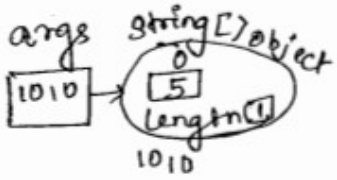


int a = I.PI(args[0]);
XRE (AIOBE:0)

Case2

java Addition 5 ↵

Exception in thread
"main" AIOBE: 1

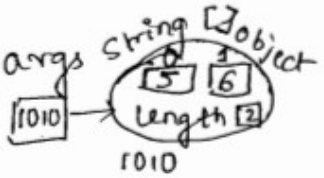


int a = I.PI(args[0]); a 5
int b = I.PI(args[1]);
XRE (AIOBE:1)

Case3

java Addition 5 6 ↵

Result: 11

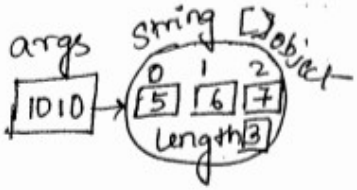


int a = I.PI(args[0]); a 5
int b = I.PI(args[1]); b 6
int c = a+b; c 11
S.O.Pln("Result:"+c);

Case 4

java Addition 5 6 7 ↵

Result: 11



int a = I.PI(args[0]); a 5
int b = I.PI(args[1]); b 6
int c = a+b;
S.O.Pln("Result:"+c)

Rule2:-

① We must pass only the expected type value from command line based on the parseXXX method we used.

for example:- for parseInt method we must pass only Int type value (Integer)

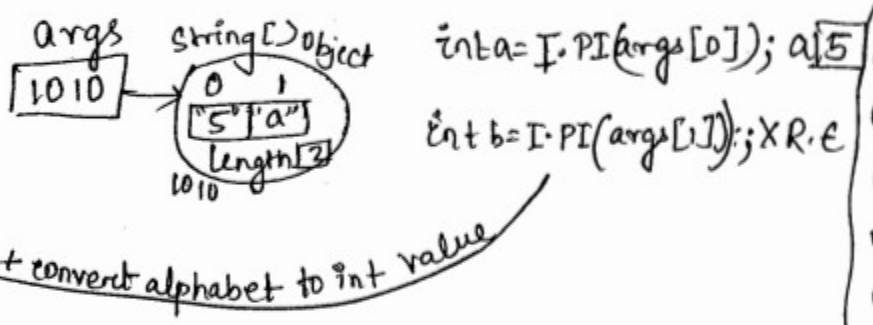
② If we pass other than Integer value, parseInt method will throw NumberFormat Exception. This point is applicable others parse methods also.

for example:-

Case5:- Diff. type of value.

> javaAddition 5 a ↵

Exception in thread "main" NFE: "a"



> javaAddition 5 6.7 ↵

Exception in thread "main" NFE: "6.7"

"6.7" is not integer number, so parseInt method throw "NFE".

In c it is possible to assign float number to int variable, In java it is not possible coz java is strictly typed programming language.

Limitation of above program:- Above program can add only 2 numbers. It is still static program coz it cannot work with diff. no of values.

A program to read all values at cmd line, as many values as sending by user.

```
Class ReadAllValues {
    public static void main (String[] args ){
        for (int i=0 ; i < args.length; i++) {
            System.out.println("values" + i + args[i]);
            System.out.println ("values" +(i+1) + "i" +args[i]);
        }
    }
}
```

Above program can read as many values as we are entering at command line.

for example:-

Execution cases

Case 1:-

>java ReadAllValues ↵ (No value passed)

No O/p (No value displayed)

Case 2:-

>java ReadAllValues 10 ↵ (One value entered)

O/p : Value 1 : 10

Case 3:-

>java ReadAllValues 10 20 ↵ (two values entered)

O/p : Value 1 : 10
Value 2 : 20

Case 4:-

```
java ReadAllvalues 10 20 a ↵
```

3 Different values entered

```
o/p: Value 1 : 10
     Value 2 : 20
     Value 3 : a
```

Case 5:-

```
java ReadAllvalues 102030 ↵
```

```
o/p: Value 1 : 102030
```

Case 6:-

```
java ReadAllvalues * ↵
```

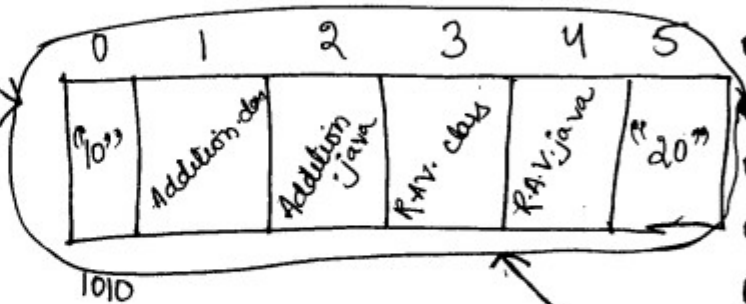
```
o/p: Value 1 : Addition.class
     Value 2 : Addition.java
     Value 3 : ReadAllValues.class
     Value 4 : ReadAllvalues.java
```

Q Why file names are passed here as "*"?
A: "*" means all files names in current directory. When we enter "*" as cmd line value jv will read all files names available in current directory, stores them in the String [] object, then passes this String [] object as argument to main method.

Case 7:-

```
java ReadAllvalues 10 * 20 ↵
```

```
o/p: Value 1 : 10
     Value 2 : Addition.class
     Value 3 : Addition.java
     Value 4 : ReadAllvalues.class
     Value 5 : ReadAllValues.java
     Value 6 : 20
```




```
Class R.A.V {
    P.S.V. main (String [] args) {
        args 1010
    }
}
```

Case 8:-

```
java ReadAllValues \* ↵
```

All file names & directory name are displayed from current drive (B: in our appl" phase case)

The meaning of star "*" is all files from current directory.
"*" means all files from current drive

Case 9:-

```
java ReadAllvalues & ↵
[No output-]
```

Here and (&) meaning sending one process cmd name as input.
Since we have not specified any cmd name no output- is displayed.

Case 10:-

```
java ReadAllvalues &notepad ↵
```

o/p: notepad will open.

Case 11:-

```
java ReadAllvalues ^ ↵
```

o/p: more? _

Cap (^) meaning next line, if you want to enter cmd line value in seperate line at end we must use ^

Case 12:-

```
java ReadAllValues ^ ↵
more? 1 2 3 ^
more? 4 ^
more? 5
Value 1: 1
Value 2: 2
Value 3: 3
Value 4: 4
Value 5: 5
```

Summary :- In cmd line arguments *, *, &, ^ are having special meaning refer above cases.

Q What is the o/p from below program :-

```
Class Test {
Public Static void main (String [] args ) {
for (int i=0 ; i <= args.length ; i++) {
System.Out.Println (args[i]) ;
}
}
}
```

Case 1 :-

javac ReadAllValues TC.java

java Test

O/p - A10BE:0

Exception coz we do not have 0 index value in this array, closely look at loop condition we have used "<=" then, here length index location will be accessed which is never available in array.

Case 2 :-

Java Test 10 ↵

O/p :- 10

Exception : A10BE : 1

In array we donot have 1 index location so we get error.

Iteration 1 : i=0 → 0 <= 1 → args[0] → 10

i=0 → 1 <= 1 → args[1] → A10BE

```
java Test 10 20 ↵
```

o/p: 10
20
AIOBE: 2

Rule 3:- (Rule 1 & 2 are available in previous pages, when we develop for loop from reading values from array, we must use condition operator only '<', should not use '<=' . If you want to use <= , we must use right value args.length - 1 , then we don't get AIOBE.)

```
Class ReadAllValues_TC2 {
  public static void main(String[]args){
  for(int i=0 ; i<= args.length ; i++){
      System.out.println(args[0]);
  }
  }
}
```

→ Only first value will be return always.

Case 1 :- `javac ReadAllValues_TC2.java`

o/p AIOBE: 0

Case 2:-

`javac ReadAllValues_TC2.java 10` →

i=0 → 0<=1 → args[0] → 10
i=0 → 1<=1 → args[0] → 10
i=0 → 2<=1 ✗

o/p Values : 10
Values : 10

Case 3:-

`java c ReadAllValues_TC2.java 10 1020`

o/p: 10
10
10

Problem in above Program ReadAllValues:-

1. In this program we can read any type value
2. If we want to accept only Integer, using above program code we cannot stop user enter other then int values.
3. In above code we must use parseInt method to identify given value int type or not
4. If given input is not intvalue, parseInt method will through except?.
5. Rewrite above program for reading and displaying only int values, if user entered other than int values terminate program.

```
Class ReadOnlyIntValues {
public static void main (String [] args) {
 for (int i=0; i<args.length ; i++) {
 int num= Integer.parseInt (args[i]);
  System.out.println ("Value" + (i+1)+ "i" + num);
 }
}
}
```

Converting given value to int to check its int or not

```
java ReadOnlyInt Values
```

O/p → No

```
java ReadOnlyInt Values 10 20
```

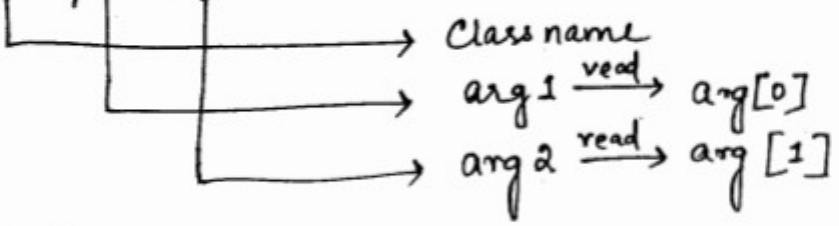
Value 1: 10
Value 2: 20

```
java ReadOnlyInt Values 10 20 a
```

Value 1: 10
Value 2: 20
Exception: NFE

Q Identify how many arguments are passing to class in the below command?

>java Hi Hello HRU?



Options:-

① 3 ✗
② 2 ✓
③ 1
④ None

In above command Hi
Hi is class name
Hello & HRU? are arguments to class Hi
So Ans is 2 arguments we are passing.

Problems of cmd line argument program ?

It has 3 problems

Problem 1:- In the 1st attempt of running program enduser will not get information about how many values, what type of values and in which order values should pass. If we commit mistake JVM through exception. AIOBE or NFE.

Problem 2:- It is very tough to remeber the order of passing values, if we pass values in wrong order eighter we will get wrong result or NFE.

Problem 3:- We cannot read values in middle of program execution

Solution:- To solve all above problems, we must use BufferedReader class given by sun.

② Using BufferedReader class we can read values in middle of program execution by specifying the order and type of value should enter.

Ex:- cmd Approach

```
>java EmpInfo 7279 HK 10,000 ↵
```

reading values → args[0] args[1] args[2]

BR Approach

```
>java EmpInfo ↵
Enter eno : 7279 ↵
Enter ename : HK ↵
Enter Sal : 10000 ↵
```

→ These messages are displaying from the class EmpInfo

→ Input values entering by end user based on msgs displaying

reading values → br.readline();

Note:- BufferedReader class also will return data as string type, we must convert ~~String into~~ given data into there primitive form by using wrapper classes. as we did in cmd App.

Comparision code:-

CmdApp

```
int x = Integer.parseInt(args[0]);
int y = Integer.parseInt(args[1]);
```

BR App

```
int x = Integer.parseInt(Br.readline());
int y = Integer.parseInt(Br.readline());
```

Working with BufferedReader class:-

1. BufferedReader is a pre-defined class given by sun in java.io package. It is used for reading values one line at a time either from keyboard or from file.
2. It has a method called 'readline' for reading one line at a time.
3. It will return value from the line as a string type.
4. Procedure to read data from keyboard using br class

Step(1) :- Import java.io package.

Step(2) :- Create BR class object - connecting to keyboard.

Step 3 :- Call br.readline() method repeatedly as many values as we wanted to read.

Step 4 :- We must handle io exception for br.readline method call sake.

Below program shows reading two int values from keyboard using BR class.

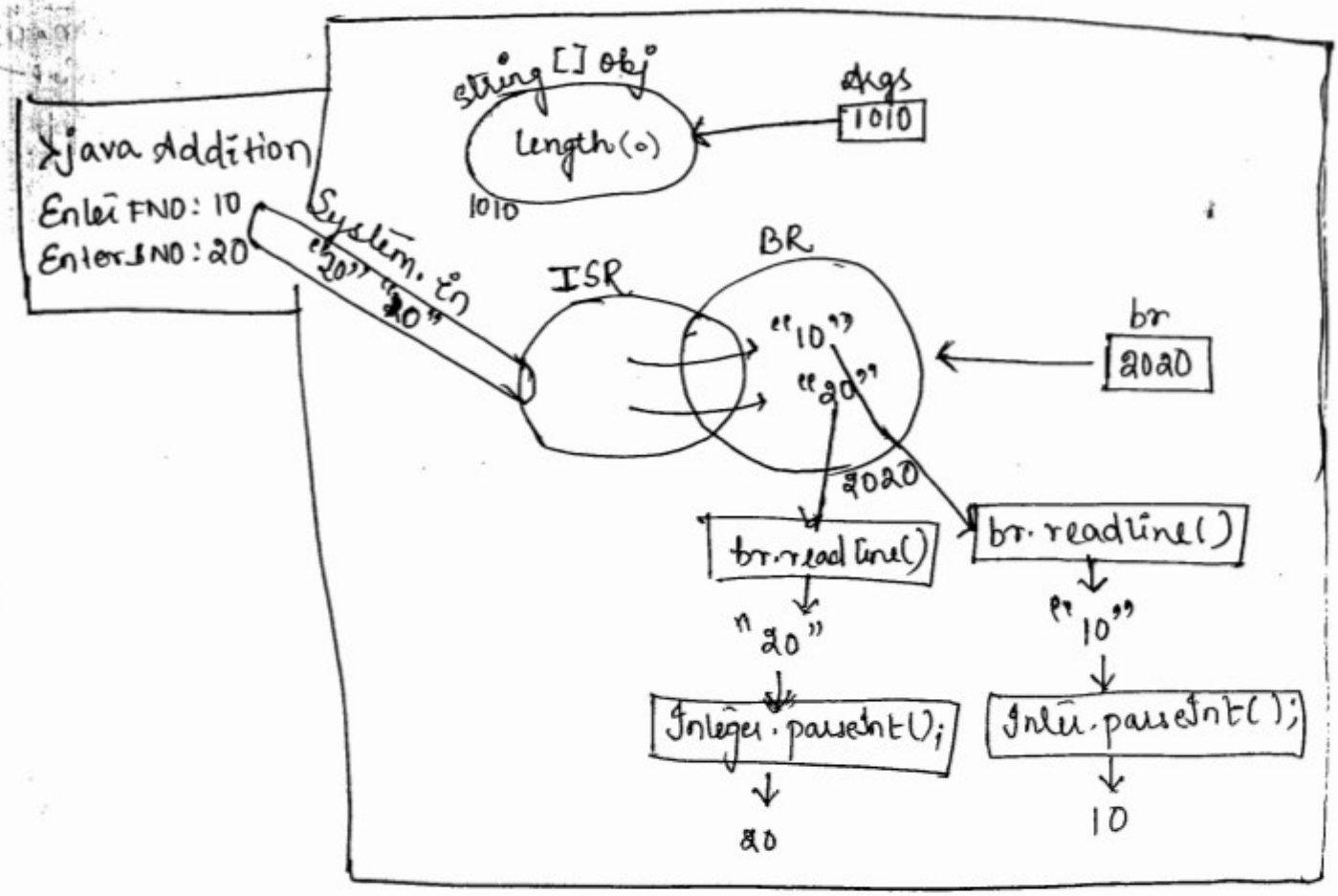
```
// Addition.java
import java.io.*;
Class Addition {
    public static void main (String [] args) throws io exception{
        BufferedReader br= new BufferedReader (new InputStreamReader (System.in));
        System.out.print ("Enter FNO:");
        int x = Integer.parseInt (br.readline());
        System.out.print ("Enter SNO:");
        int y = Integer.parseInt (br.readline());
        int z = x+y;
        System.out.println ("Result:"+z);
    }
}
```

Compilation & execution :-

```
>javac Addition.java
        |→ Addition.class
```

CMD

```
>java Addition
  Enter FNO : 10 ↵
  Enter SNO : 20 ↵
  Result :  30
```



① In every application execution jvm will create and pass String [] object as argument to main method. If we pass cmd line arguments, empty string [] object is created & pass.

② If we pass cmd line arguments, [] object created with these values.

③ In these program we are not passing cmd line arguments, so empty () object is created.

In the statement - `BR br = new BR(ISR(System.in))`

System.in is Keyboard connection object.
We want BR connected to Keyboard, so we passed System.in to BR(IBR).

Whatever the value we enter at console, soon we press enterkey, that value is passed into system.in from there it is stored in BR object, from BR object we will read the data by using br.readline method.

It return data in string form, further we pass it to Integer.parseInt for converting into Int form.

Write a program to read emp data eno, ename, sal, mobile, dept, gender, status from keyboard using BR object.

Employee Info.java

```
import java.io.*;
class Employee {
    public static void main (String[] args) throws
                                        IOException {

BufferedReader br = new BufferedReader (new InputStreamReader
                                        (System.in));
```

```
S.O.P("Enter Eno!");
int Eno = Integer.parseInt(br.readline());
```

```
S.O.P("Enter ename:");
String ename = br.readline();
```

```
S.O.P("enter sal");
double sal = Double.parseDouble(br.readline());
```

```
S.O.P("Enter MobileNo:");
Long MobileNo = Long.parseLong(br.readline());

S.O.P("Enter dept:");
String dept = String.parseString(br.readline());

S.O.P("Enter gender:");
Char gender = Character.parse br.readline().charAt(0);

S.O.P("Enter status:");
Boolean status = Boolean.parseBoolean(br.readline());

    S.O.Pln("Confirm the values to enter:");
    S.O.Pln("Eno:" + eno);
    S.O.Pln("Ename:" + ename);
    S.O.Pln("salary:" + sal);
    S.O.Pln("mobileNo:" + mobile);
    S.O.Pln("Department:" + dept);
    S.O.Pln("Gender:" + gender);
    S.O.Pln("Status:" + status);
  }
}
```

Problem with BufferedReader:-

Every value we are reading from keyboard should be converted into its primitive type using wrapper class methods.

Solution:- In java 5 scanner class is given for converting given no. into its primitive type automatically.

Working with scanner class:-

① Scanner is given in java 1.5 version in java.util package.
② So inorder to use scanner class in our program, we must import java.util package.
③ Scanner class is a convinient class for reading values from keyboard, coz, it has methods for reading & returning given data in its original primitive type.

Below of the methods available in scanner class:-

```
class Scanner implements Iterator {
    public byte nextByte() { ≡ }
    public short nextShort() { ≡ }
    public int nextInt() { ≡ }
    public long nextLong() { ≡ }
    public float nextFloat() { ≡ }
    public double nextDouble() { ≡ }
    public boolean nextBoolean() { ≡ }
    public String next() { ≡ }
    public String nextLine() { ≡ }
    ≡
}
```

→ Don't type and save the program, it is already defined by sun, available in rt.jar, it is only for listing method purpose we have written it here.

→ Scanner class also doesn't support reading char data from keyboard as char type coz there is no parseChar method

→ Using scanner also we must read character as string; it should converted to char by using charAt() method of string class.

→ For invoking (calling) above 9 methods we must create scanner class object connecting to keyboard as shown below

```
Scanner scn = new Scanner(System.in);
```

Above line will create scanner obj in JVM connecting to keyboard. Then using this obj we will invoke above 9 next() methods for reading data from keyboard.

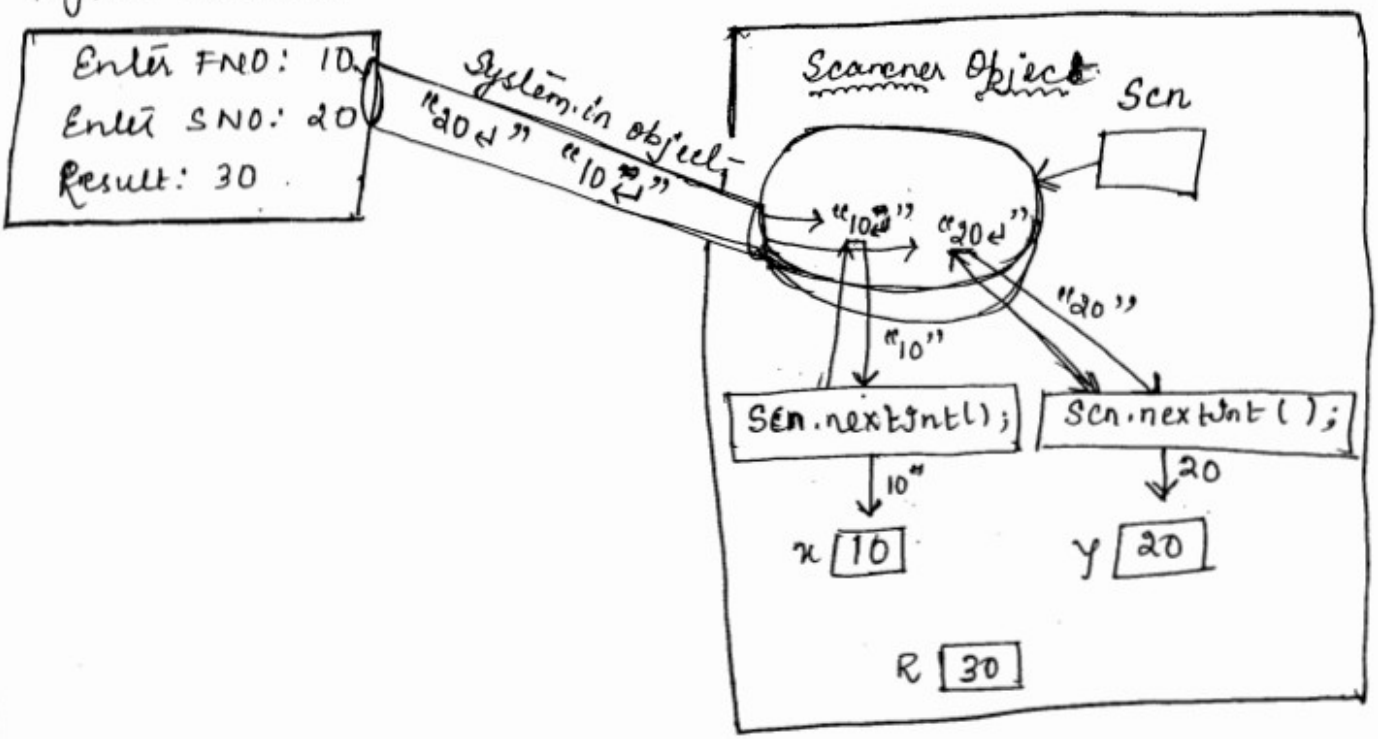
Rewrite addition class for reading inputs from keyboard using scanner object.

```
//Addition.java
import java.util.*;  // for accessing scanner class
class Addition {
    public static void main(String[] args);
    Scanner scn = new Scanner(System.in);
    S.O.P("Enter FNO:");
    int x = scn.nextInt();
    S.O.P("Enter SNO:");
    int y = scn.nextInt();
    int z = x+y;
    S.O.Pln("Result:" + z);
}
```

compilation & execution.

> javac Addition.java

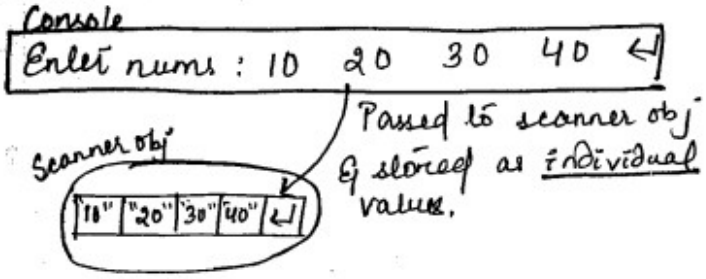
> java Addition



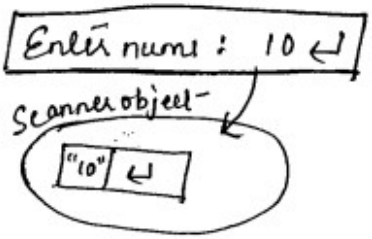
When we call scn.nextXXX() methods, first these methods will verify is scanner obj contain data or not. If data not available then only program execution is passed for reading input from keyboard. If data already available in scanner obj, program execution is not passed, nextXXX() methods will read existing data and will continue program execution with this data

Point to remember:— nextXXX() methods will pass program execution only if data not available in scanner object.

Benefits of Scanner class:- (1) Using scanner obj we can read multiple values at a time in a single input like cmd appl as below.



(2) And also we can pass single value in one input - like in BR class.

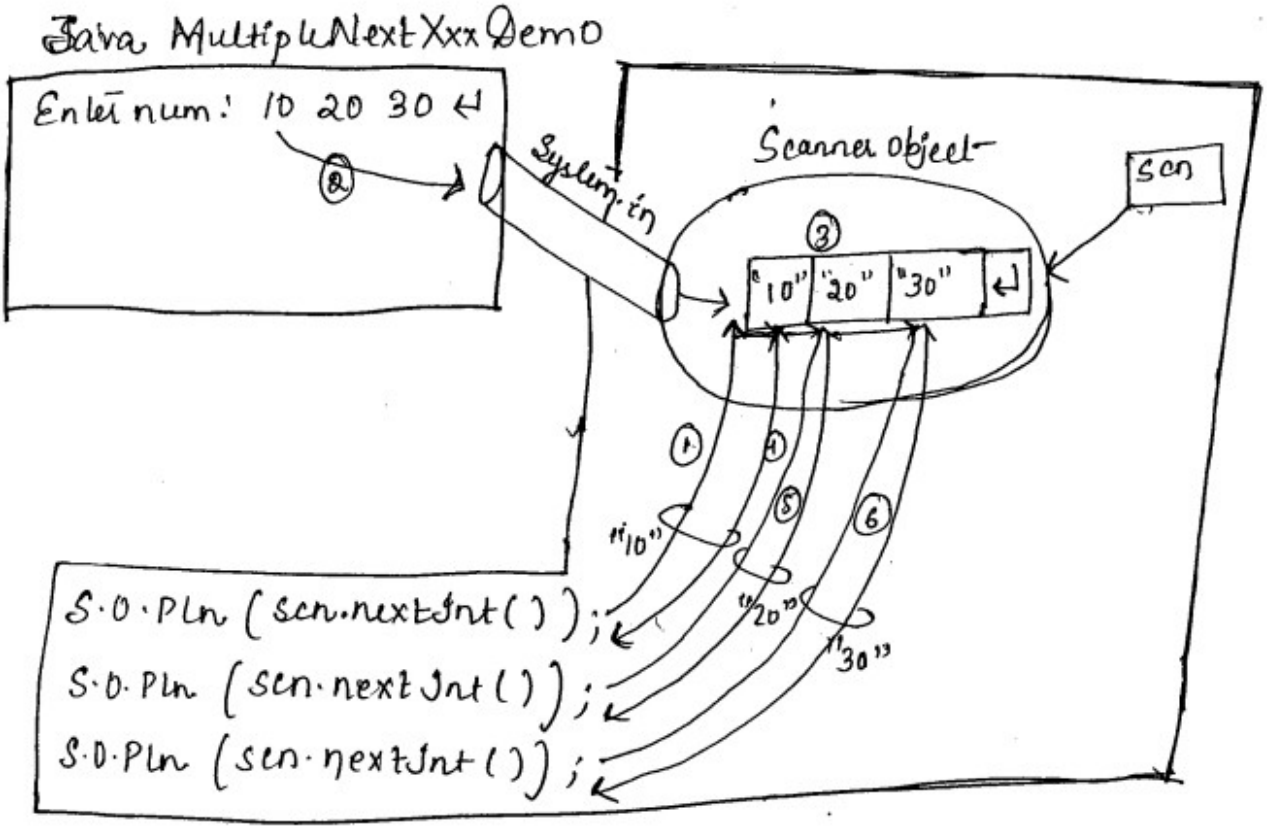


When we call nextxxx() methods multiple times, program execution may or may not pass depending on values available in scanner obj.

If we pass multiple values in a single input - from second call onwards nextxxx() methods will not pass program execution coz scanner obj already contains data - for ex:-

```
// multiple NextXXx Demo.java
import java.util.*;
class MultipleNextXxxDemo {
    P S V main(String[] args);
Scanner scn = new Scanner (System.in);
    S.O.P("Enter name:");
    s.o.p(scn.nextInt());
    S.O.P(scn.nextInt());
    S.O.P(scn.nextInt());
}
```

Case 1:- If we pass 3 values in single input- program will not be passed for second & third nextInt() method calls.



Above program execution is passed only one because we have given all 3 values in single input-

Q Why Scanner class method names are start with 'next'?

A Because these methods will read next location value in scanner obj coz scanner obj cursor bydeafaul- will be pointing to- before first location.
for example as Resultset- Object.

Q Identify how many times above program will pass for below inputs?
Enter nums: 10 20 ↵

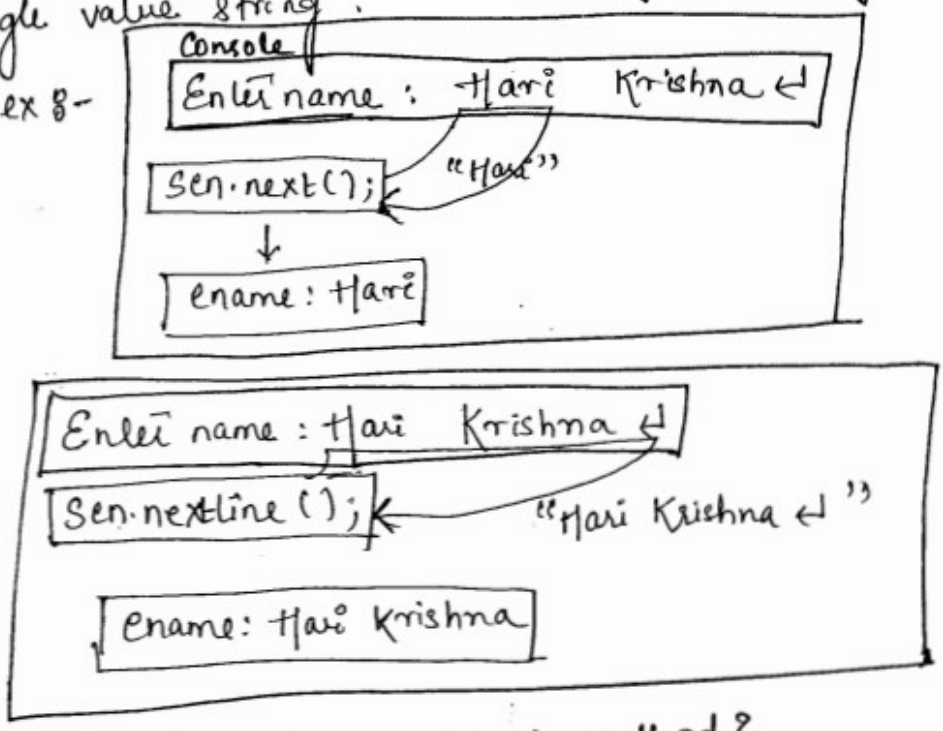
Here cursor will not passed for second nextInt(), but cursor will passed for third nextInt().

Q Diff bet" next() & nextline() method.

Ans- Scn.next() method will read only one word in the given string. If we want to read all words in the given string as individual words we must call next() method multiple times.

If we want to read a complete line including enter key character we must use nextline(). nextline() method will return all words in the given string as a single value string.

for ex Q-



Problem with scn.nextline() method?

Ans When we call scn.nextline() method after any of other & nextxxx() methods, program execution is not passed, because it will read the enter key characters left in scanner object that is passed from get input.

If we use nextInt() & nextline() methods for reading empno & empname, program execution is passed for nextInt() method for reading eno value. But program with execution will not pass for nextline() call for reading empname; here it will read & return enter key character from scanner obj that

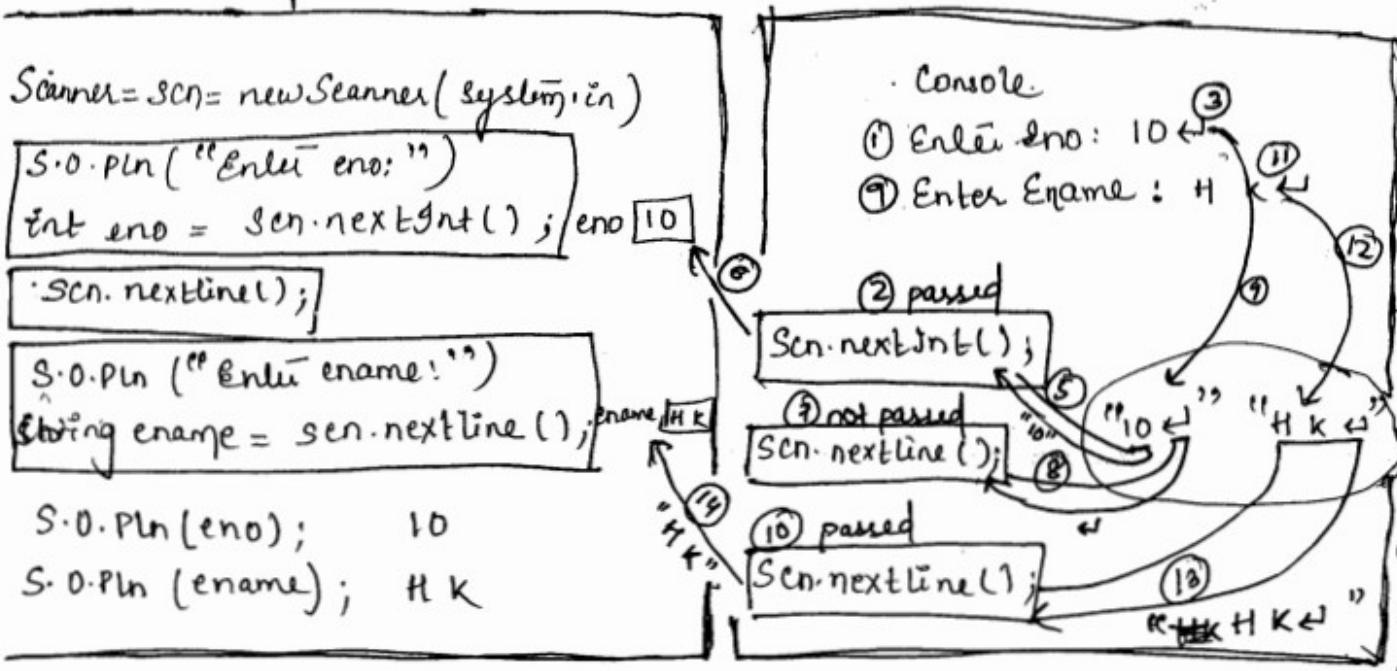
was passed with empname value.

Solution :- ① We must- call nextline() method 2 times, second time nextline() method call should assign to - ename variable.

② First time call scn.nextline() will read enter key character & make scanner obj empty.

③ Second scn.nextline() method will pass program execution, will read ename from keyboard.

Below Diagram shows nextline() method problem with solution



write a program to read employee eno, ename, sal, dept from keyboard using scanner class obj.

```
import java.util.*;
class Employee Data {
  public static void main (String [] args) {
  Scanner scn = new Scanner (System.in);
  S.O.P ("Enter eno:");
  int eno = scn.nextInt ();
  scn.nextline();
  S.O.P ("Enter ename:");
String ename = scn.nextline ();
  S.O.P ("Enter sal:");
Double sal = scn.nextDouble ();
  scn.nextline()
  S.O.P ("Enter dept:");
String dept = scn.nextline();
S.O.Pln ("Confirm the values to enter:");
S.O.Pln ("Eno:" + eno);
S.O.Pln ("ename:" + ename);
S.O.Pln ("salary:" + sal);
S.O.Pln ("Department:" + dept);
}
}
```

RE: InputMissmatchException :- We will get this exception if the passed input is not of type, the called nextXXX() type value, then this nextXXX() method will through this exception.

for ex:- S.O.P ("Enter num:");

```
int x = scn.nextInt();
```

We must pass only int or integer value.

Case 1:- 10↵ ✓

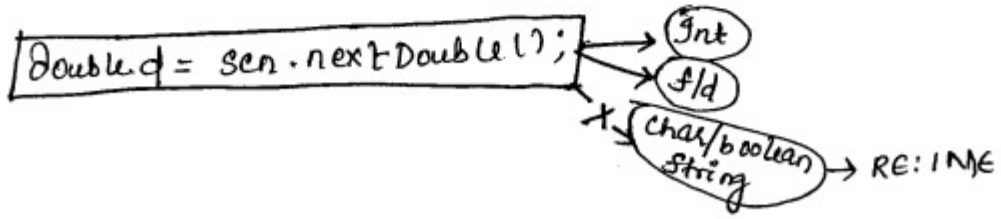
Case 2:- 10.0↵ X RE: IME

Case 3:- a↵ X RE: IME

Case 4:- true↵ X RE: IME

Case 5:- abc↵ X RE: IME

Similarly `Double d = scn.nextDouble();`



Q Identify IME in the below program?

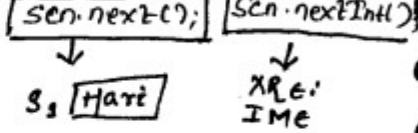
Ans

```
Scanner scn = new Scanner(System.in);
S.O.Pln("Enter values:");
String s1 = scn.next();
int i1 = scn.nextInt();
S.O.Pln(s1);
S.O.Pln(i1);
```

Case 1:- ~~10 20~~ Hari 7279 ✓

Case 2:- 4587 897 ✓

Case 3:- Hari Krishna X



NOTE:- ① Only scn.next() & scn.nextLine() methods will not throw any exception because they will read data as string type, any data can represent as string type.

② Remaining seven nextXXX() methods will throw IME, coz they must convert given value to their internal mapping primitive type.

Problems in the above 3 approaches to input values

① In all above 3 approaches we cannot hide the data we are typing.

② If we are entering confidential values such as password, atm card pin number, OTP number etc, the data we are typing should hide for security point of view, this facility is not available in all above 3 approaches.

Solution :- We must use console class.

Working with console class :-

① Console is a pre-defined class given by sun in java.io package for reading normal values and confidential values from console → (cmd prompt window).

② The class console is introduced in java 1.6. so to use this class we must install jdk 1.6 or higher version in our computer.

③ We cannot create console obj directly using new keyboard like as ~~BR~~ object & Scanner obj created.

④ Below statement is wrong ~~cons~~

```
Console cns = new Console();
```

Above statement meaning create new console (new cmd prompt window) which is wrong idea.

⑤ We must read values from the same console in which our program is running.

⑥ So, SUN has given a method called `console();` in ~~the~~ class System as static method for obtaining current console object in which our program is running, then we must place below statement in our program to obtain console object which is already created.

```
Console cns = System.console();
```

↳ Here we are obtaining existing console obj, using this object we will read username and password from keyboard

⑦ Console class have 2 methods for

(i) `public String readLine()` → It is used for reading normal data like username, mobileno, email add, etc. This method will not hide the input characters we are entering.

(ii) `public char[] readPassword()` → This method meant for reading confidential value as password, OTPno, pinnumber etc. This method will hide input characters. While entering characters this method will not show the characters & even dots are not displayed and even also cursor won't move by typing characters.

Q Develope a program for user registration for reading username, password, email, add etc using console object.

```
// User Registration.java
import java.io.*;
class UserRegistration {
    P S V main (String [] args) {
        S.O.P ("Username:");
        String usn = cns.readLine();
        S.O.P ("Password:");
        Char [] pwd = cns.readPassword();
        S.O.P ("Email:");
        String email = cns.readline();
        S.O.P ("Address:");
        String addr = cns.readline();
        S.O.Pln ("Username:" + usn);
        S.O.Pln ("Password:" + new String(pwd));
        SOPln
        String (pwd);
        S.O.Pln ("Email:" + email);
        S.O.Pln ("Address:" + addr);
    }
```

Problem with Console object:

It will throw ~~Null pointer exception~~ NullPointerException

Console object is available only in cmd prompt window.

② System.Console() will return Console obj only if we execute java program in cmd prompt window.

③ If we run java program in editplus in eclipse console or in eclipse console, System.Console() will return NULL it represents it is not pointing to any object.

④ Then at statement cns.readLine() and cns.readPassword() method calls will throw NullPointerException for testing.

Case 1:- Run above class in cmd prompt window.

→ It is executed successfully

Case 2:- Run above program in editplus console window.

→ will we get NPE.

do below configuration to run java program in editplus console.

Step 1:- Open editplus
2:- Click tools menu
3:- Click ~~user~~ configure user tools
4:- Click add tool
5:- Click program
6:- Enter menu text: JVM
7:- Enter command: java
8:- Select argument: file name without extention
9:- Select initial directory: file directory
10:- Check capture output check box from the statement

After above configuration completed click ok button, open user registration .java file, press ctrl + 1 buttons then program executed in editplus console, you will se NPE.

Problems In All Above 4 Approaches :-

all above 4 approaches development is CUI based (command user interface). In this approach we have 3 problems

problem 1 :- It is not easily navigatable, all option at a time we cannot show, selection option cannot be provided

problem 2 :- We cannot correct the input given before execution.

problem 3 :- We cannot display images and different colors and fonts.

Solution :- We must develope GUI (Graphical user interface) based appl^n using AWT, ~~Stream~~ Swing Programming.

[[Refer last chap in vol 2 text book]]

Working with System properties :- [[refer chap no-2 in vol 2 text book]]

Modifiers & Accessibility modifiers

13/09/17

Modifiers and Accessibility Modifiers

01. What is a modifier, and why modifier ?
02. How many modifier java support and what are they ?
03. Purpose of each modifier ?
04. In which place modifier is allowed in programming element declaration ?
05. Compilation Errors in using a modifier ?
06. To which programming Element a modifier is allowed & not allowed, why ?
07. Complete Discussion on Accessibility Modifiers ?
08. Interview and SCJP questions ?
09. Sample programs and project based code ?

---

01. What is a modifier ?

- A keyword that modifies default operation of a programming element is called modifier.

For Example :-

the statement `int x = 10;` → x [10] will create memory stores value 10 in the memory. Later in the next lines of the code we are allowed to change its value with the statement `x = 20;` → x [~~10~~ 20]

- If we don't want to change variable value, we must declare this variable as ~~final~~ constant by using a keyword final. As shown below :-

```
final int x = 10;
```

then if we want to change its value

Compiler will throw error.

`x = 20` ✗ CE: can not assign value.

In above statement, the keyword final modified the default operation of a variable, that is by default we can change a variable value, but if declare the variable as final we can not change its value, if we try to change value compiler will throw error.

So we can call final keyword is a modifier keyword.

We have 13 modifiers in java, some of them are:-

private, protected, public, static, final .....

What is the use of modifier?

A Modifier is used to change default operation of an programming element.

Example:-

```
class A {
    int x = 10;
}
```

it will get memory when we create object, else does not have memory.

```
class A {
  static int x = 10;
}
```

it will get memory without object creation.

A.java

```
class A {
   int x = 10;
   p s v main(S[] args) {
     Sopln(x); X CE
   }
}
```

the variable x do not have memory by default So compiler will throw error.

A.java

```
class A {
   static int x = 10;
   p s v main(S[] args) {
     Sopln(x);
   }
}
```

Output:- 10

Above case, the variable x has memory without object creation because of static keyword.

04. In which place modifiers is allowed in programming element declaration?

Ans:- We must use modifiers before class keyword, Datatype and return type keyword in a class, variable and method creation

For example:-

```
final class Example {
    final int x = 10;
    final void m1(){}
}
```

✓

```
class final Example {
    int final x = 10;
    void final m1(){}
}
```

✗

02. How many modifiers java supports and what are they?

Ans:- Java supports 13 Modifiers and these 13 modifiers we (Hk) can divide into 3 groups based on their functionality.

(i) Accessibility Modifiers (4)

1. private.
2. <default> / <package-private>.
3. protected.
4. public.

(ii) Execution level Modifiers (8)

1. static.
2. final.
3. abstract.
4. native.
5. transient.
6. volatile.
7. synchronized.
8. strictfp.

(iii) File Level Modifier (1)

1. interface

Proof for this list of modifiers, in google search for class modifiers inside this, you will find all these 13.

Note:- default is not a keyword from AMs point of view. If we do not use private, protected or public then default access level will be applied

07. Complete Discussion on Accessibility Modifiers.

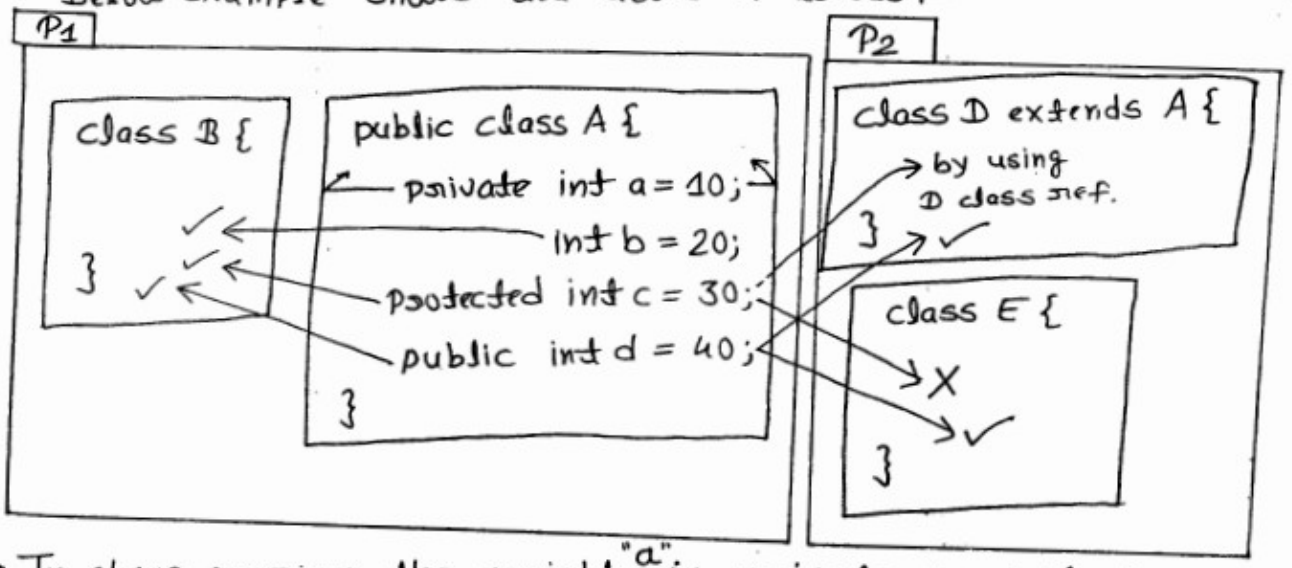
Ans:- A keyword that is used for setting access scope permissions is called accessibility modifier.

It means by using accessibility modifier we can specify where a programming element is allowed to access.

Java supports 4 access levels, they are:-

1. Class level (within the class) ← private
2. package level (within the package) ← default
3. outside package but only inside sub class that to by using the same sub class reference. ← protected
4. Any where in the project. ← public

Below example shows all above 4 levels:-



- In above program, the variable "a" is private variable because it is declared with private modifier. Because the var. a is private, we are allowed to access it only inside the members of class A. If we access this variable from other classes, class B, compiler will throw error "a has private access".
- The variable b is default variable, because it is declared without modifiers private, protected and public. Because this variable b is default, we are allowed to access it either in same class A or from other class members,

but we are allowed to use/access this variable b only within the current package members. If we access it in other package classes compiler will throw error "variable b is not public".

- The variable c is protected variable, because it is declared with protected modifier. Because this variable c is protected variable we can access it either in same class or in same package classes or also in other package classes. The rule here is in other package class should be sub class and moreover we must access this variable using the same sub class reference.
- The variable d is public variable, because it is declared with the public modifier. Because the variable d is public, we can access it anywhere in the project. It means we can access public variable either within the same class or in within the same package or in other package classes either sub class or normal class that either with same class reference or with sub class reference.

Que:- What is the difference between default and protected modifiers?

Ans:- default ~~class~~ members can be accessible only with in the same package class either in normal class or in sub class, either by using its own class reference or by using its sub class reference.

- protected members can be accessible either with in the samp package classes or in other package classes.
  → As long as we are accessing protected members with in the same package classes, it will act exactly as default members. It means within the same package classes we can access protected members either in normal class or in sub class either by

Q. What is the difference between protected & public ?

Ans:
- Both protected and public members are allowed to in other package classes, with minimal difference. public member can be access either in normal classes or in sub classes either by using its own class reference or by using any sub class ref.
- But protected members is allowed to access with some restriction, that is protected member is allowed to access only inside sub class ~~reference~~ that to only by using this same ~~ete~~ sub class ref. It means protected members is not allowed to access in other package in normal class and with its own class reference or with other sub classes reference.

16/09/17

Q. What is the output from below program ?

Ans:- Example.java Case #1: Accessing member in other class without package concept.

```
class Example{
  private int a=10;
          int b=20;
protected int c=30;
public int d=40;
public static void main(String[] args)
   Example e1 = new Example();
    S.o.p.ln("a:"+e1.a);
    S.o.p ln("b:"+e1.b);
    S.op ln("c:"+e1.c);
    S.op ln ("d:"+e1.d);
  }
}
```

Sample.java

```
class Sample {
    public static void main(String[] args) {
        Example e1 = new Example();
        //S.op.ln("a:"+e1.a); X CE: a has private access
        S.opln("b:"+e1.b); ✓
        S.opln("C:"+e1.c); ✓
        S.opln("d:"+e1.d); ✓
    }
}
```

In above program we have not used package in both Java file, So we must assume both java files created in same package. So inside class sample default, protected and public members of class Examples are accessible.

Case #2: Accessing one class member in another class within the same package

A.java

```
package p1;
class public class A {
    private int a = 10;
            int b = 20;
    protected int c = 30;
    public int d = 40;
p.s.v main(String[] args){
    A a1 = new A();
    S.opln("a:"+a1.a); ✓
    Sopln("b:"+a1.b); ✓
    Sopln("C:"+a1.c); ✓
    Sopln("d:"+a1.d); ✓
}
```

B.java

```
package p1;
class B {
    p s v main(String[] args)
        A a1 = new A();
    //S.opln("a:"+a1.a); X CE
        S.opln("b:"+a1.b); ✓
        S.opln("C:"+a1.c); ✓
        S.opln("d:"+a1.d); ✓
    }
}
```

The class A and B are created in same package p1, hence we can access default, protected and public members of class A in class B by using class A reference, Because within the same package accessed.

Case #3: Accessing different package sub class

C.java

```
package p2;
import p1.A;
class C extends A {
  p.sv main(String[] args){
   A a1 = new A();
  //Sopln("a:"+a1.a); x
  //Sopln("b:"+a1.b); x
  //Sopln("c:"+a1.c); x
    Sopln("d:"+a1.d); ✓
    Sopln();
    C c1 = new C()
  //Sopln("a:"+c1.a); x
  //Sopln("b:"+c1.b); x
    Sopln("c:"+c1.c); ✓
    Sopln("d:"+c1.d); ✓
  }
}
    A a2 = new C();
    Sopln(a2.c); x
    Sopln(a2.d);
  }
}
```

class C defined in different package, we can not access so we can not access private and default members. also we can not access protected members using its own class reference.

This protected member we can access using sub-class reference.

Here a2 type is class A so we can not access protected member even though object is sub type.

Case #4 Accessing member in other package's normal class members.

D.java

```
package p2;
import p1.A;
class D{
  psv main(string [] args){
    A a1 = new A();
   //System.out.println("c:"a1.c); X
    System.out.println("d:" a1.d);
    System.out.println();

    C c1 = new CC();
    // S.opln("c:" +c1.c); X
     S.opln("d:" + c1.d);
  }
}
```

Inside another package, Inside normal class we can not access a protected member either using own class reference or by using sub class ref. Compiler throw error.

Case #5 Accessing protected member in other package inside sub class using own class reference, using other sub class ref and using same subclass reference.

E.java

```
package p2;
import p1.A;
class E extends A {
  P.S.V main(String[] args) {
   A a1 = new A();
  // S.opln("c:"+a1.c); X
   S.opln("d:"+a1.d); ✓
   S.opln();

   C c1 = new CC);
  // S.opln("c:"+c1.c); X
   S.opln("d:"+c1.d); ✓
   S.opln();

   E e1 = new E();
   S.opln("c:"+e1.c); ✓
   S.opln("d:"+e1.d); ✓
  }
}
```

own class reference not allowed to use to access protected in another package.

diff. sub class reference not allowed to use in another sub class to access protected members.

Same sub class reference it is allowed

Q. How should we declare a variable to be access only by Sub class object reference?

Ans:- protected.

Q. We have a predifined class java.lang.Object, which is super class of all classes. In this class we have 12 methods among these methods, 1 method is declared as private 2 methods are declared as protected and remaining 9 methods are declared as public, Why clone() and finalize() these 2 methods are declared as protected in Object class?

Ans:- To force us to access these two methods only in sub-class that by using the same sub-class reference.

It means SUN dont want allow us to access clone() and finalize() method on one class object in different class, because these two methods creates data security level problems.

So, these 2 methods must be accessed inside a sub class on the same sub class object.

Q. Is protected accessibility modifier rule applied to static variable and static method ?

Ans:- Not applicable to static members it means protected accessibility modifier static variable and static method will behave exactly same as public.

- protected modifier rule is not applicable to static, because static members are accessed without object directly by using class name.
- private and default accessibility modifiers rules are applicable to static variable and static methods.

Q. Identify which accessibility modifier is allowed to which programming elements

(or)

Are all 4 Accessibility Modifiers allowed to all programming elements

(or)

Is it possible to apply private and protected AMs to a class ?

(or)

Is it possible to apply AMs to a method parameter and local variable ?

(or)

Is it possible to declare inner class as private and protected ?

Ans:- • All 4 AMs are not allowed to apply to all programming elements. Based on AMs functionality we are not allowed to apply some of the AMs to some of the programmingo elements.

We can not apply on declare private and protected AMs to outer class, because private and protected modifiers are allowed to only class members that means the programming element that is declared inside a class. Private is not allowed to outer class, because the meaning of private is this member is allowed to access only current enclosing class. Because outer class does not have enclosing class, we can not apply private.

We can not declare outer class as protected, because protected accessibility modifier means this member is allowed to access outside package in its enclosing class's sub-class. Because outer class doesn't have enclosing class, protected not allowed to outer class.

For example:-

```
// private class I1 { } X CE: modifier not allowed here.

class I2 { } ✓

// protected class I3 { } X : protected mod. not allowed here.

public class I4 { } ✓
```

Note:- If we declare class as public, java file name must be same as this public class name.

To understand which PE, AM's are allowed or not allowed read by heart

1. private and protected Modifiers are allowed to only class member, it means the members those are declared outside method and inside class block { }.
2. Whereas default and public modifiers are allowed to package and class members both, it means these are allowed to outer class and the members defined inside outer class.

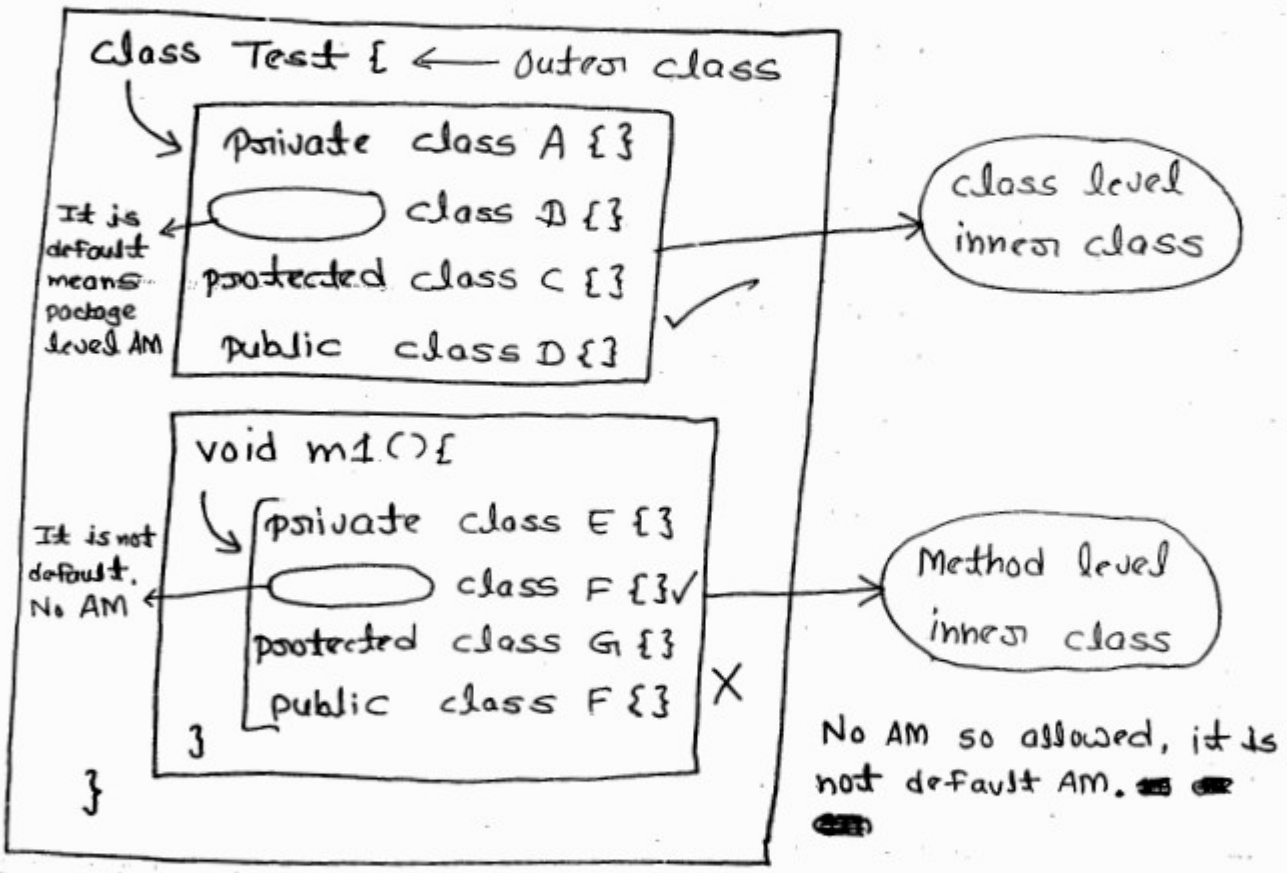
Outer class is a member of package.

Q. What are the modifiers allowed to inner class ?

⇒ we can apply all 4 AMs to the ~~class~~ inner class created directly inside a class directly.

We can't apply all 4 AMs to the inner class i.e. created inside a method, because AMs are allowed to class members not method members.

Example:-

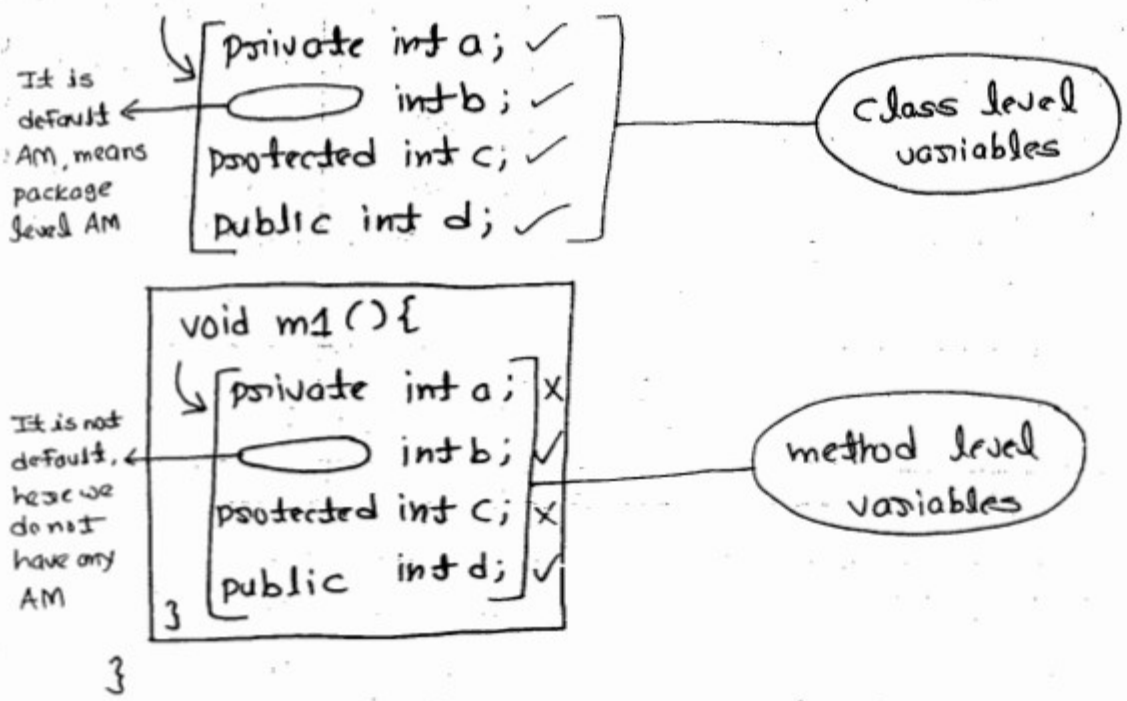


Q. What are the modifiers allowed to variable ?

⇒ like innerclasses we can also create variable either inside a class or inside a method.

All 4 AMs are allowed to class level variables.

But All 4 AMs are not allowed to method level variables, parameter and local variables.



class level variables → All 4 AMs allowed ✓

Method level variables → All 4 AMs not allowed X

Q. What are the modifiers allowed to a method ?

⇒ Java supports 2 types of method :-

(i) abstract method

(ii) concrete method.

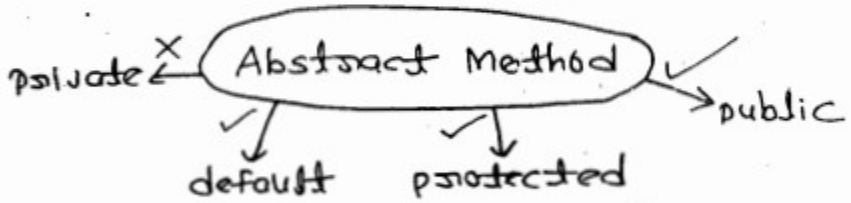
A method that doesn't have implementation body {} is called abstract method.

For e.g. `abstract void m1();` ← ; mandatory

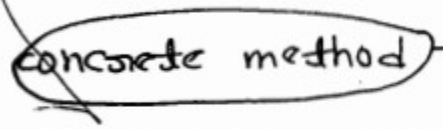
A method that has implementation body is called concrete method.

For e.g. void m2(){}

Since abstract method doesn't have implement body it must be inherited to sub-class, sub-class programmer must provide implementation body to abstract method. So, to inherit abstract method to sub-class we can not declare abstract method as private. we can declare abstract method as default/protected/public.

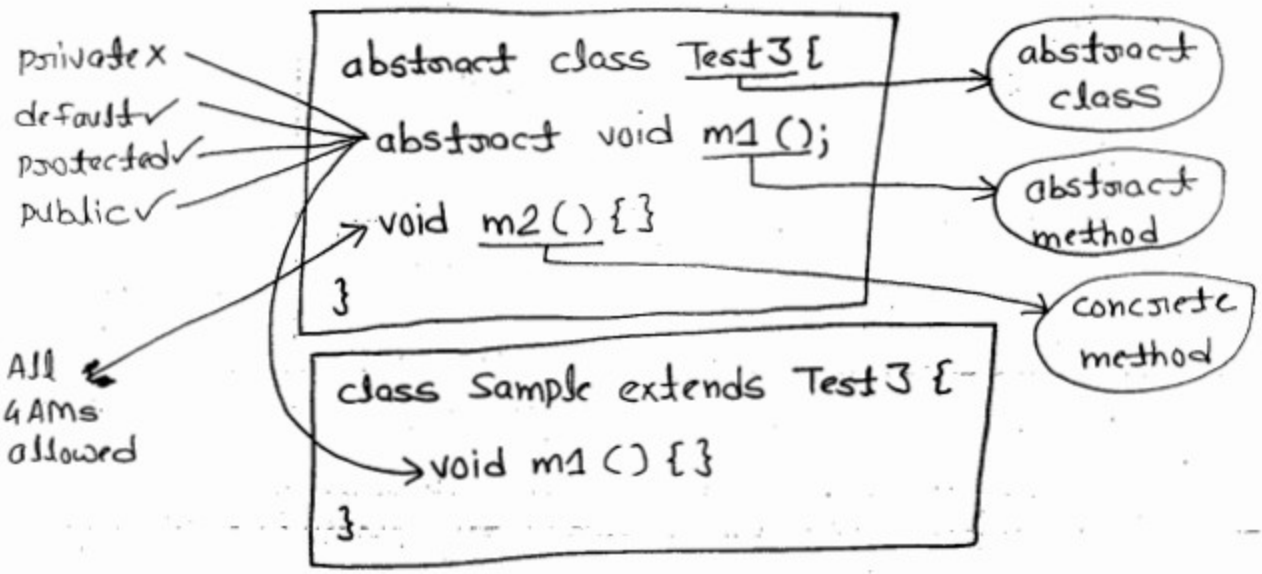


Concrete method already has implementation body, so it is optional to override concrete method in subclass, hence we can declare concrete method as private to stop inheriting to sub-class and also we can declare concrete method as default or protected or public, to inherit concrete method to subclass.

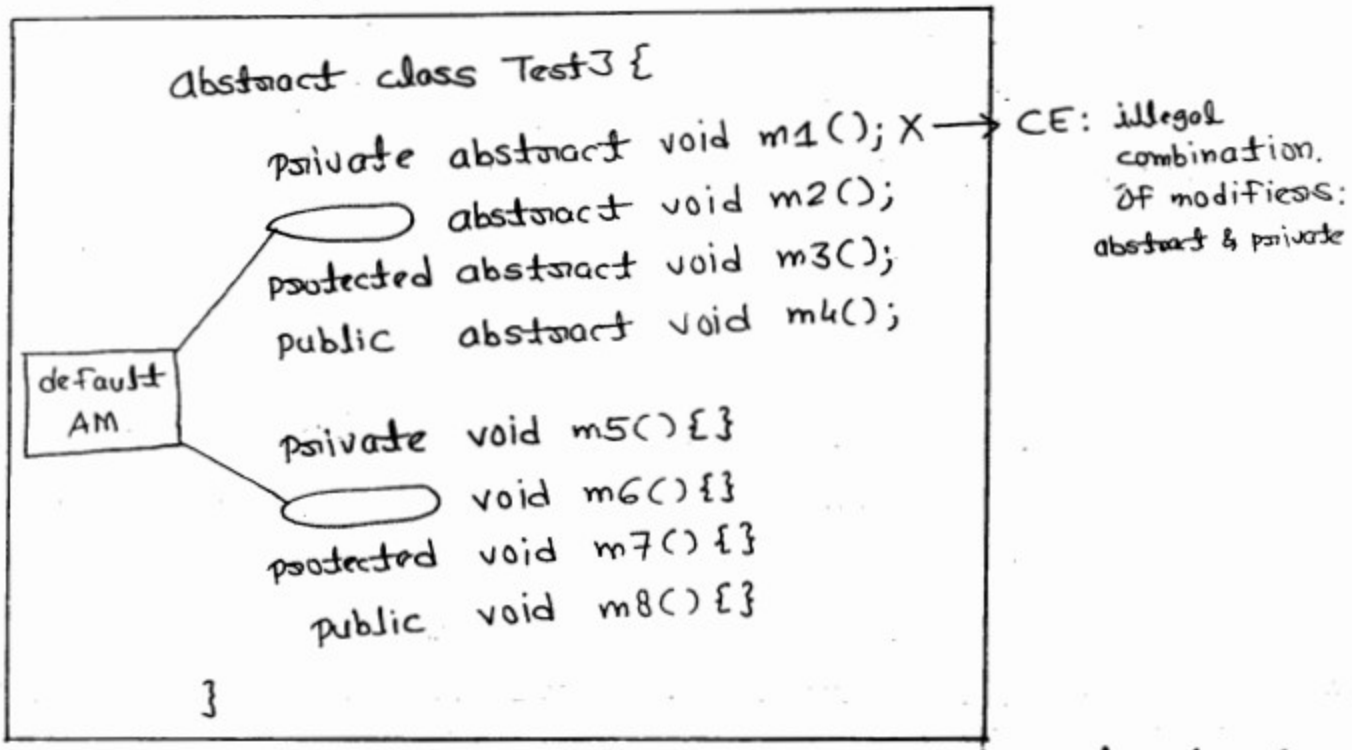


Rule: If a class contain abstract method declare this class as abstract else compiler will throw error.

Below program will show you creating abstract class, abstract method, concrete method, sub-class and allowed AMs



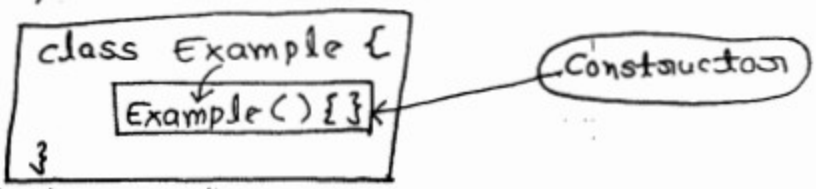
Identify compile time errors in the below program :-



Q. What ~~AMs~~ are the AMs ~~are~~ allowed to constructor?

⇒ • A constructor is a kind of method whose name is same as class name and doesn't have return type, even we can't place void.

For example:-



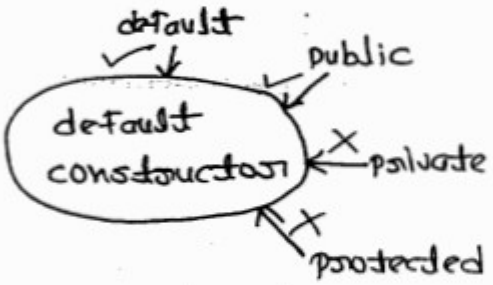
Java supports 2 type of constructor :-

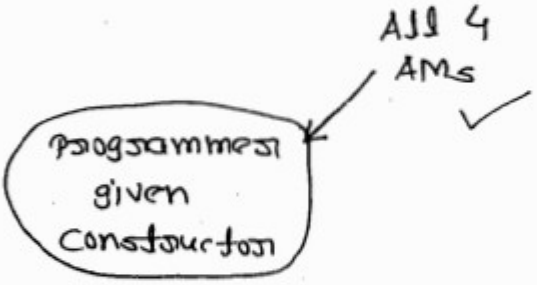
(i) default constructor (compiler given)

(ii) programmer given constructor.

default constructor AM are same as class AMs, so its AMs are default and public. private or protected are not allowed to default constructor, ofcource we can't add private and protected AMs to default constructor, because it is created by compiler.

To the programmer defined constructor (parameterized or non-parameterized) all 4 AMs are allowed.

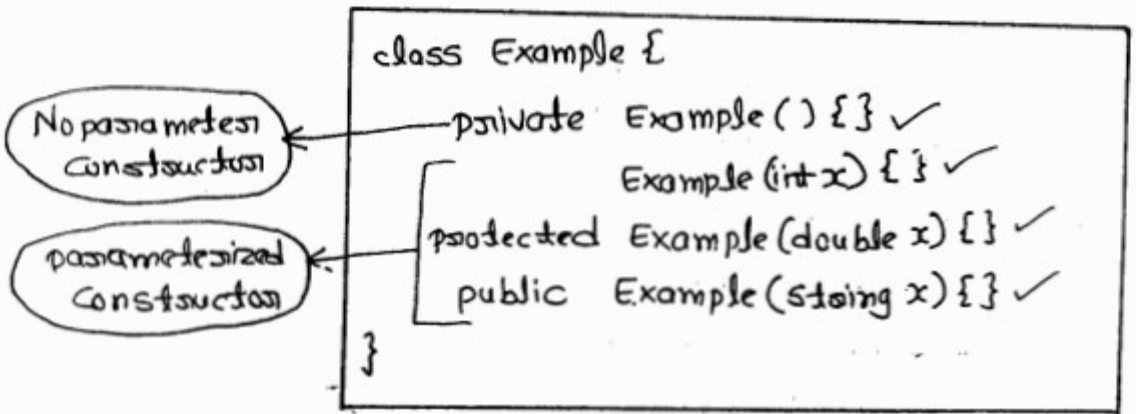




Q. Can we declare constructor as a private ?

⇒ Yes, we can declare constructor as a private, so that other class programmer can't create object of our class.

Below program will show you declaring constructor with all 4 AMs.



```
class Test4{
// Example e1 = new Example(); X CE
   Example e2 = new Example(5); ✓
   Example e3 = new Example(5.0); ✓
   Example e4 = new Example("Hk"); ✓
}
```

We cannot access private constructors in other class to create object

Q. What are the AMs are allowed to a block ?

⇒ All 4 AMs are not allowed to a block, because we can't access a block by default, because block doesn't have name.

```
class Test5{
   private {} X
   {} ✓
   protected {} X
   public {} X
}
```

It is not default AM

Here No AM

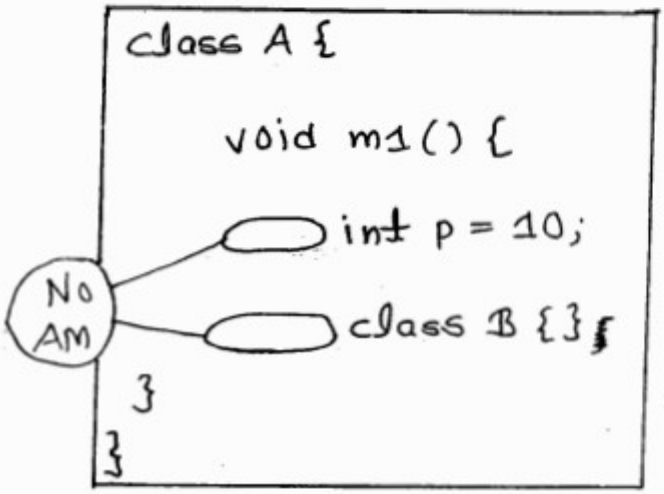
PEs / AMs	Package	outer class	Inner class		Variables		Methods		Constructors		Blocks
			CLIC	MLIC	CLV	MLV	AM	CM	default	Param. & Non Param.	CL & ML
private	X	X	✓	X	✓	X	X	✓	X	✓	X
default	X	✓	✓	X	✓	X	✓	✓	✓	✓	X
protected	X	X	✓	X	✓	X	✓	✓	X	✓	X
Public	X	✓	✓	X	✓	X	✓	✓	✓	✓	X

Q. What is the default AM to the members created inside a method ?

⇒ • No A.M. applied to a member declared inside a method. If we apply AM (private, protected, public) to inside method members compiler will throw error.

• If we don't apply these 3 AM, compiler it itself will not apply any AM.

For example:-



In above program, for variable p and did not apply any AM and compiler also will not apply and AM. Hence the default modifier of method level member is no modifier.

Q. What is the default AM of class and its a members?

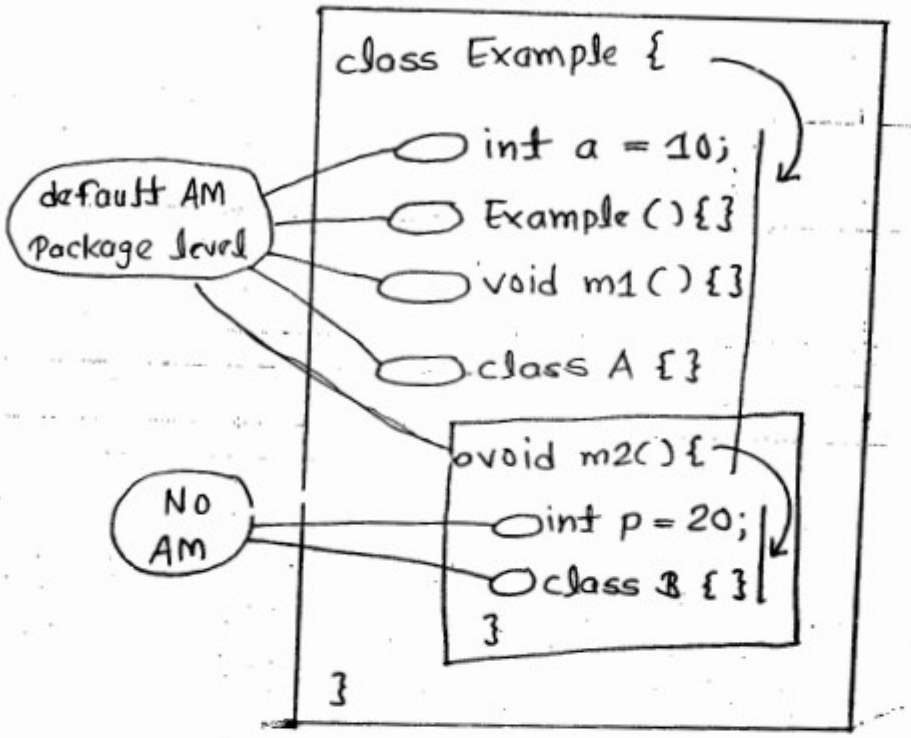
⇒ default AM.

It means if we declare a class, variable or method or innerclass or construction inside a class without applying AMs (private/protected/public) compiler will not add any of these 3 keywords and left that programming element as it is.

• Hence the class and its member are accessible in

For e.g.

Create a class as shown below without applying any Accessibility Modifiers.



Compile above programs and read its byte code, by using "javap" command. like javac and java commands "javap" command also java software binary file it is available inside JDK/bin folder. It is used for reading byte code of a class or interface or enum or annotation.

Q. What is the default AMs of an interface and its members.

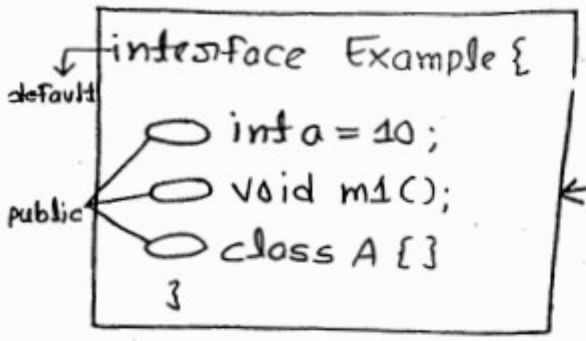
⇒ Interface ~~members~~ default AM is "default".

Interface members default AM is "public".

It means if we do not apply private/protected/public AM to interface declaration, compiler will not add any of these three AM, here interface is accessible upto current package.

If we do not apply private/public/protected AM to the interface members declaration compiler will add "public" to interface members declarations.

For e.g.



~~compile~~ Compile and read bytecode of this interface using javap command you will observe the changes added by compiler

>javap Example

we can declare members inside a class as private/default/protected/public., but we can't declare members inside interface as private or protected, compiler will throw error. If we declare member inside interface, compiler will not throw error, ~~rather~~ rather compiler will add public to their declaration.

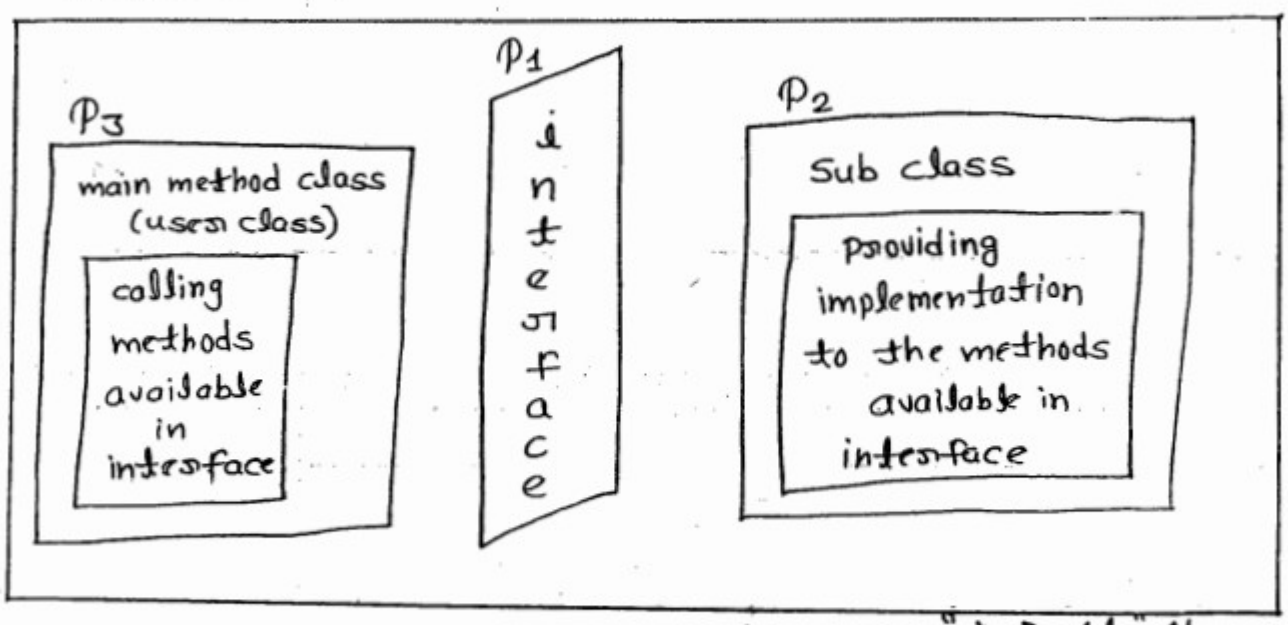
Q. Why we can't declare interface members as default or protected, why they are by default "public"?

⇒ interface is a mediator between subclass ~~programmer~~ and user class programmers. It will specify ~~them~~ what methods should implement in sub class and what method should invoke or call in user class.

Hence the variable and methods declared inside interface must be accessible to sub class and user class for implementing and accessing them.

Therefore for accessing interface members inside sub class and inside user class from all different packages of this project, interface members must be public.

Below diagram will show you interface requirement acting as a mediator between sub-class and user class :-



If we declare interface members as "default" they are not accessible from others

If we declare interface members as "protected", they are accessible inside sub class but not from user class from other package.

To access interface members from all packages in project from both user class and sub class, interface members must be public.

Q. Is it possible to apply more than one AM to single programming element.?

⇒ Not possible, because there is no meaning of declaring member as private, protected and public.

For e.g.

```
class Example{
    private protected public int a = 10; XCE
}
```

CE: Illegal combination of private and public.

Variables and types of variables

Que.? / Types of Variables.	Purpose.	Life time		Scope.	Modification Effect	Alias Name	Default Value	Allowed Modifiers
		Created	Destroyed					
Static Variable	used to provide one copy memory to store value common to all object.	memory is created when class is loaded into JVM.	memory destroyed when class is unloaded from JVM (OR) JVM shutdown.	can be accessed in all methods of current class and also can be accessed in other class method whereever class is accessed	✓	1.) class variable 2.) Field 3.) property	✓	1.) All 4 AMs 2.) static, final, transient, volatile.
Non Static Variable.	used to provide seperate copy of memory for each object individually to store the object specific value.	memory is created only when new instance/object is created.	memory is destroyed when its instance/object is destroyed.	can be accessed in all methods of current class and also from other class method but only by using object reference	✓ provided if all methods are using same object.	1.) instance variable 2.) Field 3.) property 4.) Attribute	✓	1.) All 4 AMs 2.) static, final, transient, volatile.
Parameters	Used to receive and store input values of an operation.	memory is created when its method is called.	memory destroyed when its method execution is completed and control go out of this method.	can be accessed only in it enclosing method till last line of this method.	✗	1.) Method variables 2.) Auto variables 3.) stack variables	✗	final
Local Variable	used to store output value or result of this operation.	memory is created when method is called and its creation statement executed.	memory destroyed when its enclosing block (or) enclosing method execution is completed.	can be accessed only its enclosing block (or) only its enclosing method till last line.	✗	1.) Method variable 2.) Auto variables 3.) stack variables	✗	final