**LANGUAGE FUNDAMENTAL**

**Data types**

**Int numbers**

**Float decimal-10.22…7**

**Double decimal -10.12321…16**

**Char ‘a’,’4’,’$’**

**String “welcome to java lang\_”**

**Boolean True/False**

**Int a=10;**

**Float a=10.2f;**

**Double d=10.23;**

**Char ch=’a’;**

**String str = “welcome”;**

**Boolean status = false;**

**Identifiers/ varriables**

A name in the program is an identifier it may be class name or method name, variable name or label name.

Int a=10

Int a=10,b=20,addation;

**Rules for defining Identifiers**

1. A java identifier is a sequence of characters, where each character may be a letter from *a-z* or *A-Z* or a digit form *0-9* or currency symbol *$* or connecting punctuation *\_* , if we are using any othersymbol we will get Compile time error .

Int a =10;

Int \_a=20

Int sum\_1 = 20

Int var\_a=10;

1. Identifier should not be starts with digit.
2. There is no length limit for java identifiers but it is not recommended to take more than 15 length.

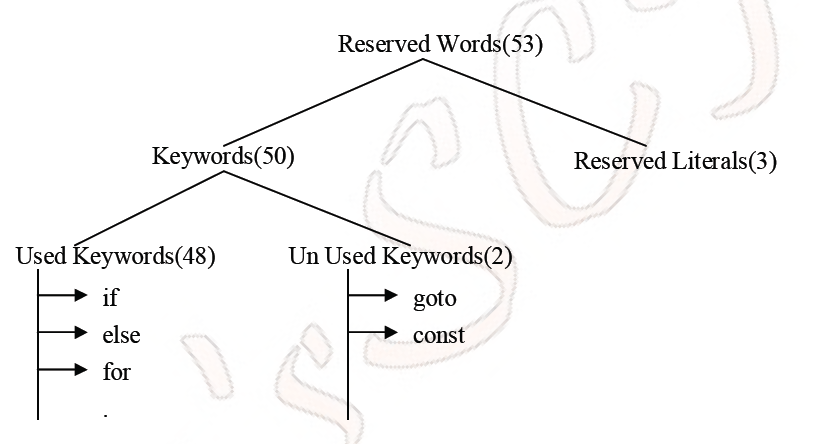
Int arithematicvarr =10;

1. Java Identifiers are case sensitive

Note: All Reserved keywords is not eligible to make an identifier

**Reserved Words**

Some identifiers are reserved to associate some functionality or to represent values, such type of reserved identifiers are called “*ReservedWords*”.

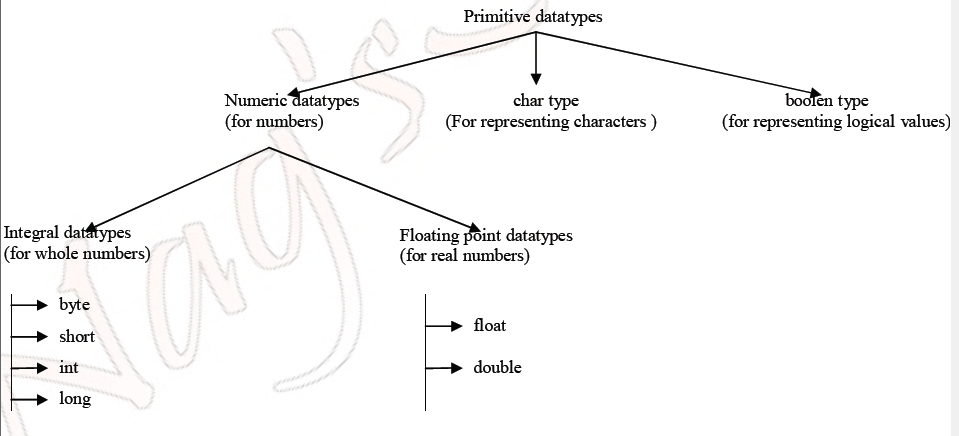


**Note:** All reserved words in java contain only lower case alphabet symbols.

**Datatypes:**

In java every variable has a type, every expression has a type and all types are strictly defined.

All the assignments should be checked by the compiler for the type compatibility. Hence java language considers as strongly typed language.

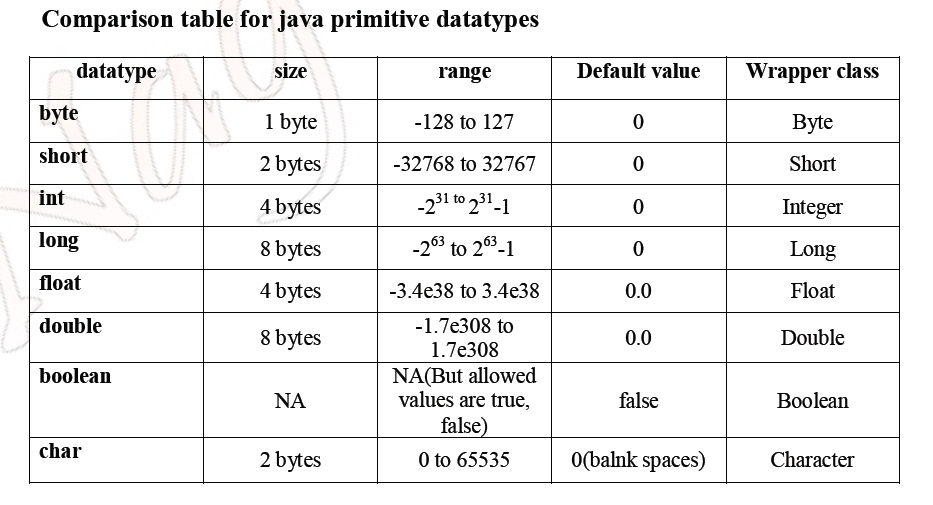


+

Note:

Except boolean and char all the remaining data types are signed data types i.e. we can represent both +ve and –ve numbers.

Double a;



WAP to print welcome message ?

Method:

Void demo()

{

body

}

Int demo()

{

Body;;;

Return 0;

}

Static void accept()

{

body

}

Class Xyz

{

Void sai()

{

System.out.println();

}

Void nikitha()

{

}

Public static void main(String sai[ ])

{

System.out.println(“welcome to java language”);

}

}

Public : it’s a modifier

Private

Protected

default

syntax;

class class\_name

{

Body

}

Object creation syntax

Classname obj = new Calssname();

class Test

{

int a,b; non-static variable

static int b; static variable/method/blocks

public static void main(String[] args)

{

Test t1=new Test();

System.out.println(t1.a);

System.out.println(t1.b);

System.out.println(b);

}

}

How to create object

Syntax

ClassName obj = new CalssName();

Test t = new Test();

Output:

0

0

There are two types of assignments

1. Implicit Assignment (JVM)
2. Explicit Assignment (Programmer)



**String literal**

A sequence of character with in double quotes is String literal.

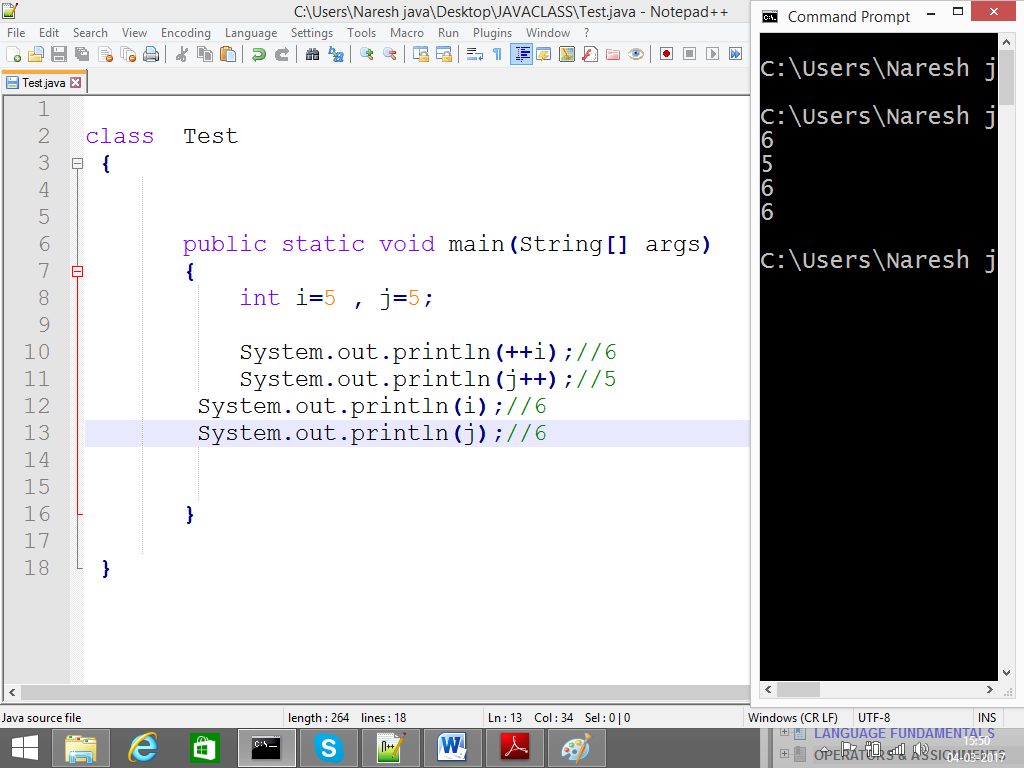


**OPERATORS & ASSIGNMENTS**

1. increment & decrement operators.

I++ /++I

i-- /--i



1. arithmetic operators. + , - , \* , / ,%

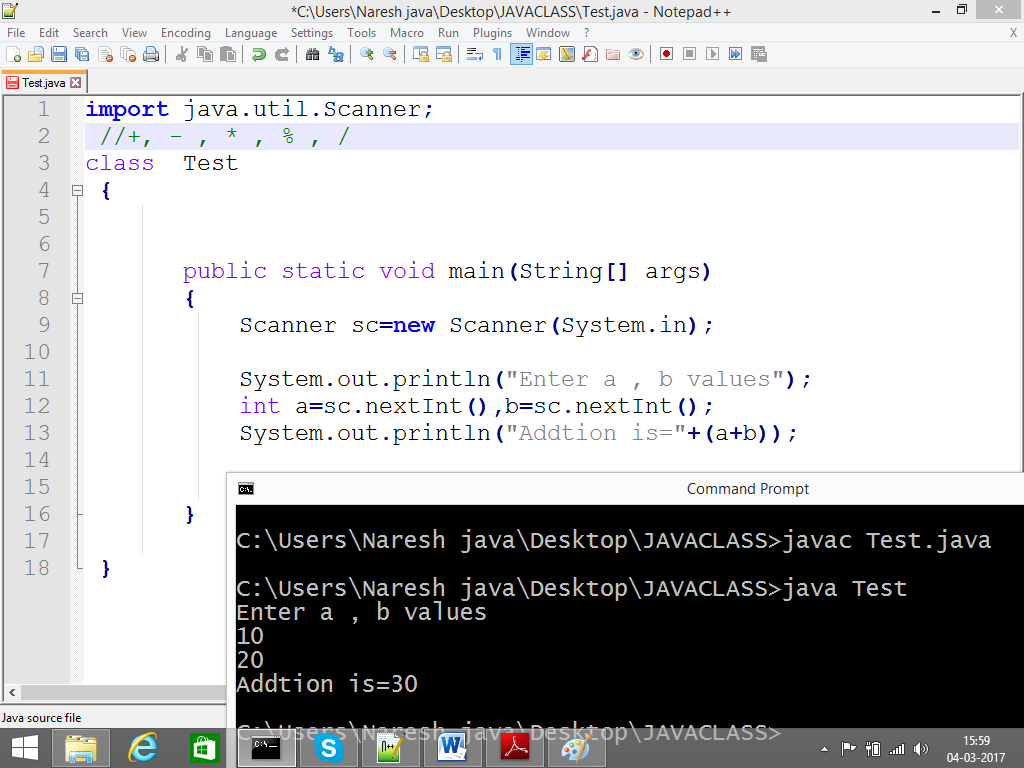
6/2=3

6%2=0

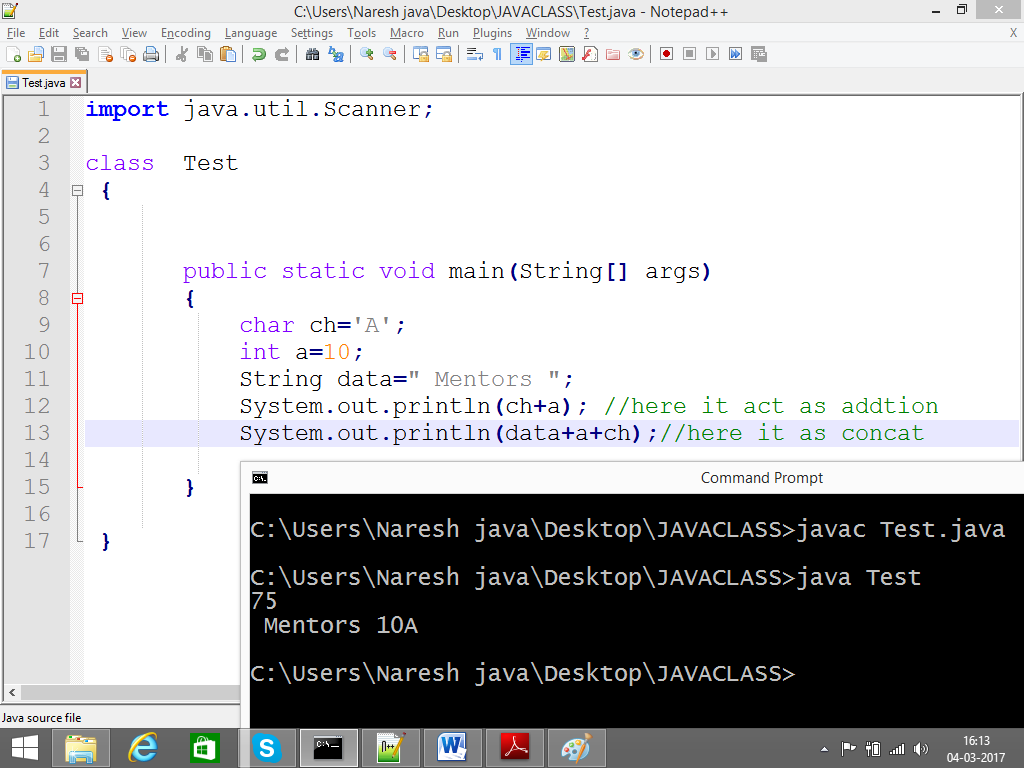
2)6(3-->Qu

6

0 -->reminder

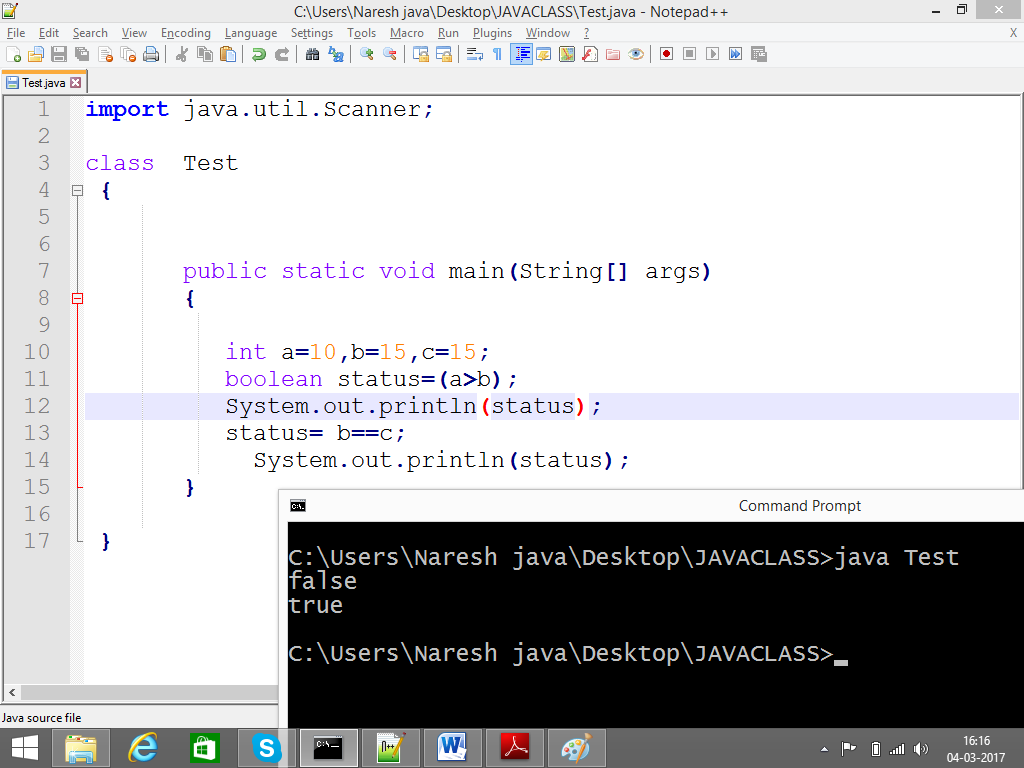


1. string concatenation operators. +



1. Relational operators

< , > ,<= ,>= ,== ,!=



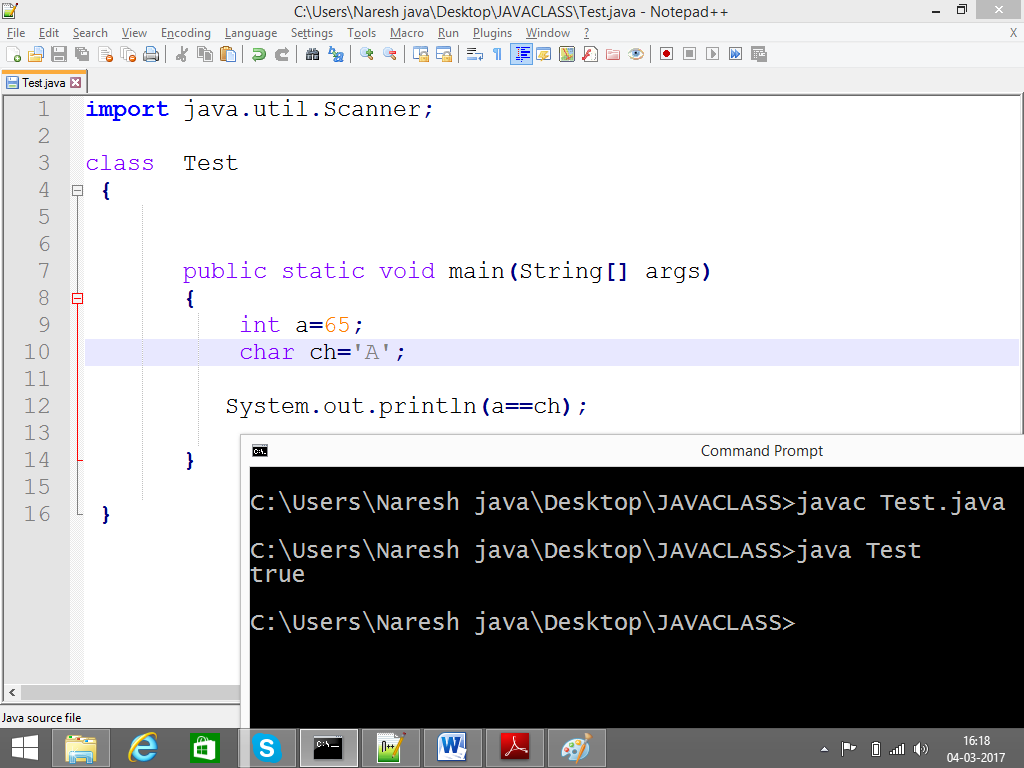
1. Equality operators

= = And !=

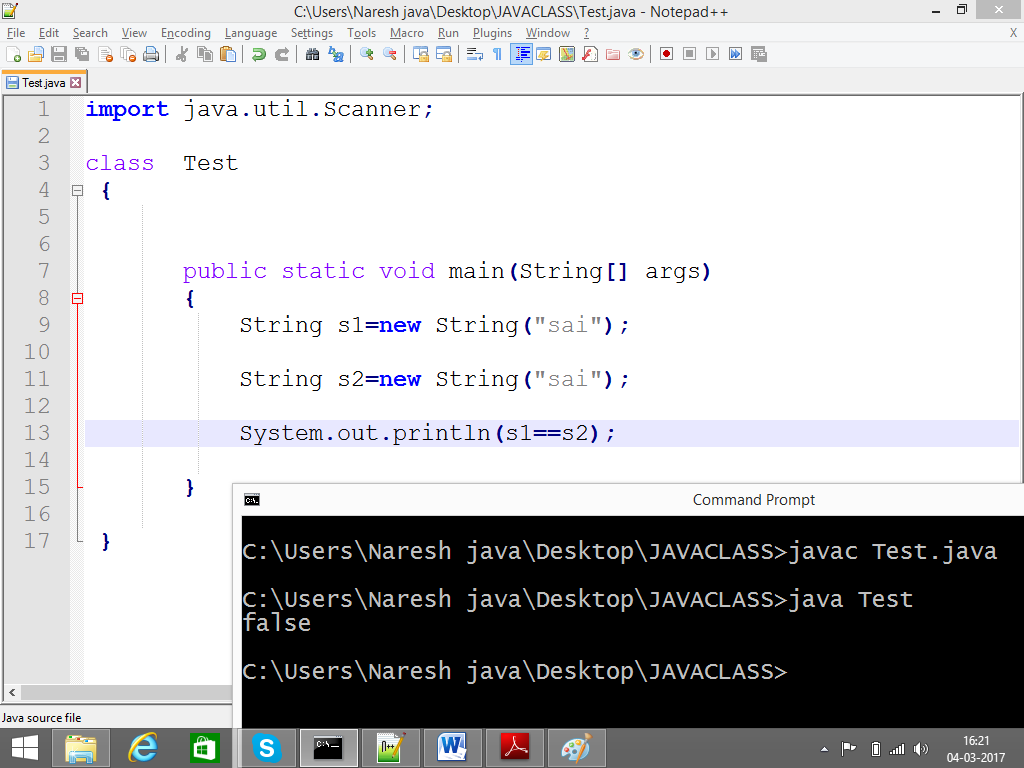
In primitive level it checks content.

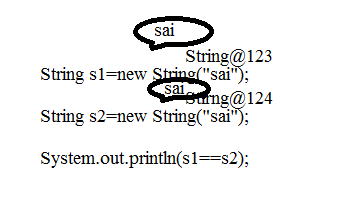
=

= =



In reference level it checks address.



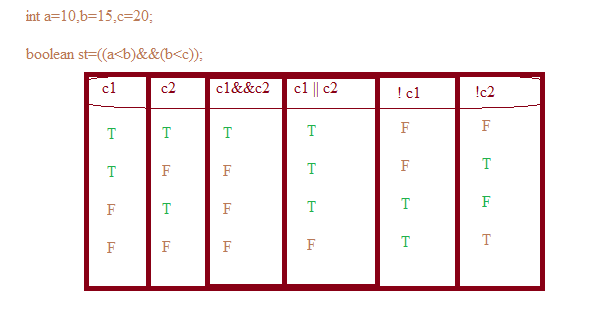


Logical Operators:

&& , || , !

At a time to check more than one condition then we should go for logical operators.

Truth Table:



8) instanceof is key word it act as a operators

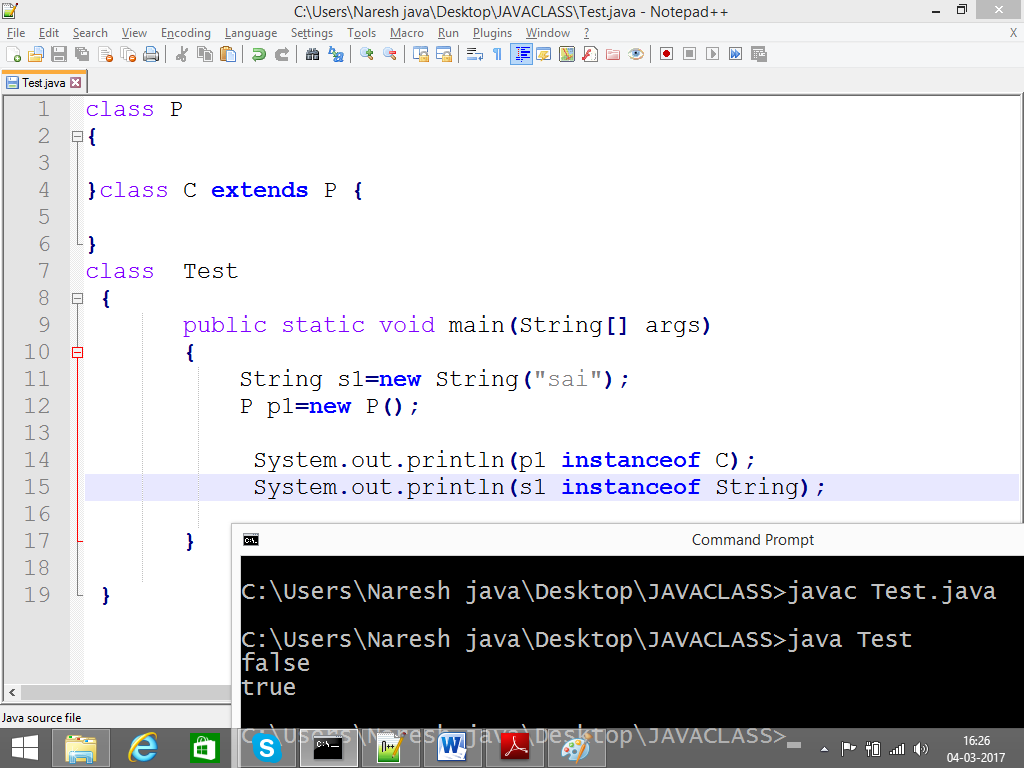
It is used to whether the object references is to particular class are not ?

Int a=10;

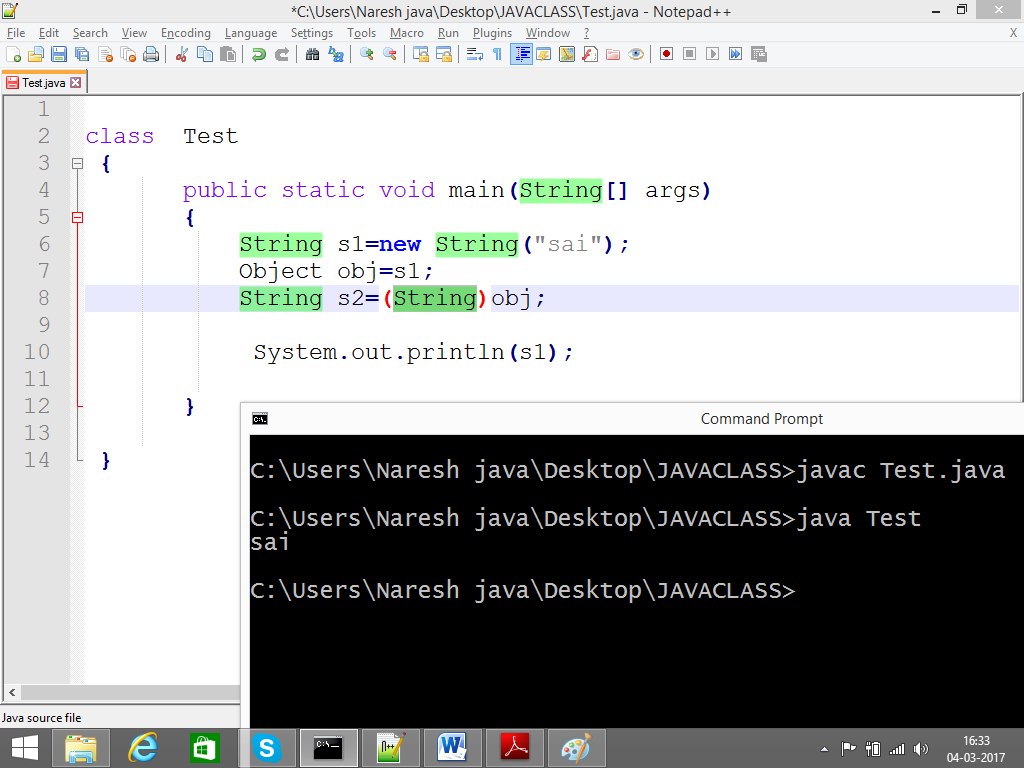
Float b=(float)a;

System.out.println(a); //10

System.out.println(b);//10.00

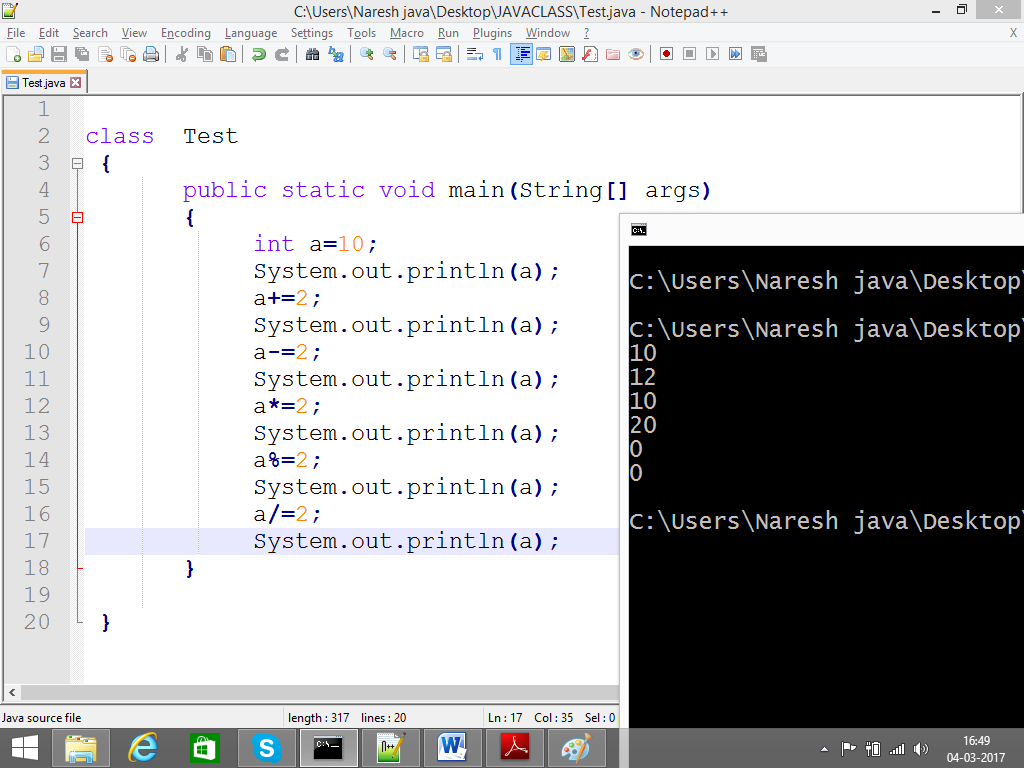


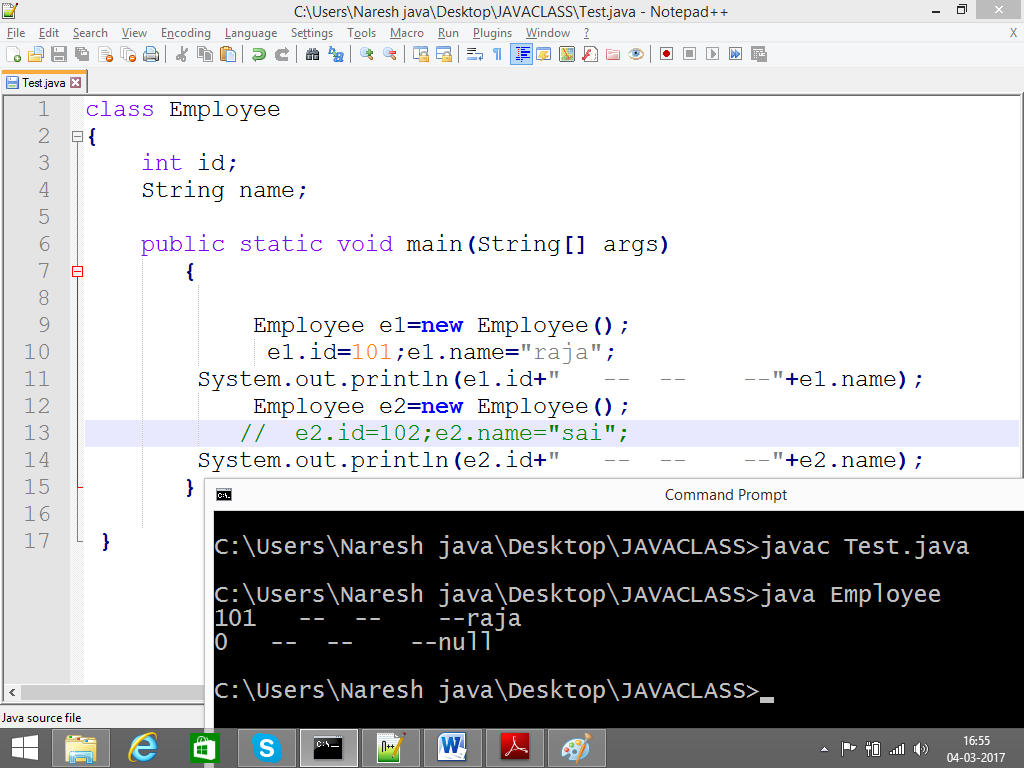
9) type cast operators



10) assignment operator

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



12) new operator 

13) Dimensional ‘[]’ operator is used to make a partition to the memory (Unit)

Collection of similar elements (array)

Int a=10,20,30/ drawback int a=10,b=20...

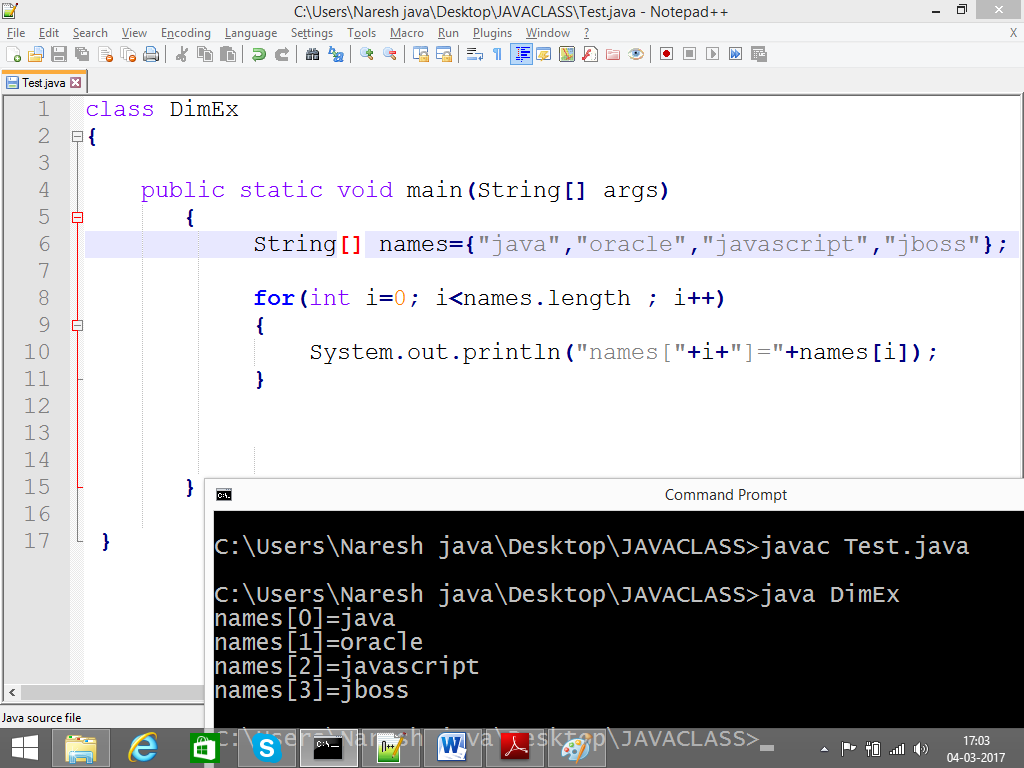
Arrays always start with zero

Array: int a[5]={10, 20, 30, 40, 50}

0-4 a[0] a[1] a[2]a[3]a[4]

Int a=10;

Float b=a;



14) Precedence of java operators

( ), \* , % / , + - …etc

**FLOW CONTROL**

**Introduction**

Flow control describes the order in which all the statements will execute at run time.

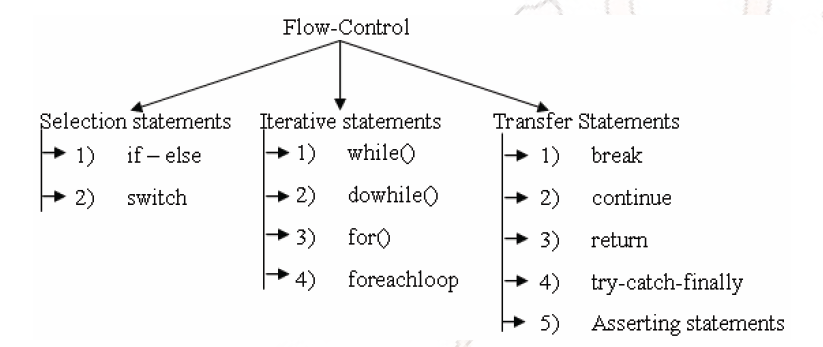
Flow controls are categorized into 3 types those are

1) Selection statements

2) Iterative statements

3) Transfer statements

The following fig will explains the flow controls categories

Syntax of IF:

If(condition){

}

If(cndition){

}

Else{

}

Nested if else

If()

If(){

}else{

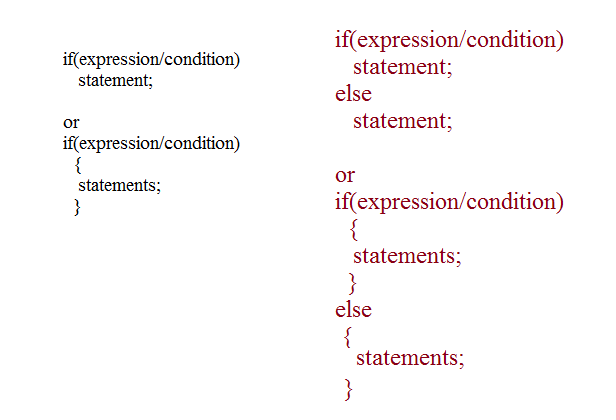
}

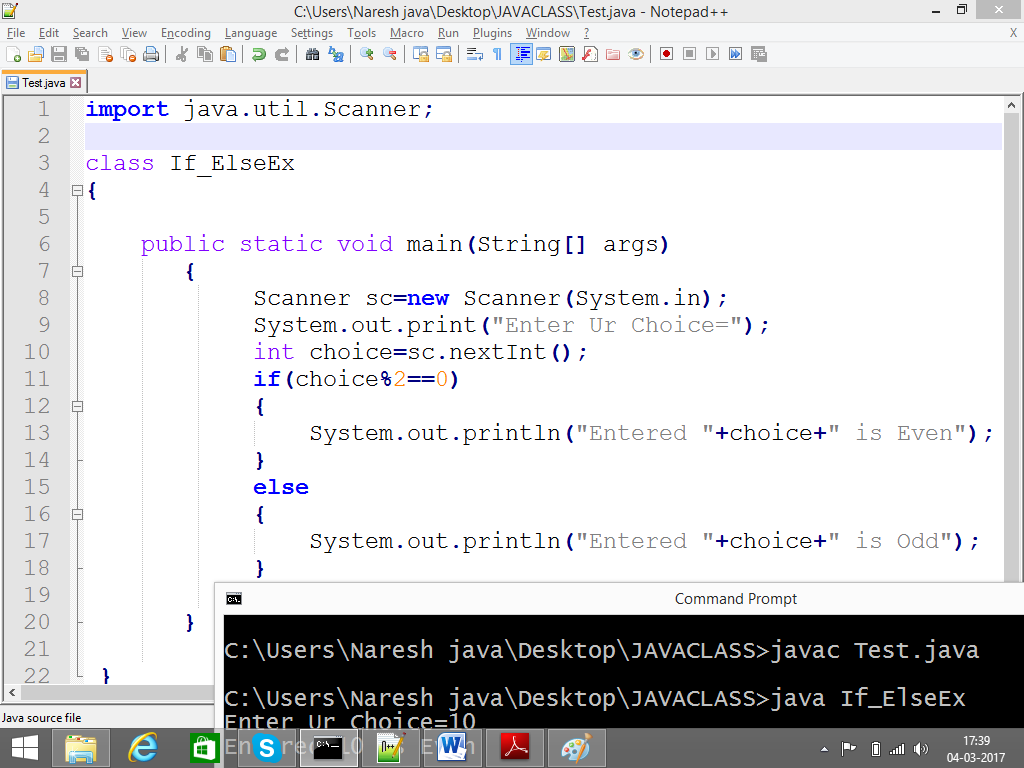
}

Else

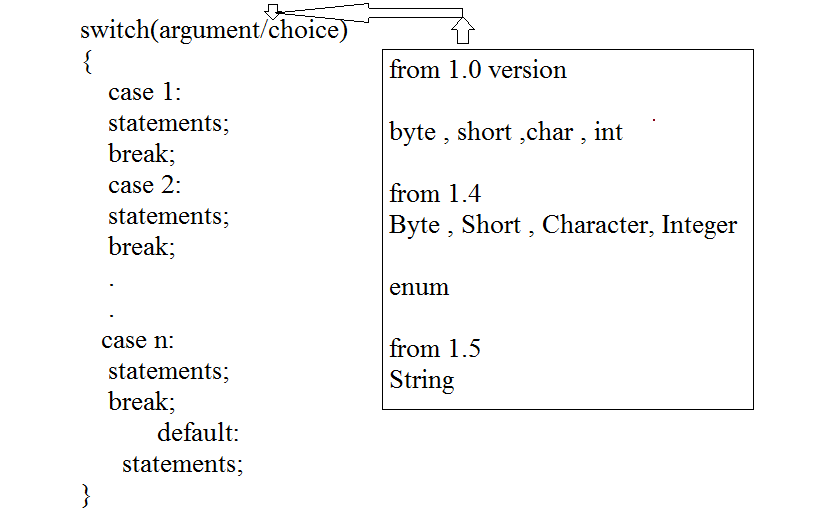
{

}

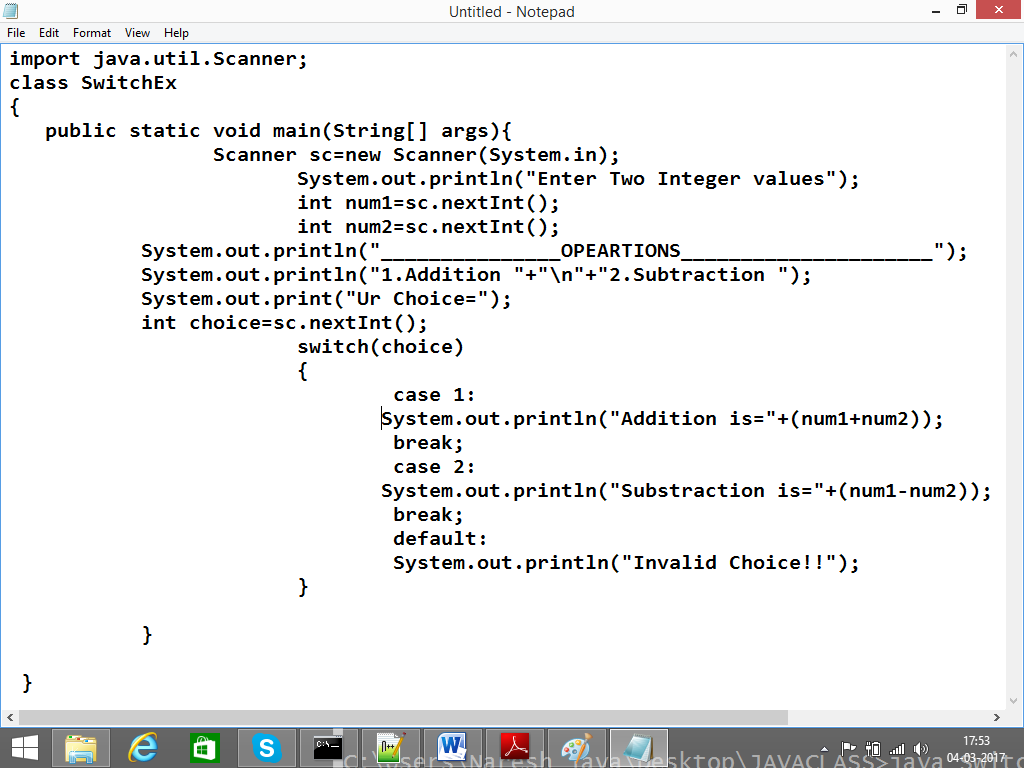


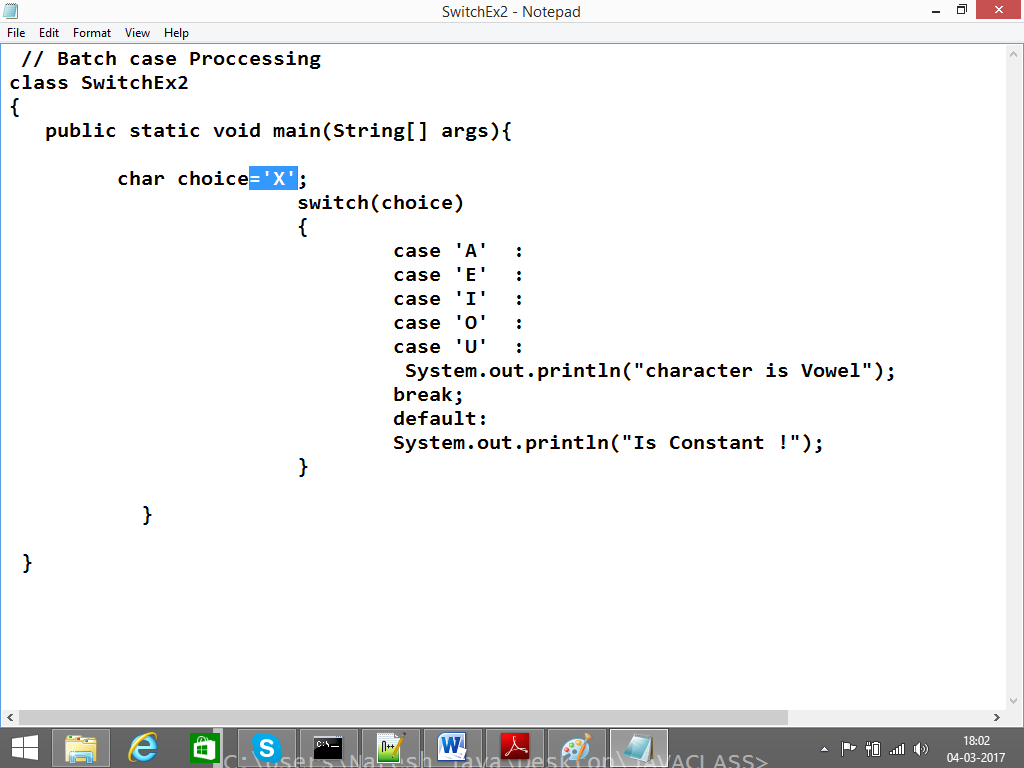


SWITCH statement:

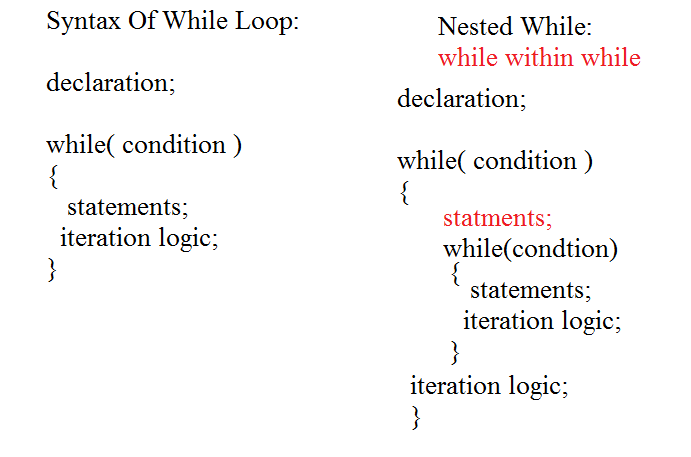


Example:

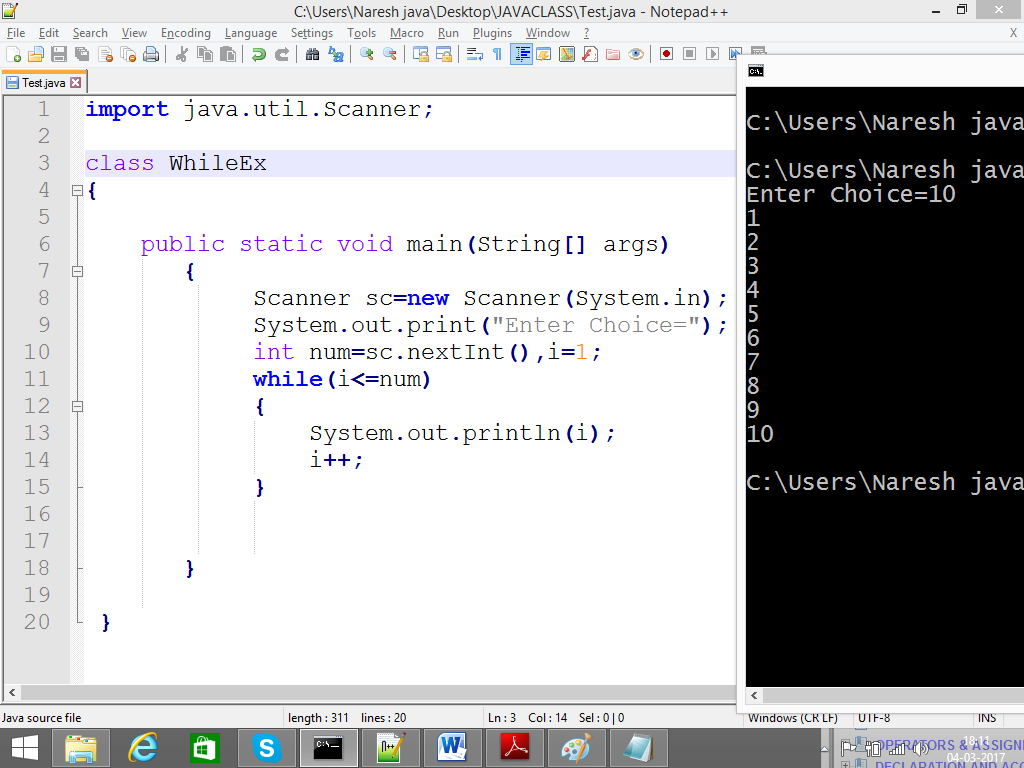


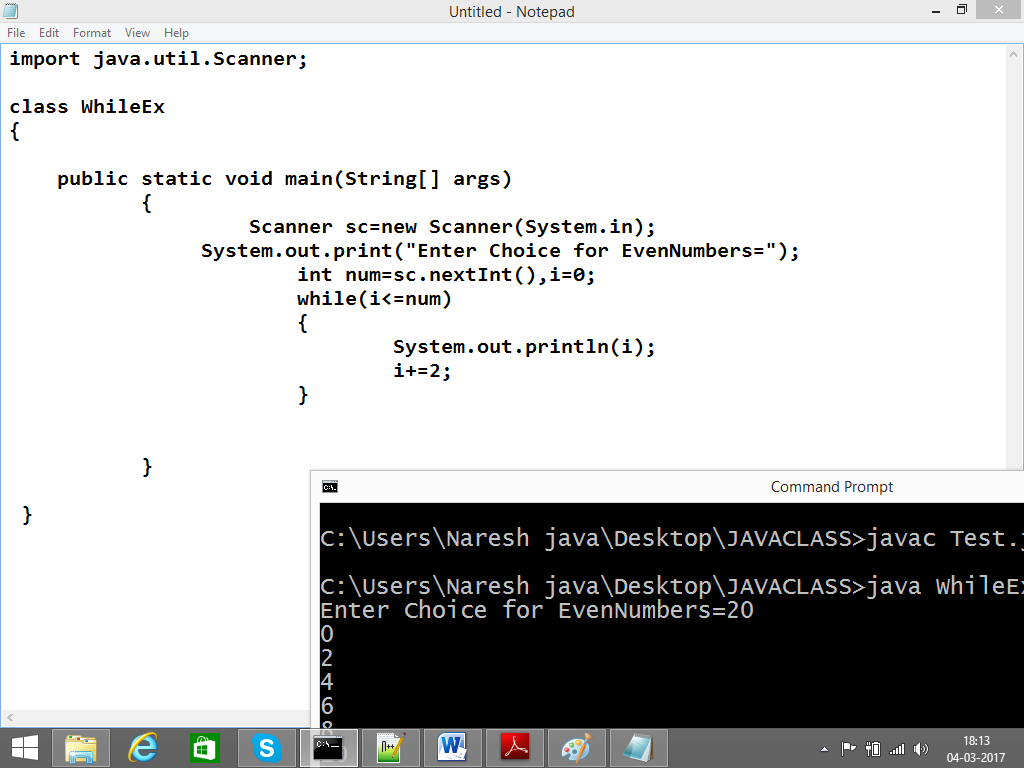


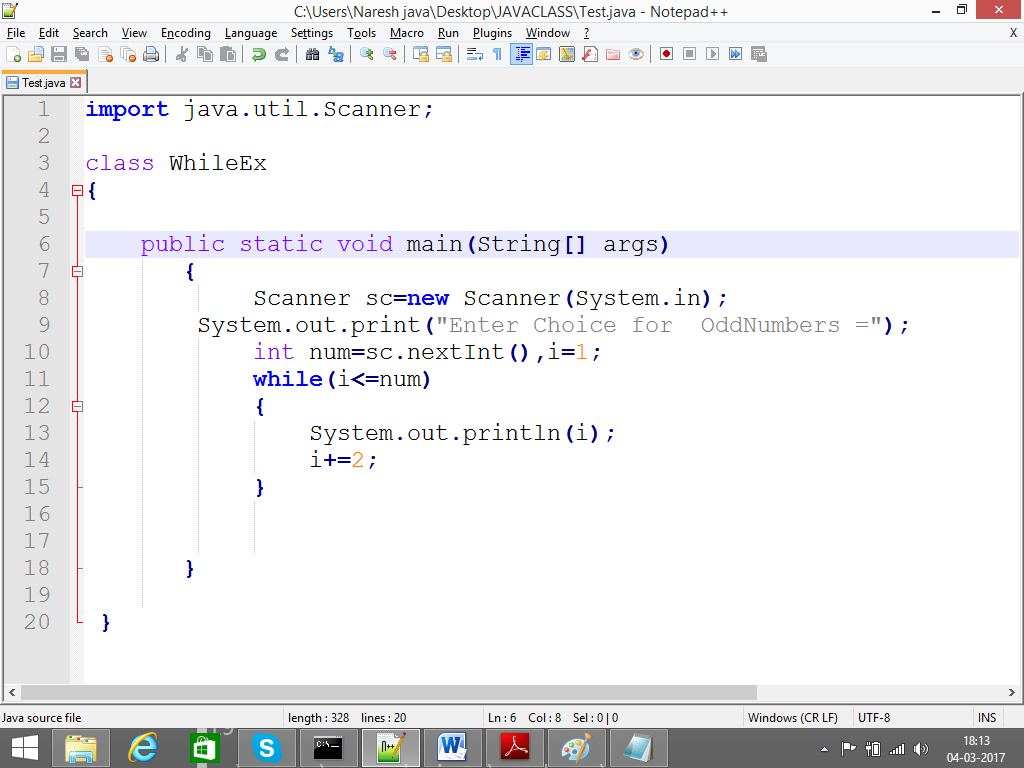
While Loop:

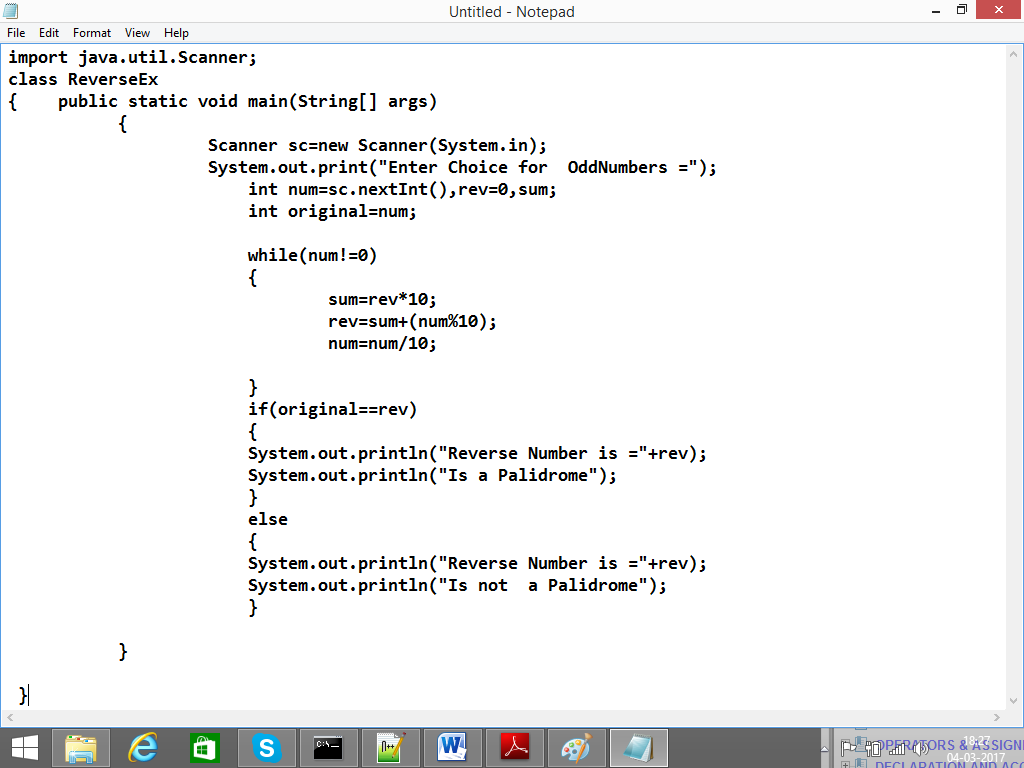


Example:

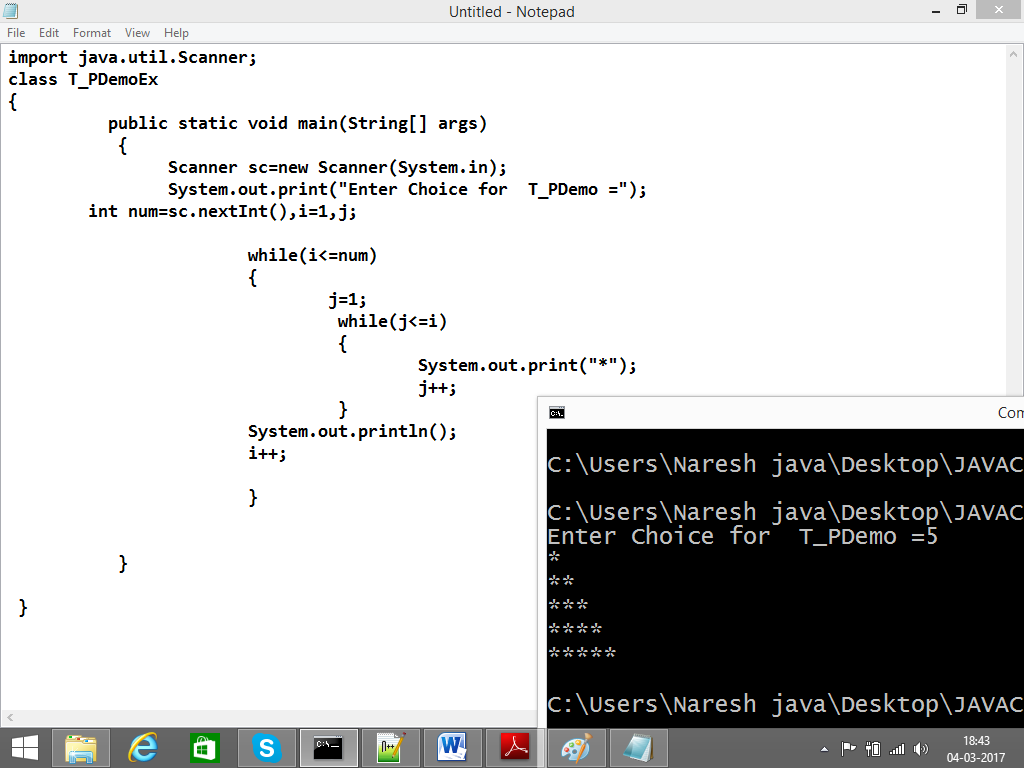




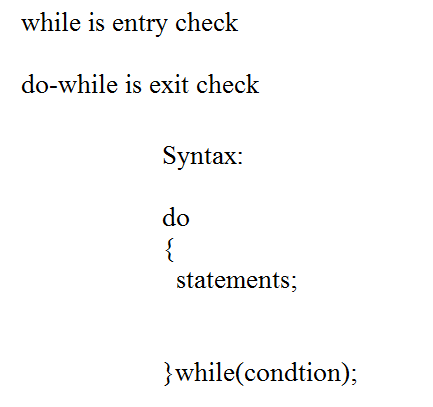




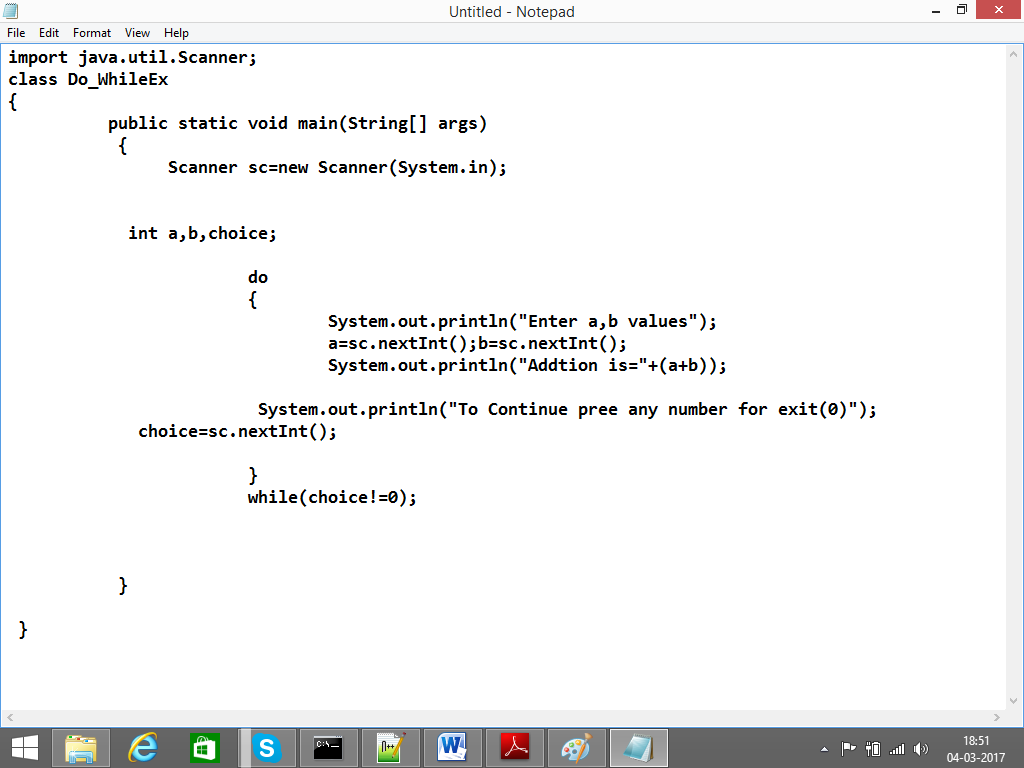
Nested While Examples:



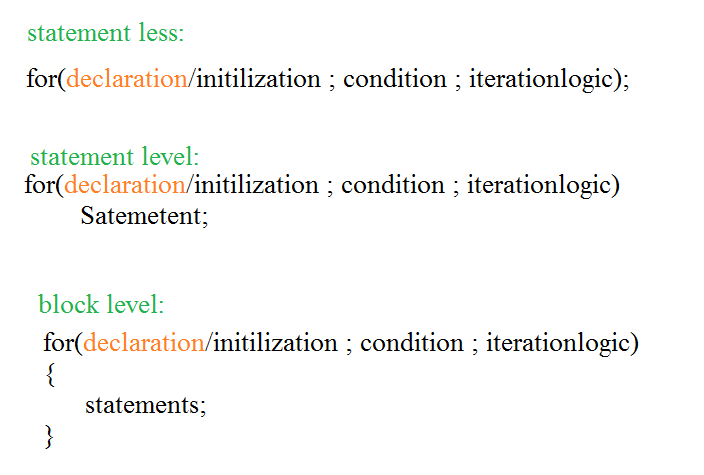
Do-While:



Example :

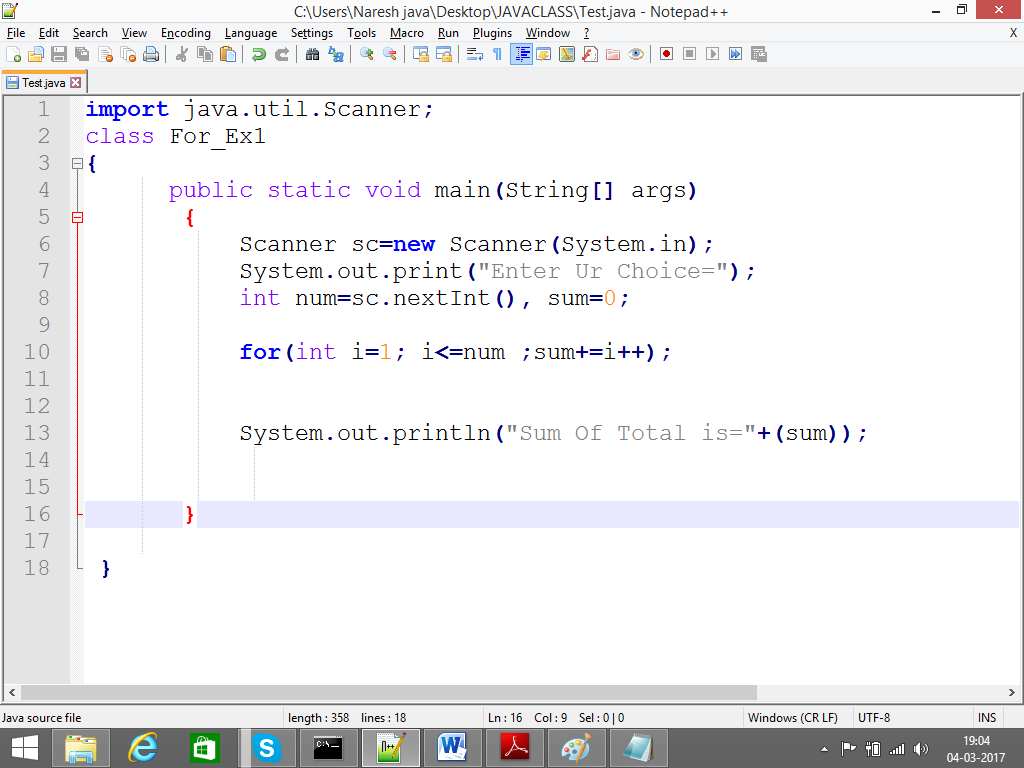


For Loop:

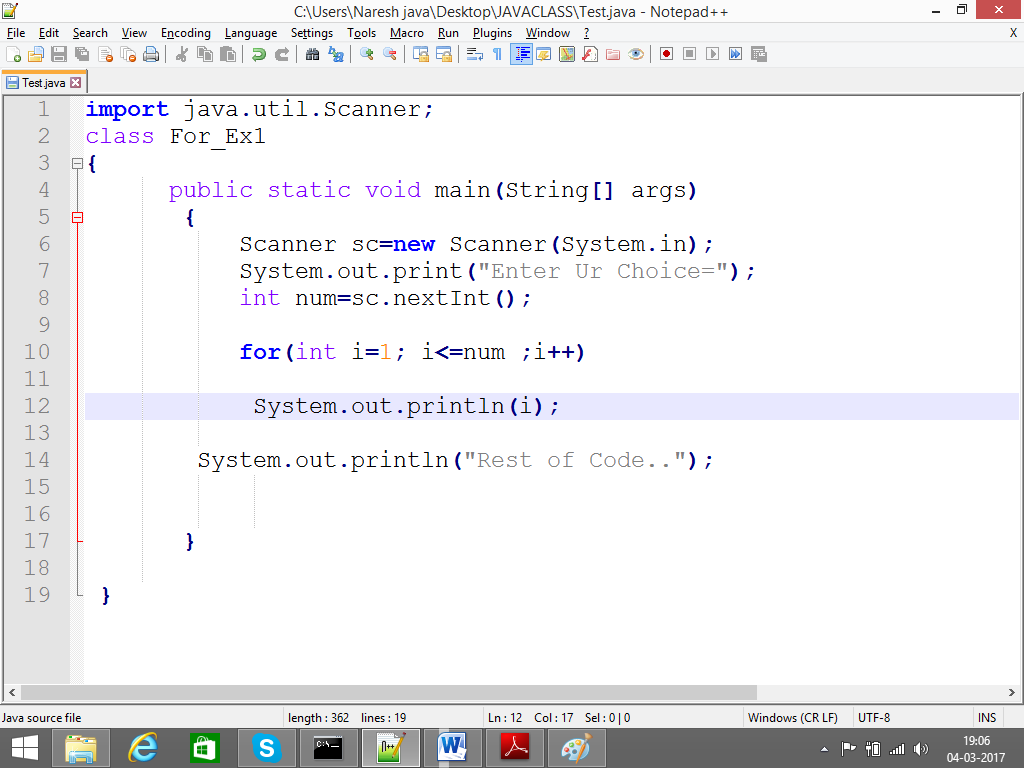


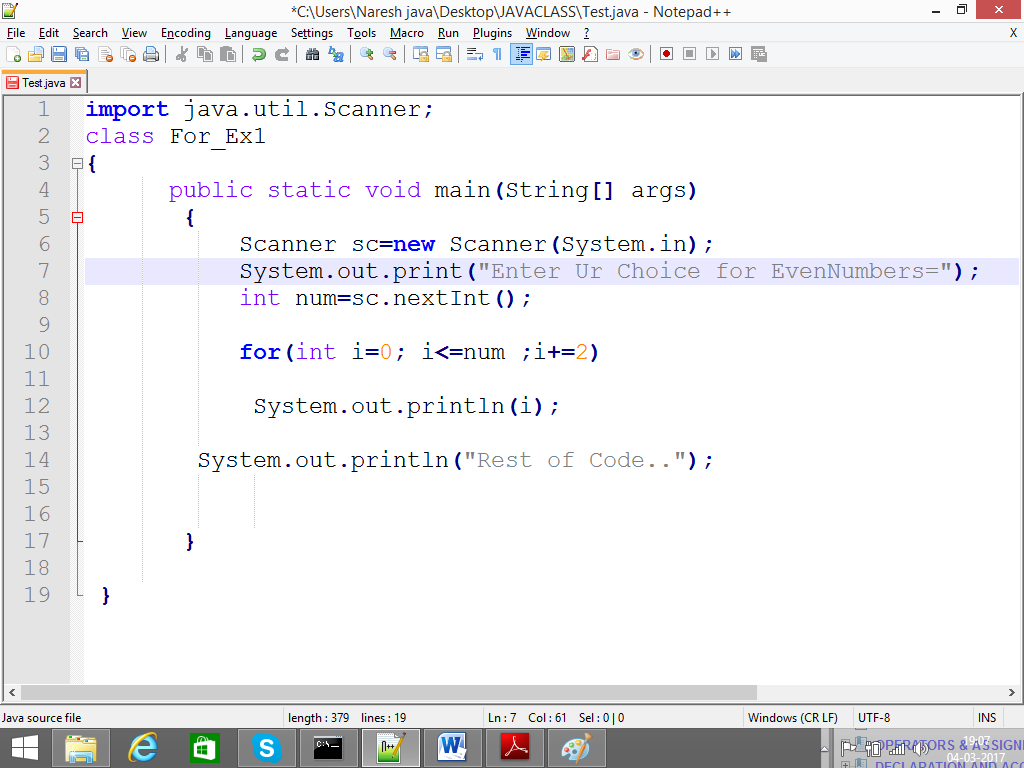
Examples: statement less

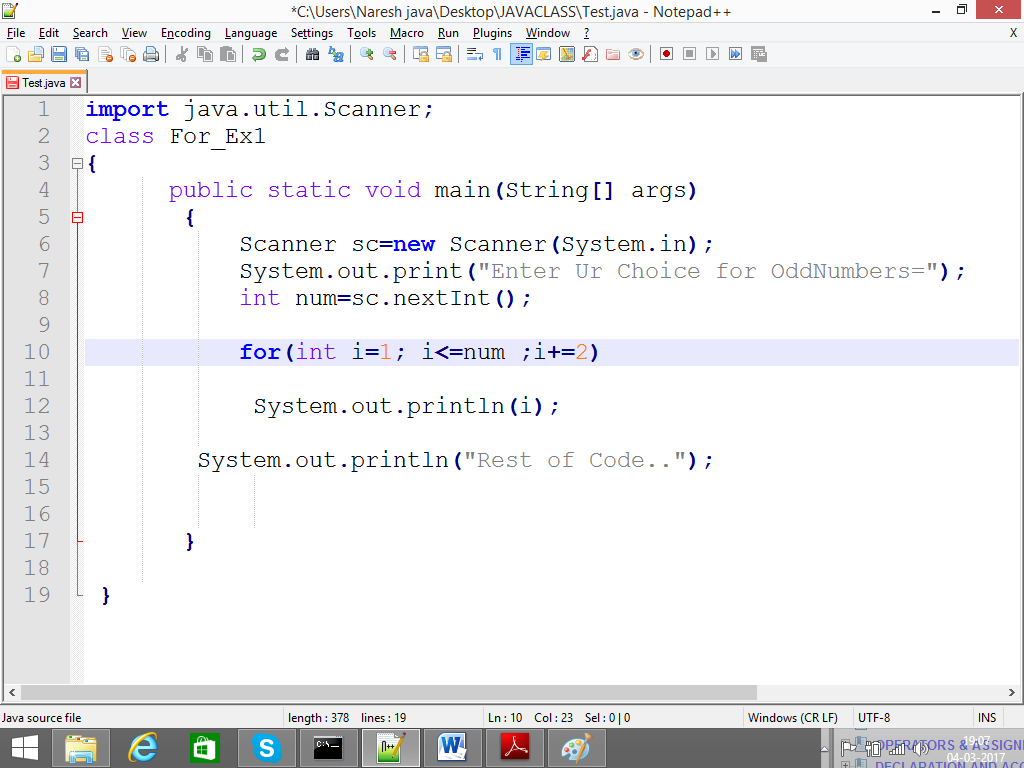
class



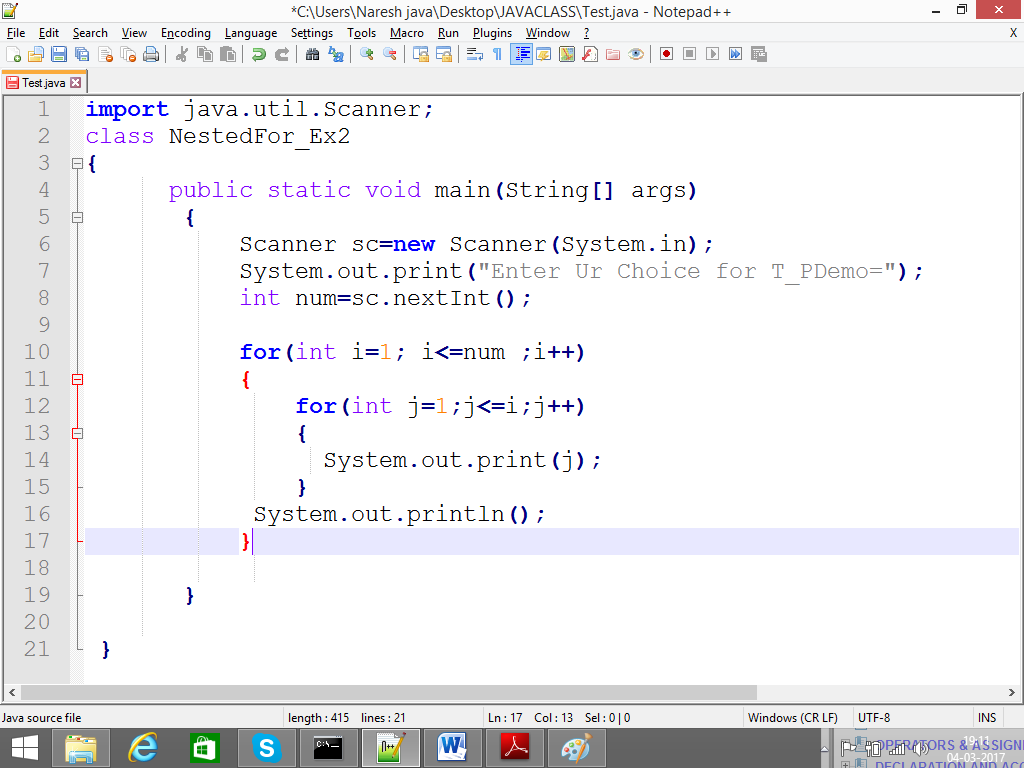
Statement level:





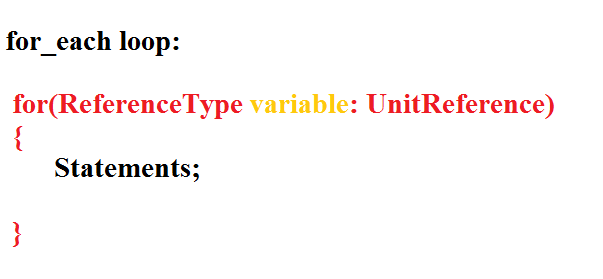


Nested For\_Loop:

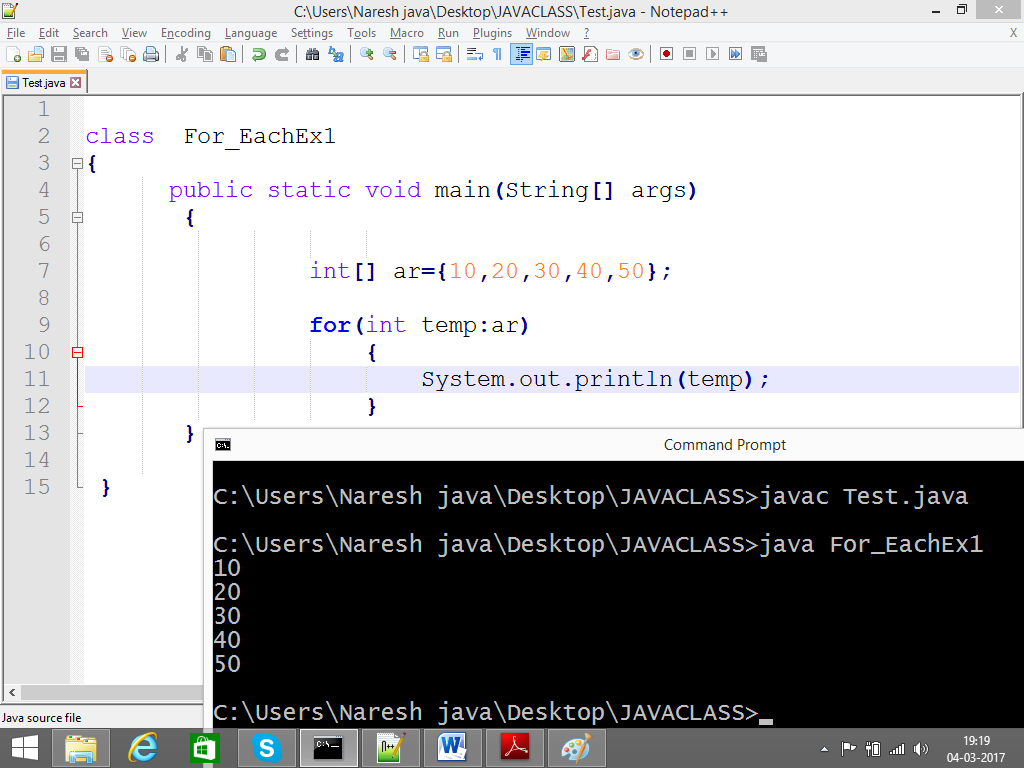


For Each Loop: it is available from 1.5 versions

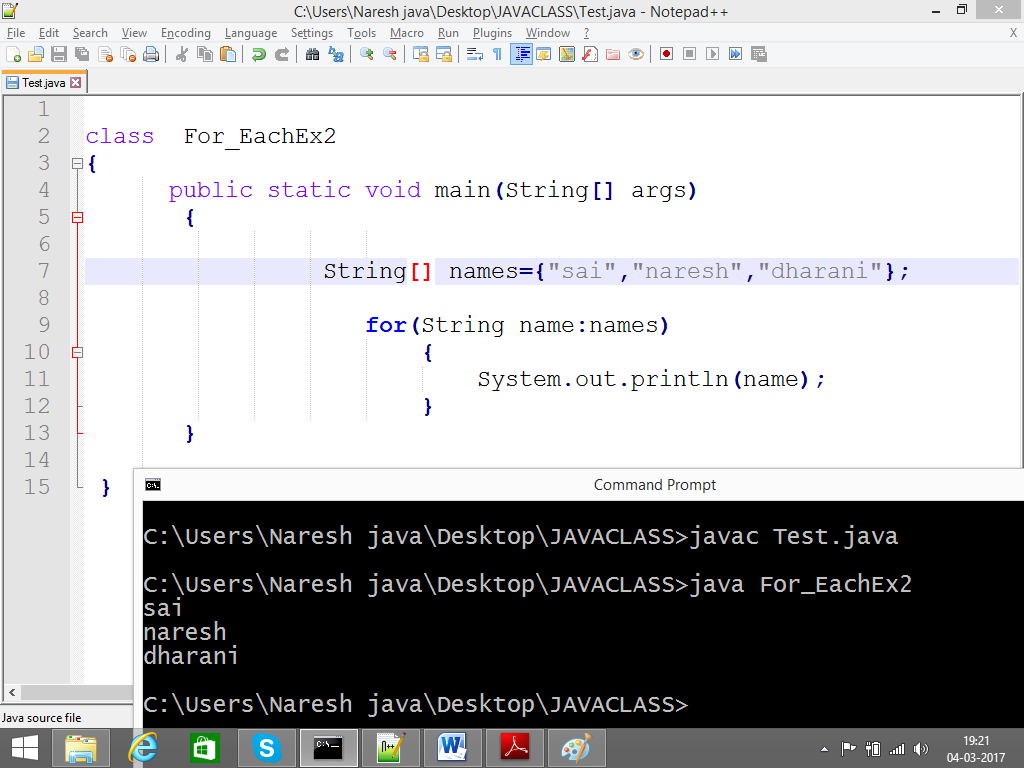
It is used to fetching / retrieving Unit (collection of elements) Elements



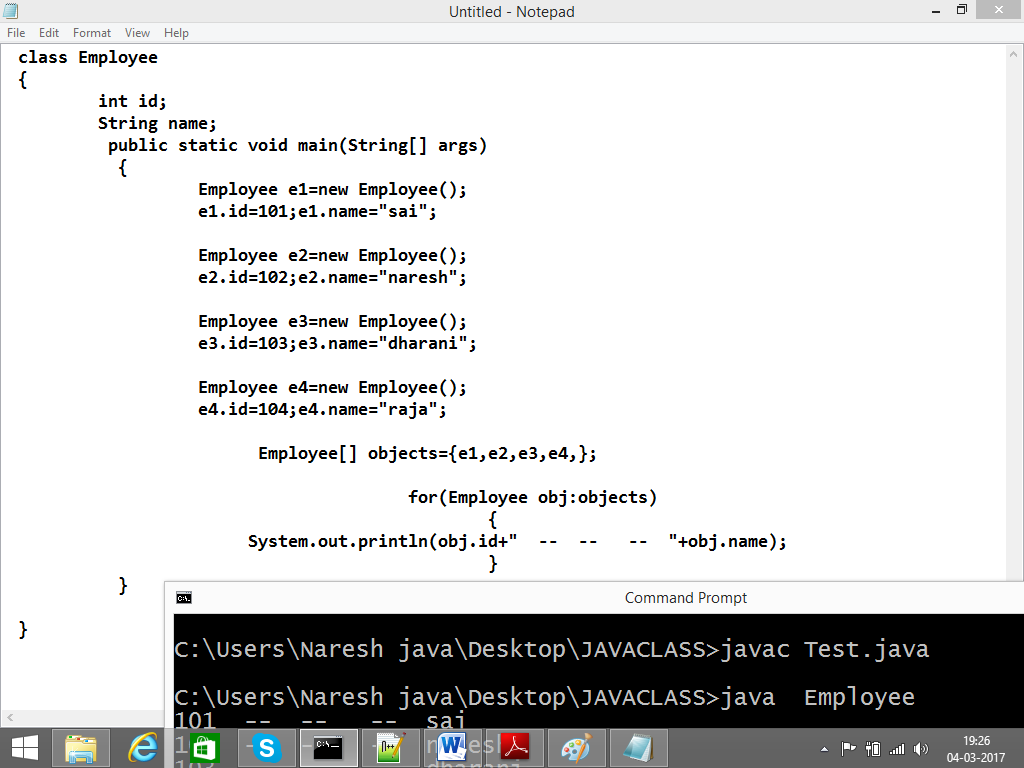
Example:



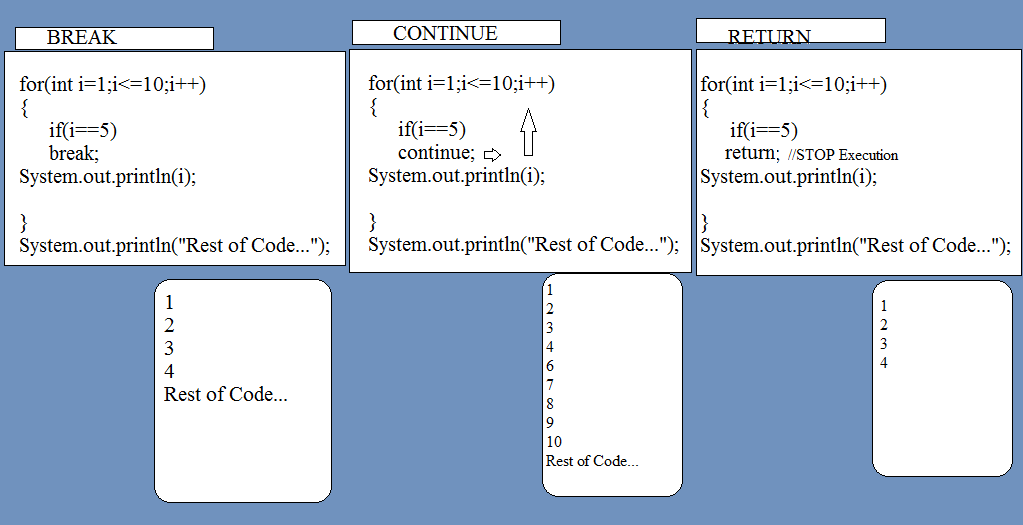
Example 2:



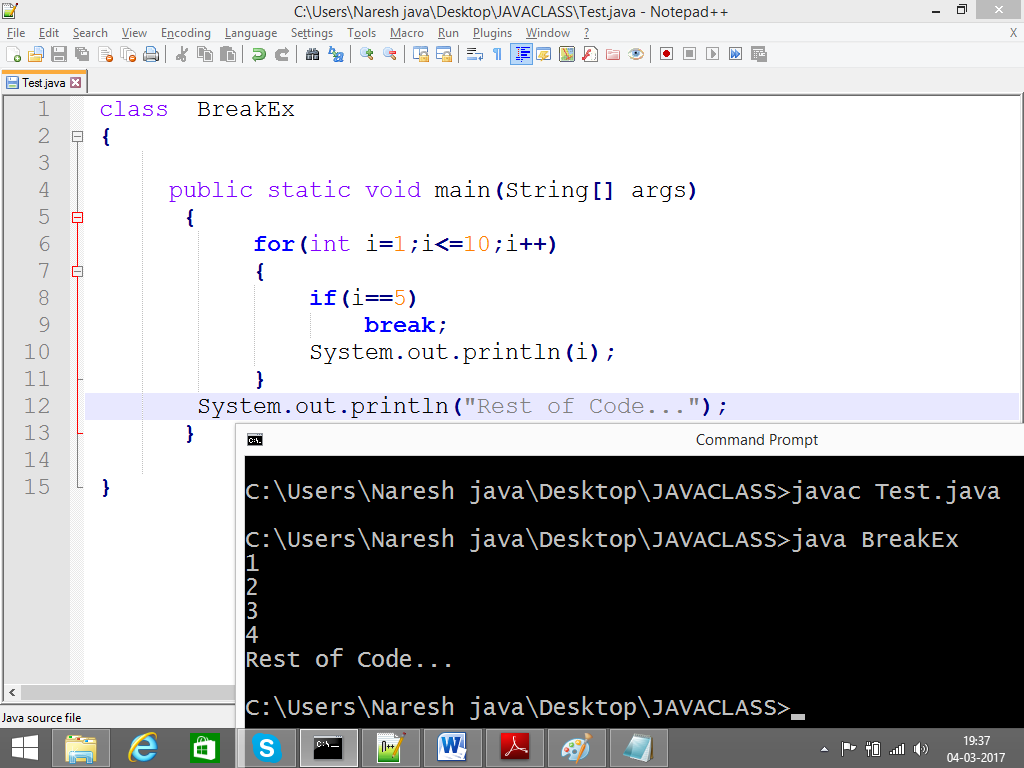
Example3:



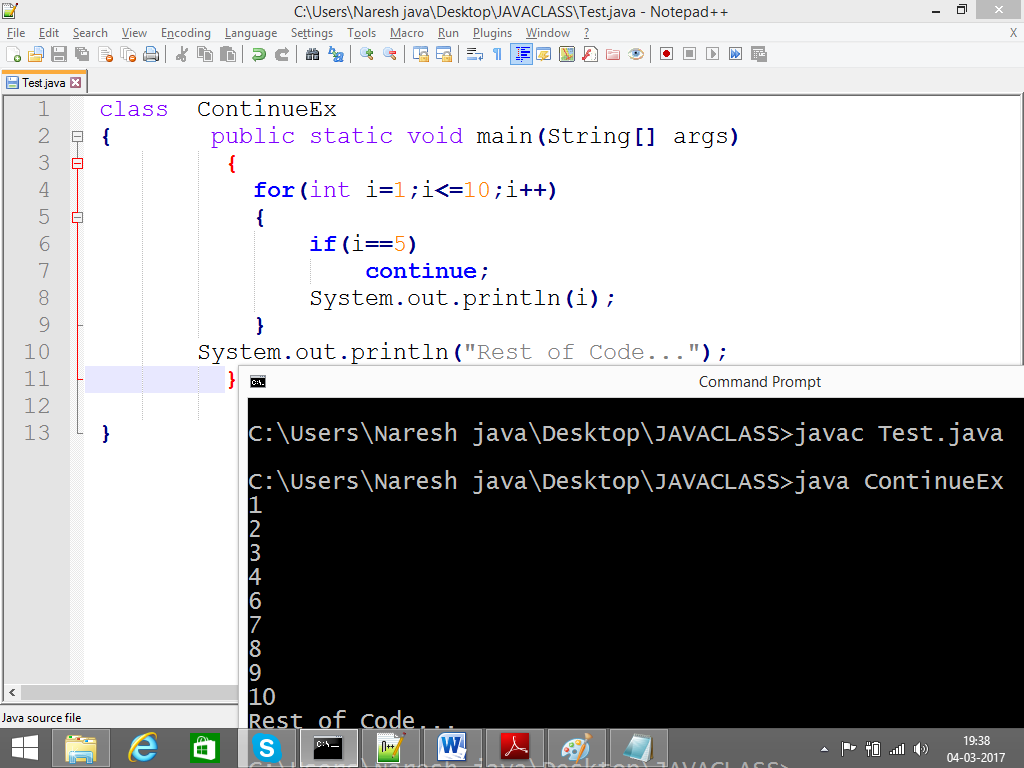
Transfer statements: break, continue, return, try/catch, assert



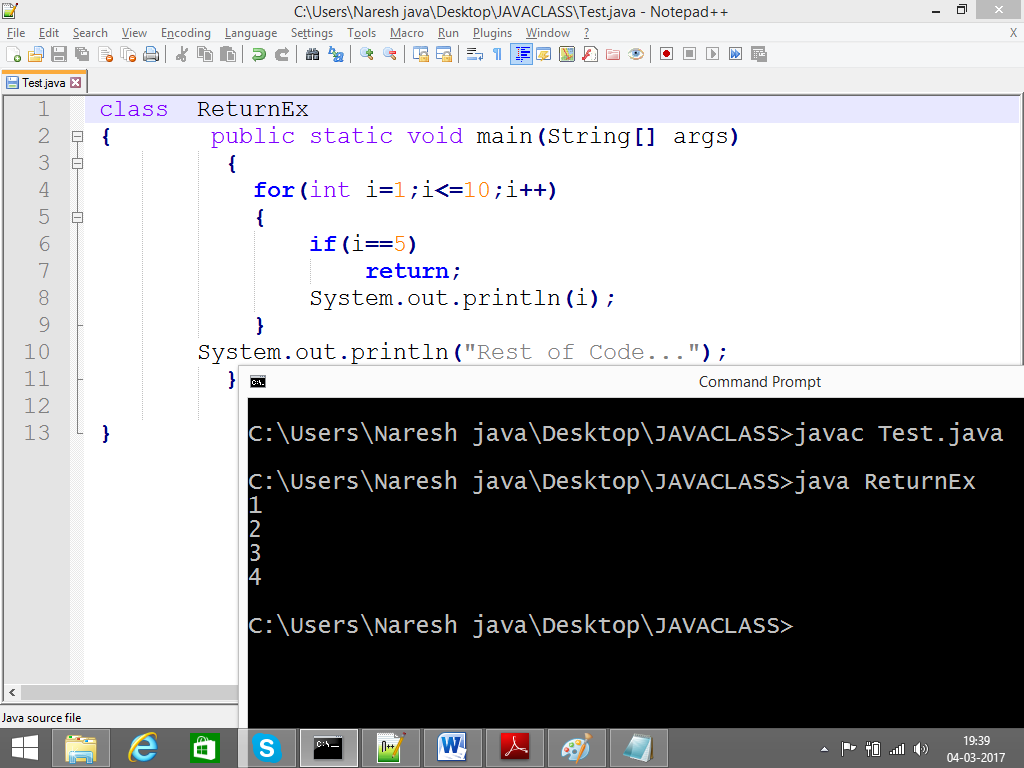
BREAK Example:



Continue Example:



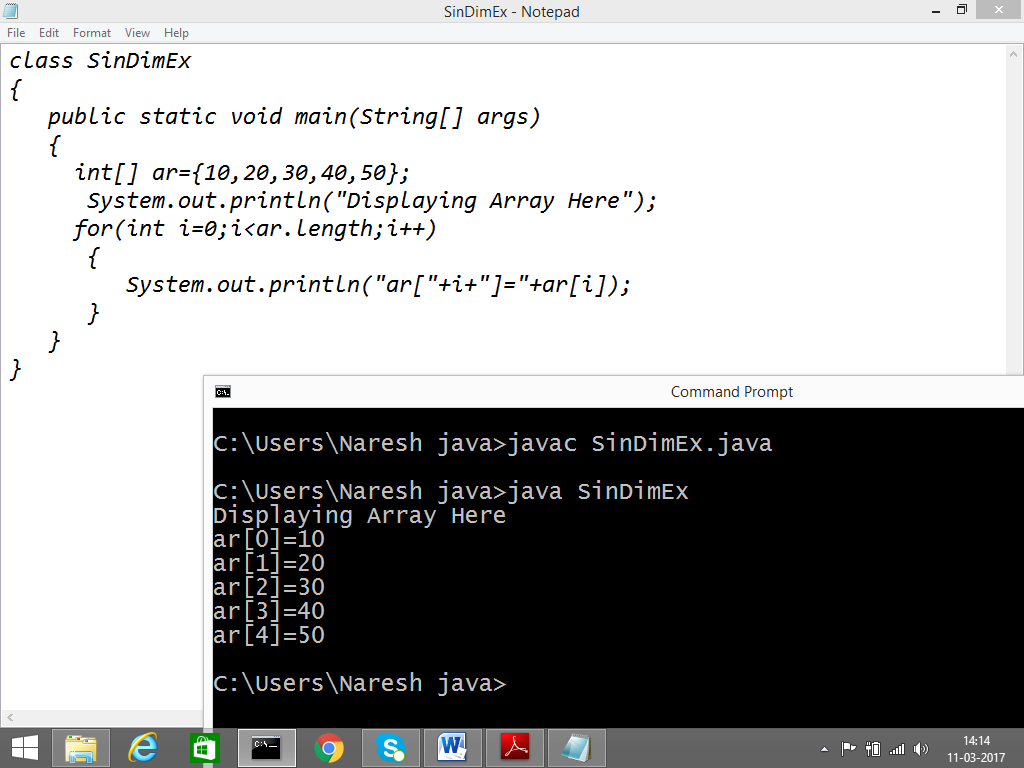
Return Example:



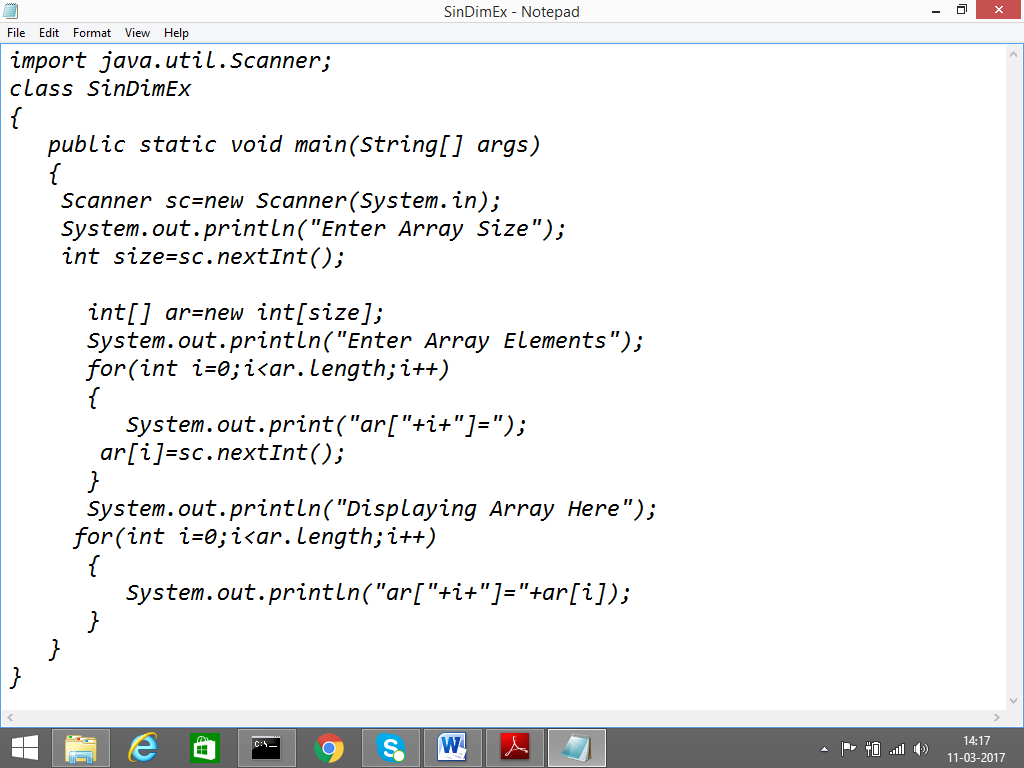
Thank You>

-------------------------------------------------------------

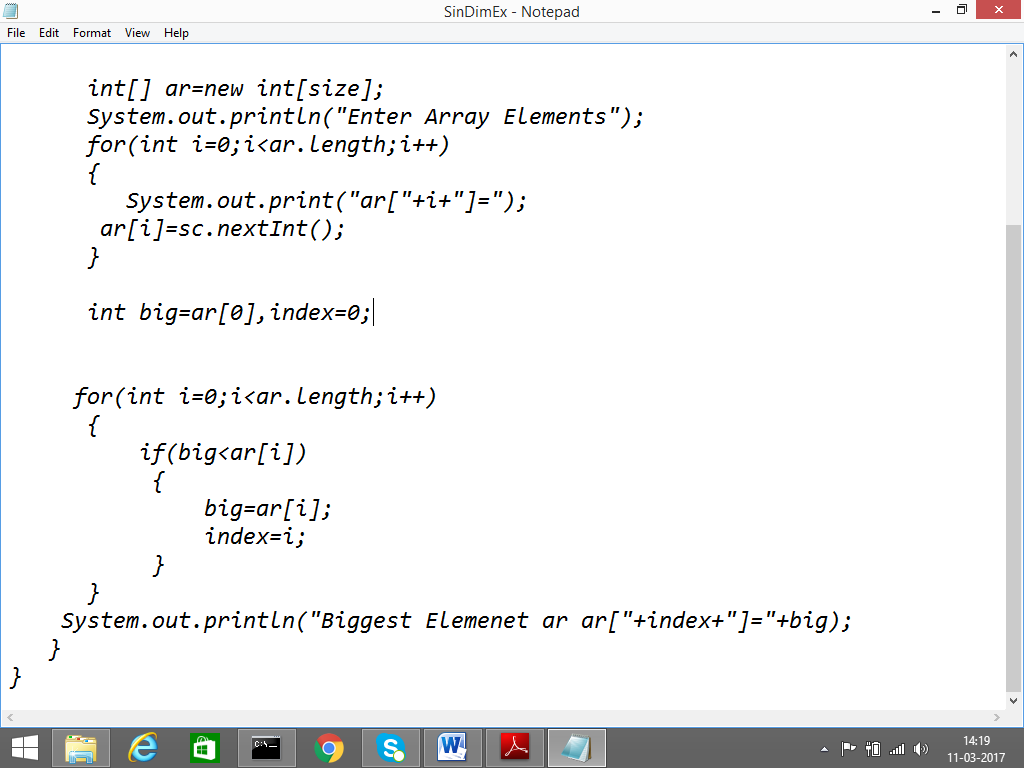
Arrays:



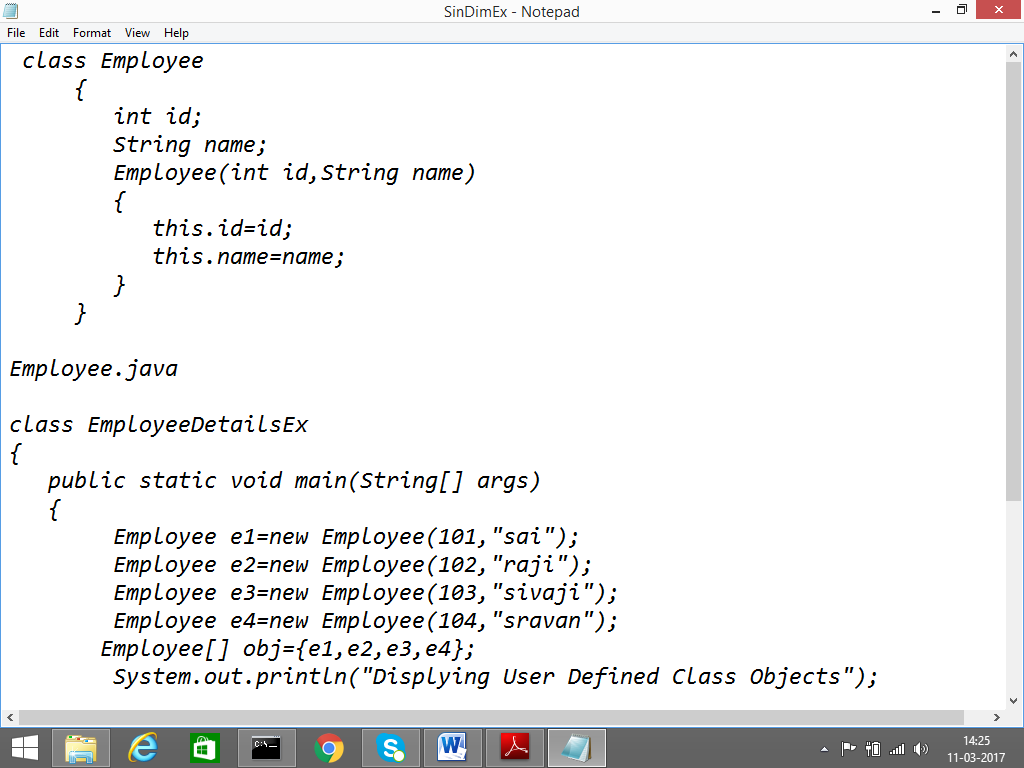
Way-2

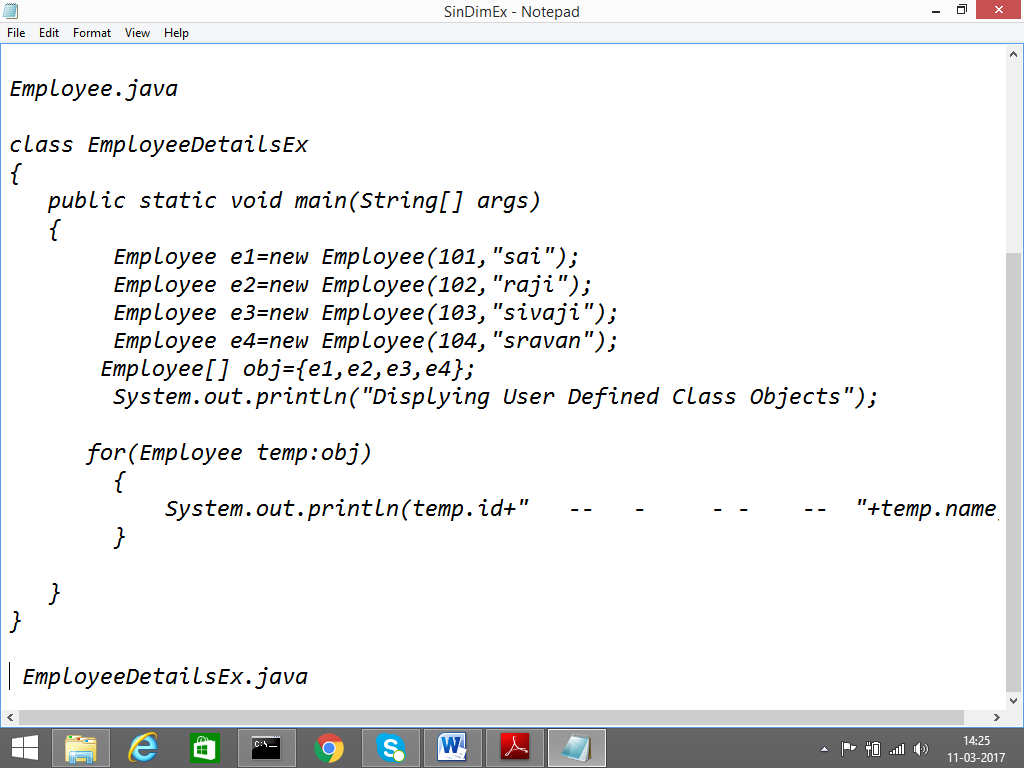


Find Big Element from given Array?

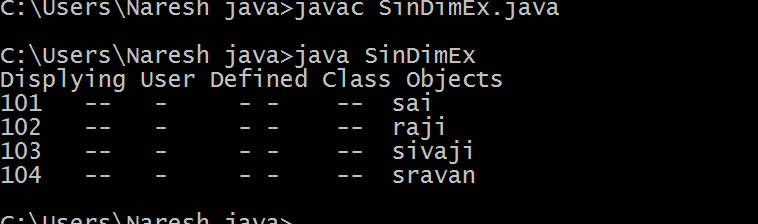


User Defined Array Displaying :



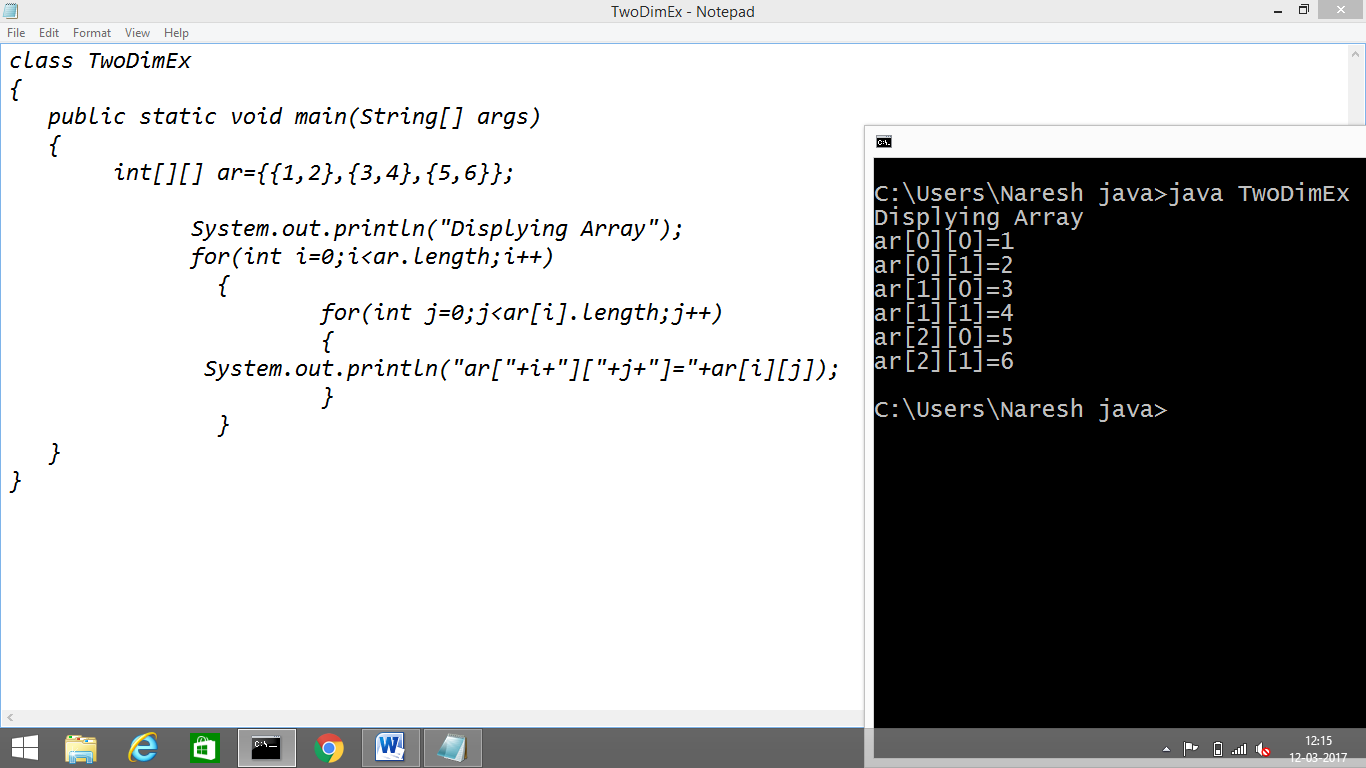


OutPut:

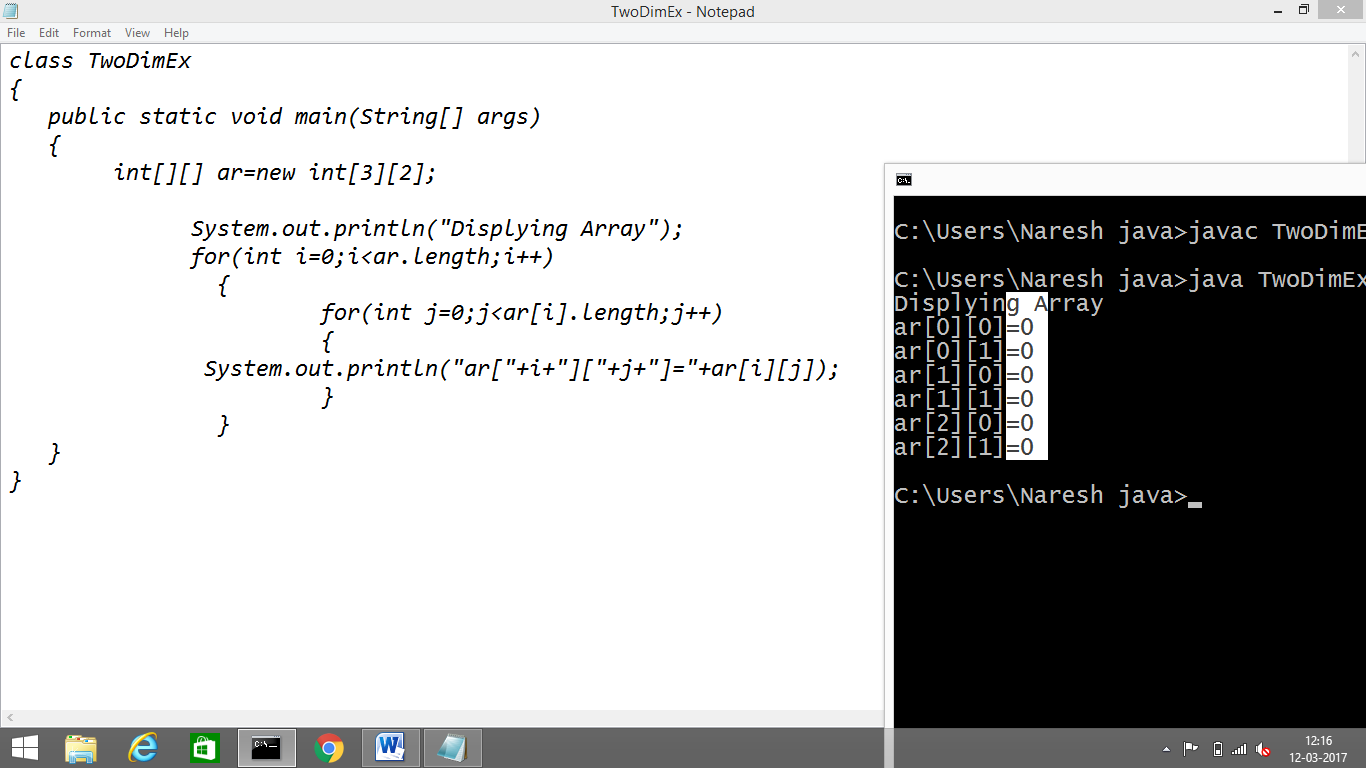


TWO DIM ARRAY:

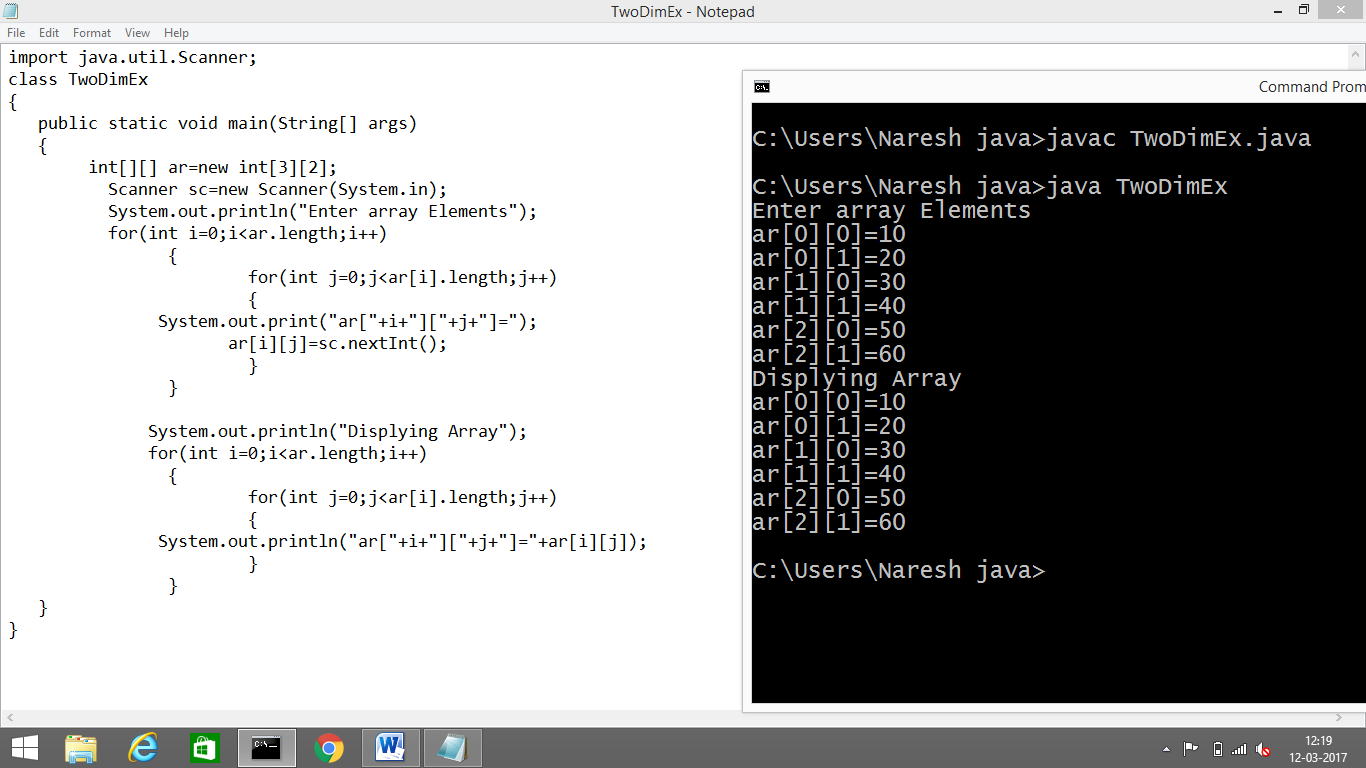
Static approach:



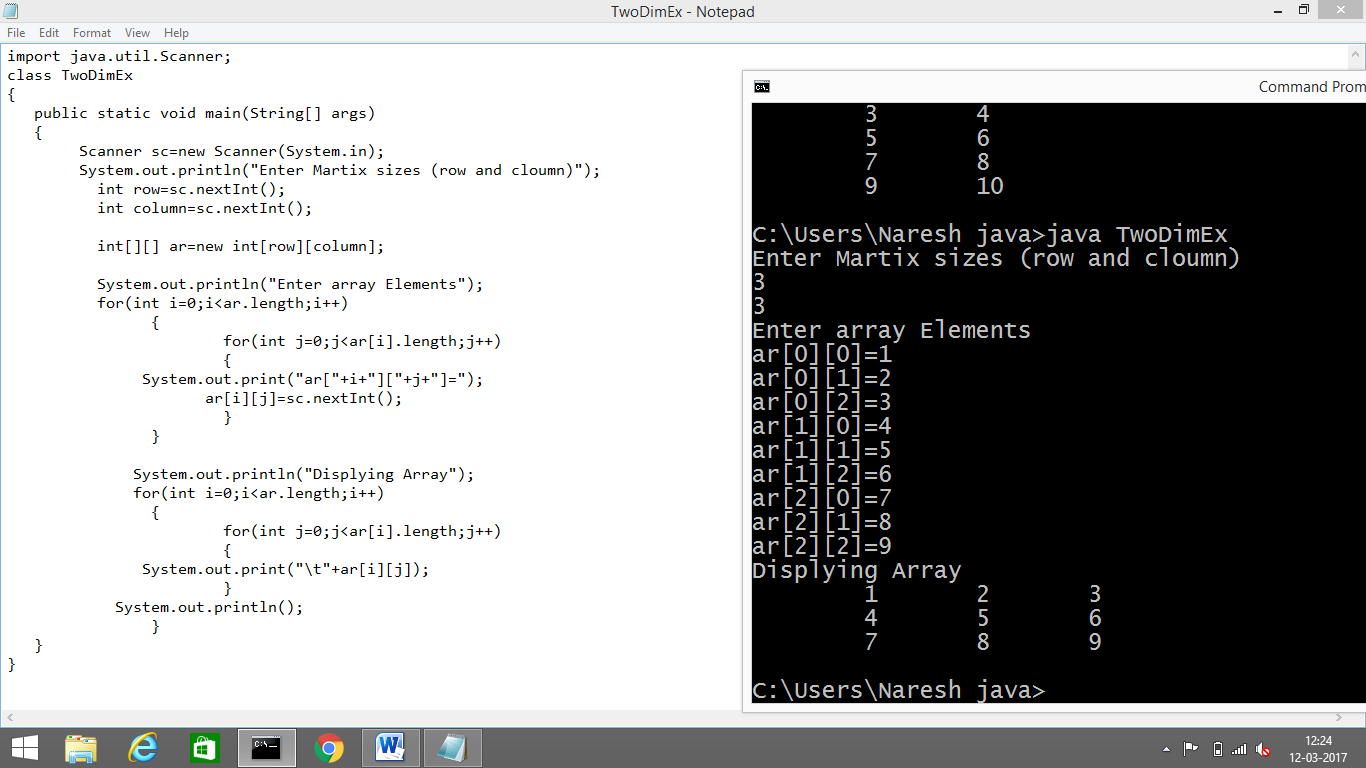
Dynamic approach:



Fill The Elements into array by Using Scanner:

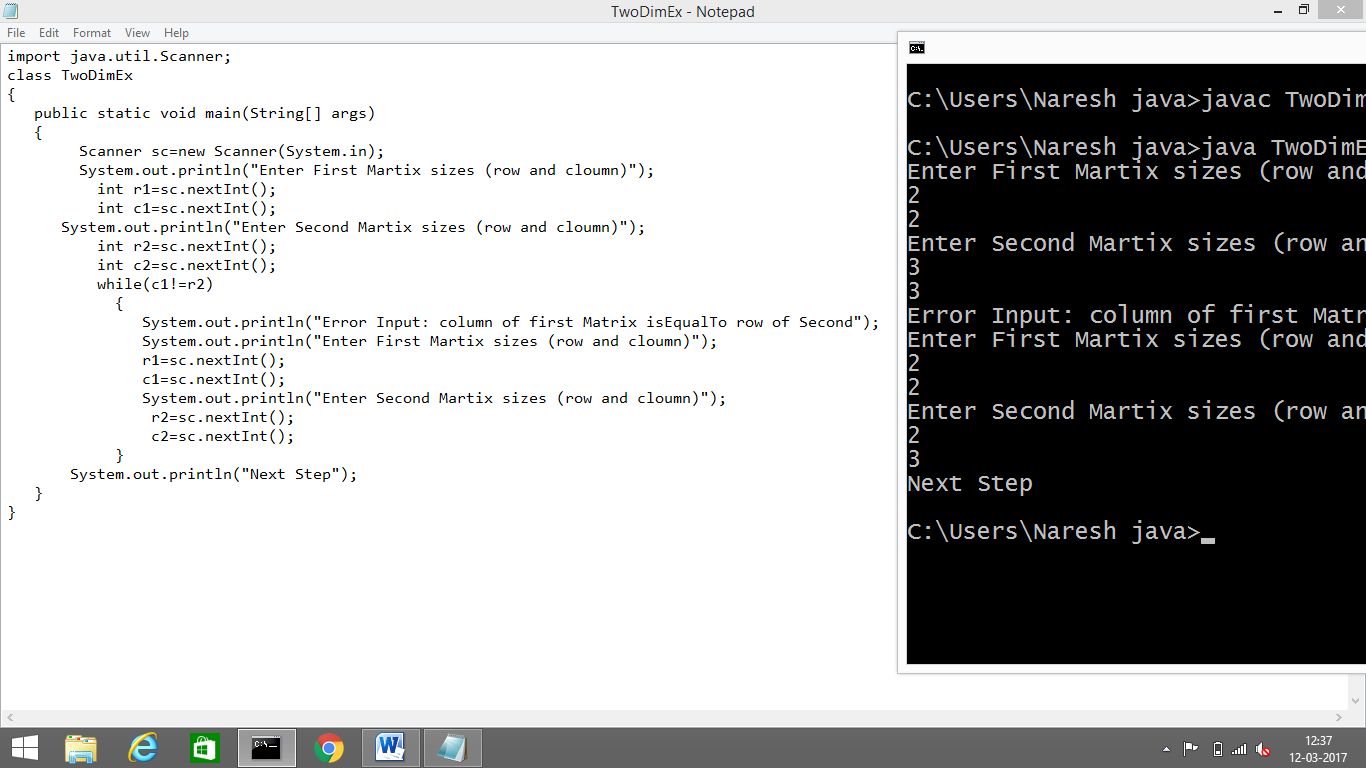


Array Matrix Displaying:



Matrix Multiplication Steps:

Step:1: declaration formaula



Declaration of Matrix :

Multiplication with formula : Rule 1-Column of First matrix is equal to row of second matrix

Rule 2 for multiplication: 1st row \* 1 column of second matrix

import java.util.Scanner;

class TwoDimEx

{

public static void main(String[] args)

{

Scanner sc=new Scanner(System.in);

System.out.println("Enter First Martix sizes (row and cloumn)");

int r1=sc.nextInt();

int c1=sc.nextInt();

System.out.println("Enter Second Martix sizes (row and cloumn)");

int r2=sc.nextInt();

int c2=sc.nextInt();

while(c1!=r2)

{

System.out.println("Error Input: column of first Matrix isEqualTo row of Second");

System.out.println("Enter First Martix sizes (row and cloumn)");

r1=sc.nextInt();

c1=sc.nextInt();

System.out.println("Enter Second Martix sizes (row and cloumn)");

r2=sc.nextInt();

c2=sc.nextInt();

}

int[][] first=new int[r1][c1];

int[][] second=new int[r2][c2];

int[][] multi=new int[r1][c2];

System.out.println("Enter First Matrix Elements");

for(int i=0;i<r1;i++)

{

for(int j=0;j<c1;j++)

{

first[i][j]=sc.nextInt();

}

System.out.println();

}

System.out.println("Enter Second Matrix Elements");

for(int i=0;i<r2;i++)

{

for(int j=0;j<c2;j++)

{

second[i][j]=sc.nextInt();

}

System.out.println();

}

//formula

for(int i=0;i<r1;i++)

{

for(int j=0;j<c2;j++)

{

for(int k=0;k<c1;k++)

{

multi[i][j]+=first[i][k]\*second[k][j];

}

}

}

System.out.println("Displying Result Matrix");

for(int i=0;i<r1;i++)

{

for(int j=0;j<c2;j++)

{

System.out.print("\t"+multi[i][j]);

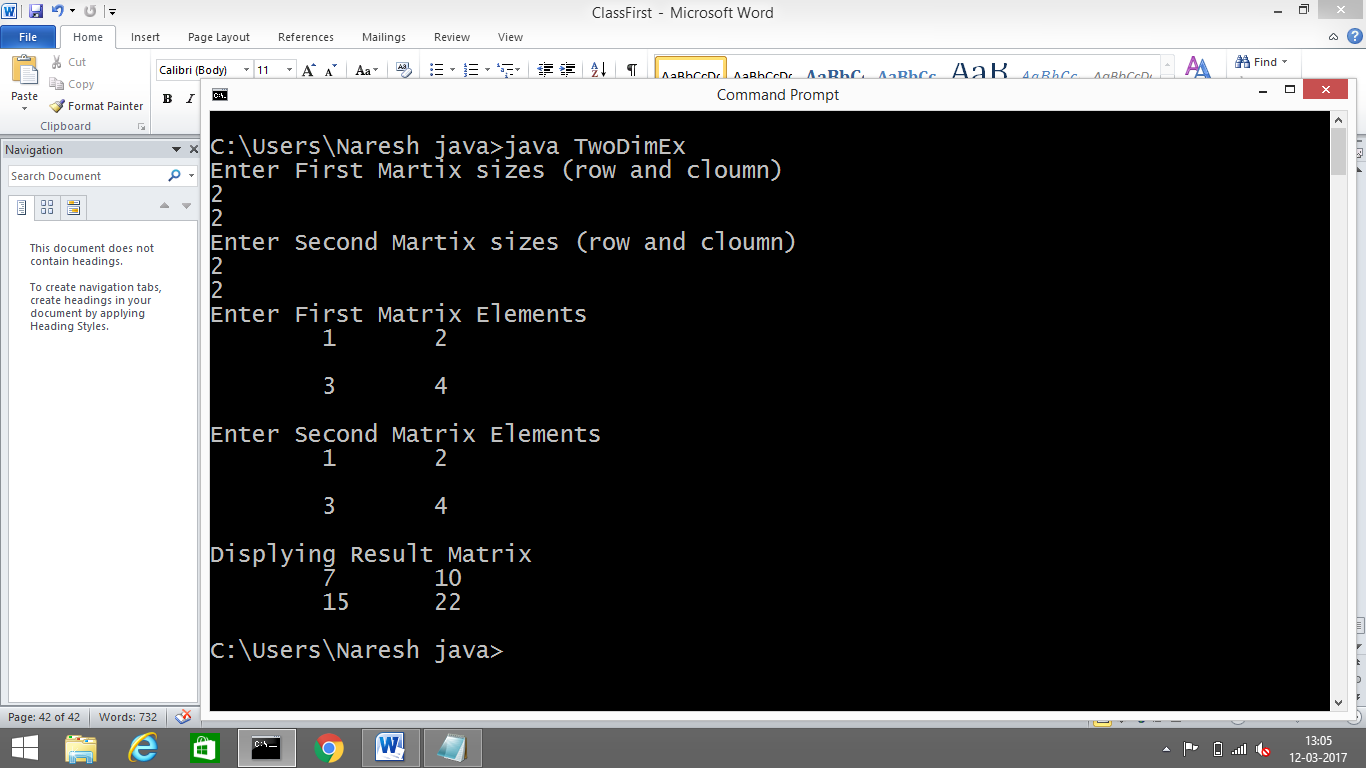
}

System.out.println();

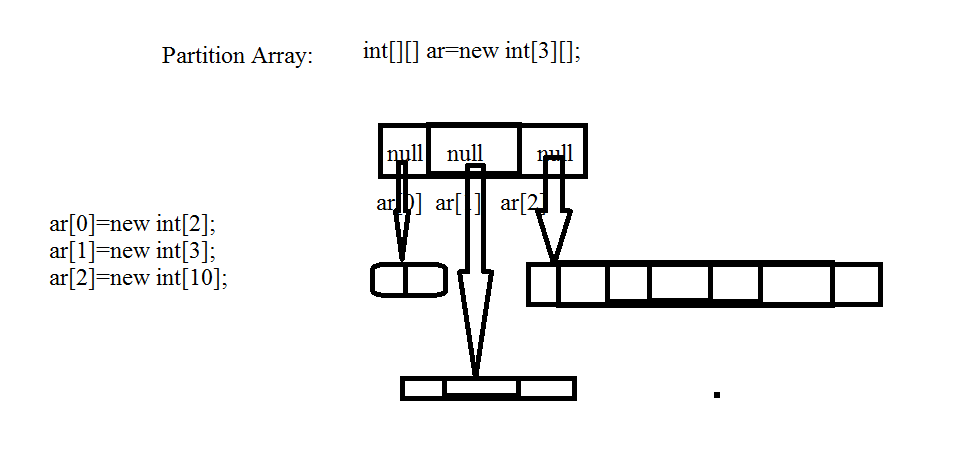
}

}

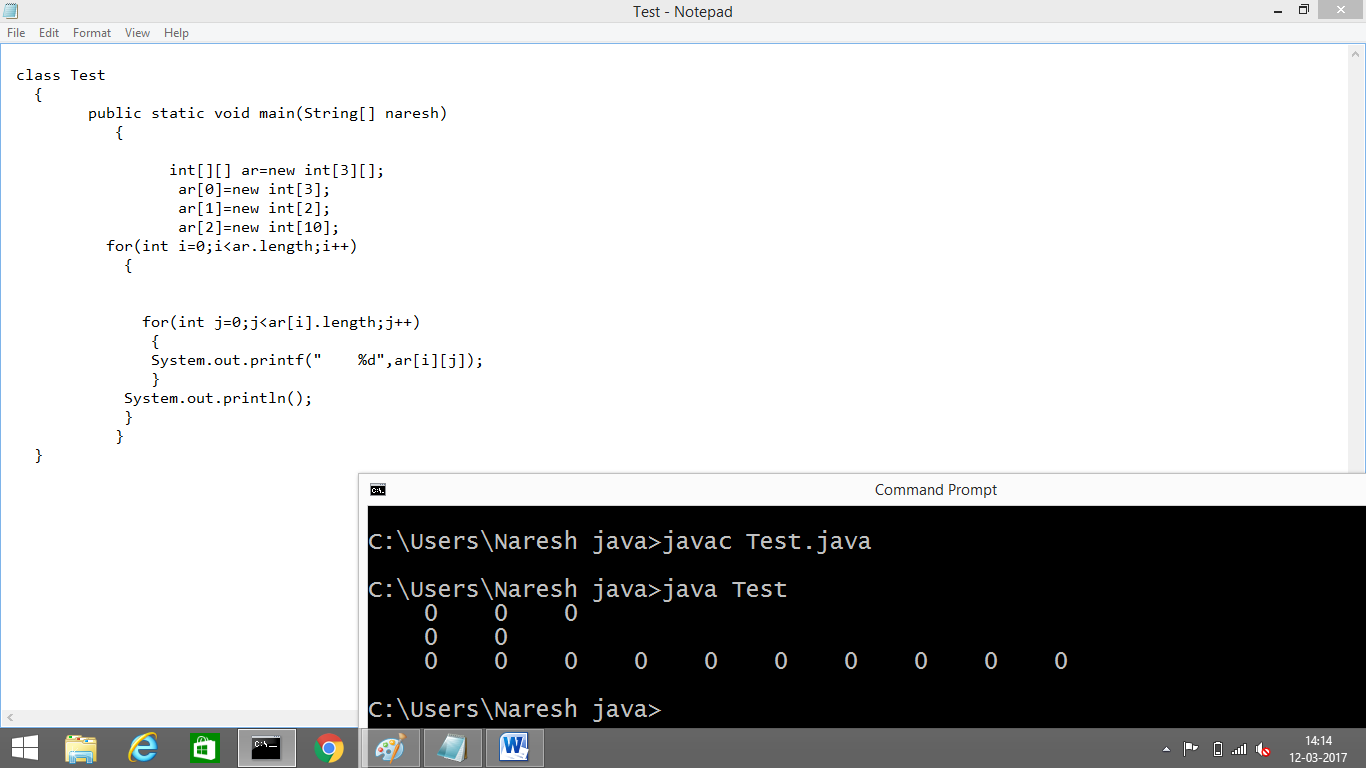
}



Partition Array:



Example:



Var\_arg Methods:

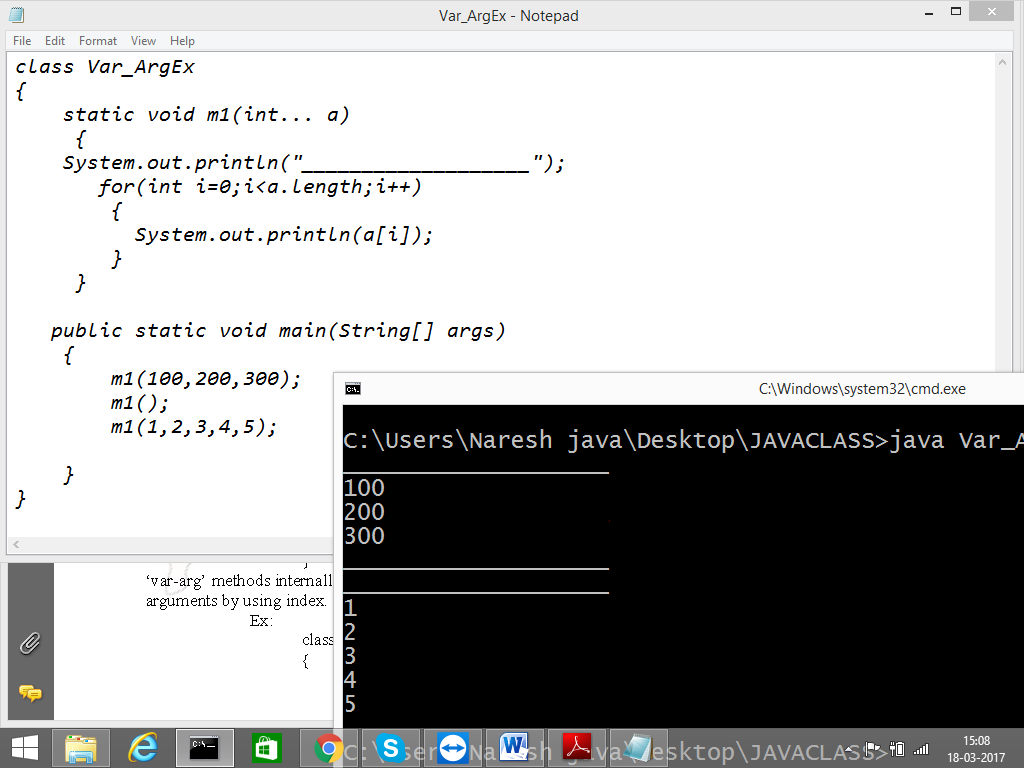
From 1.5 version on words we are allowed to declare a method with variable no of arguments such type of methods are called var-arg methods.

Syntax:

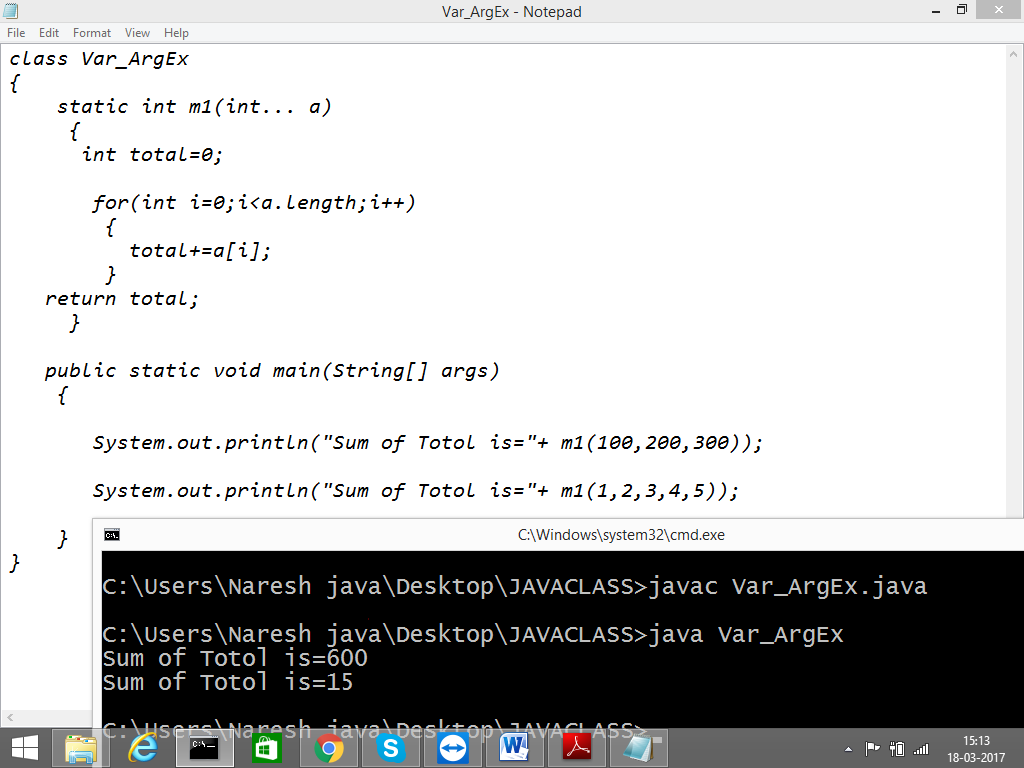
m1(int… i);

this methods is applicable for any no of int arguments including zero no of arguments.

Example:



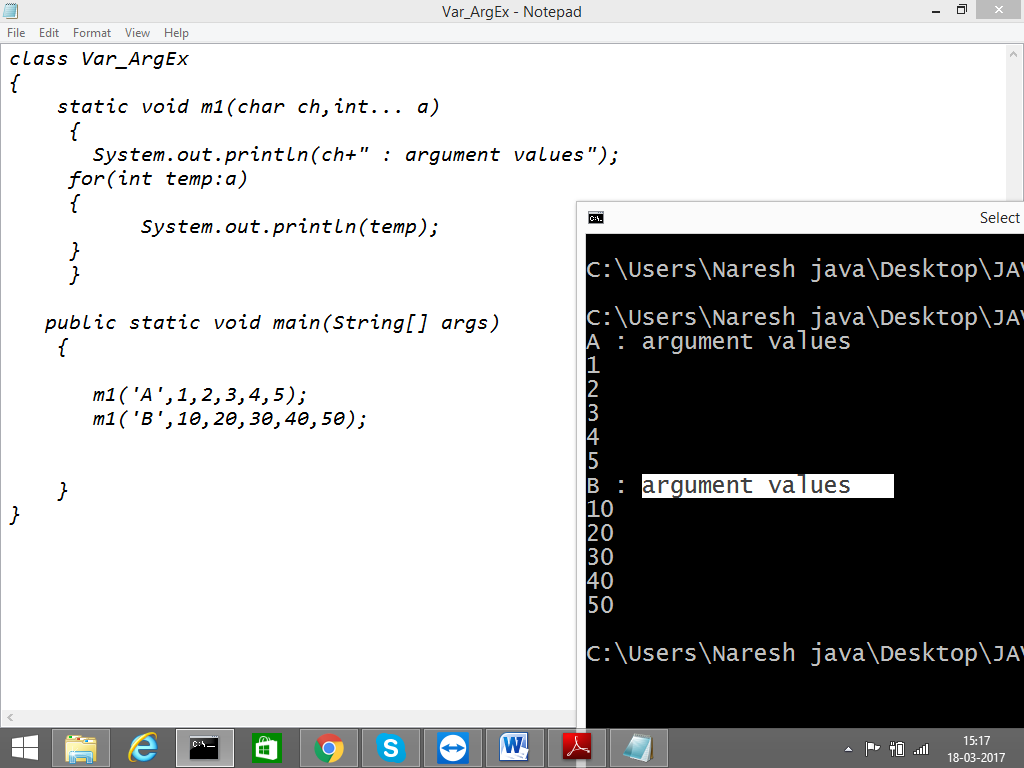
One More Example to print sum of total for given arguments:



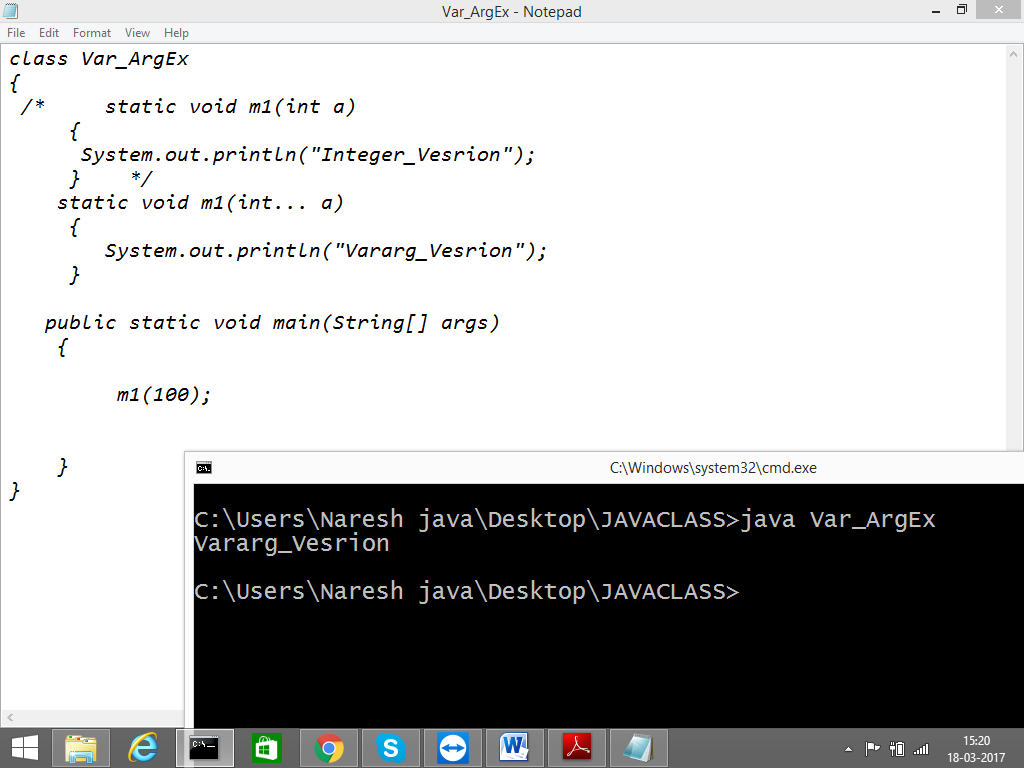
Note: ‘var-arg’ methods internally implemented by using single dimensional arrays. Hence we can differentiate arguments by using index.

One More Example For Var\_arg:

If we are mixing general parameters with var-arg parameter then var-arg parameter must be the last parameter otherwise compile time error.



Note: Var-arg method will always get least priority i.e if no other method matched then only var-arg method will be executed.



**Command line arguments & main() method**

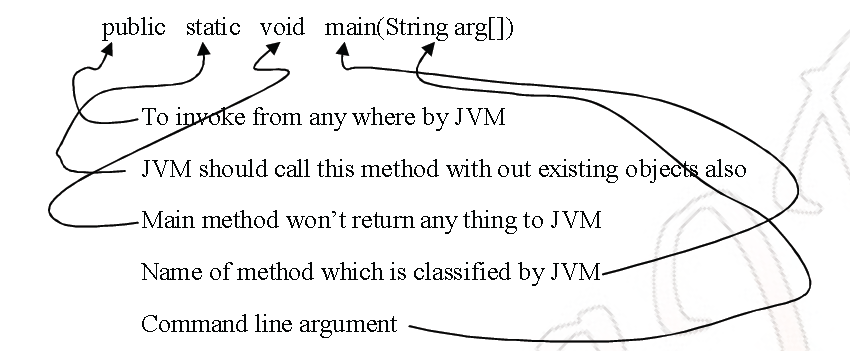
JVM always calls main method to start the program. Compiler is not responsible to check whether the class contain main() or not.

Hence if we are not having the main method we won’t get any C.E. But at runtime JVM

raises

Error: Main method not found in class Test, please define the main method as

public static void main(String[] args)



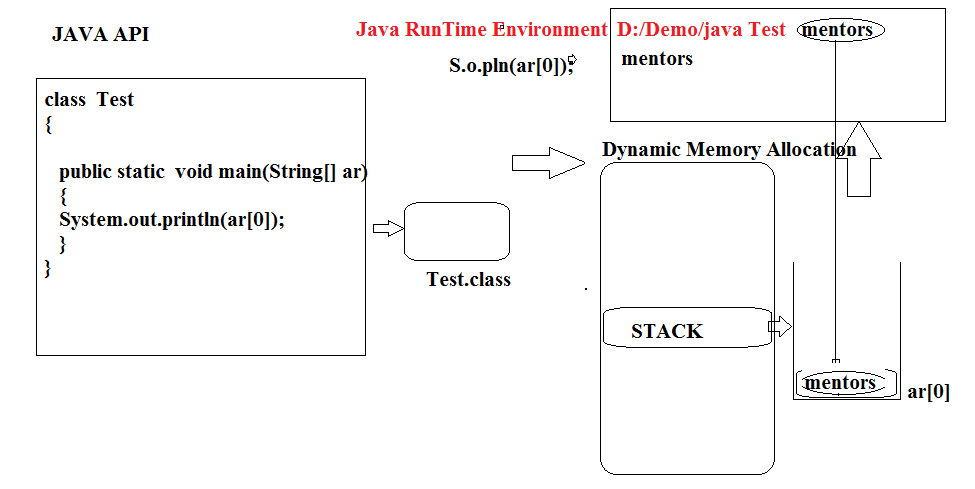
The **following main() are valid?**

public static final synchronized void main(String arg[]){}

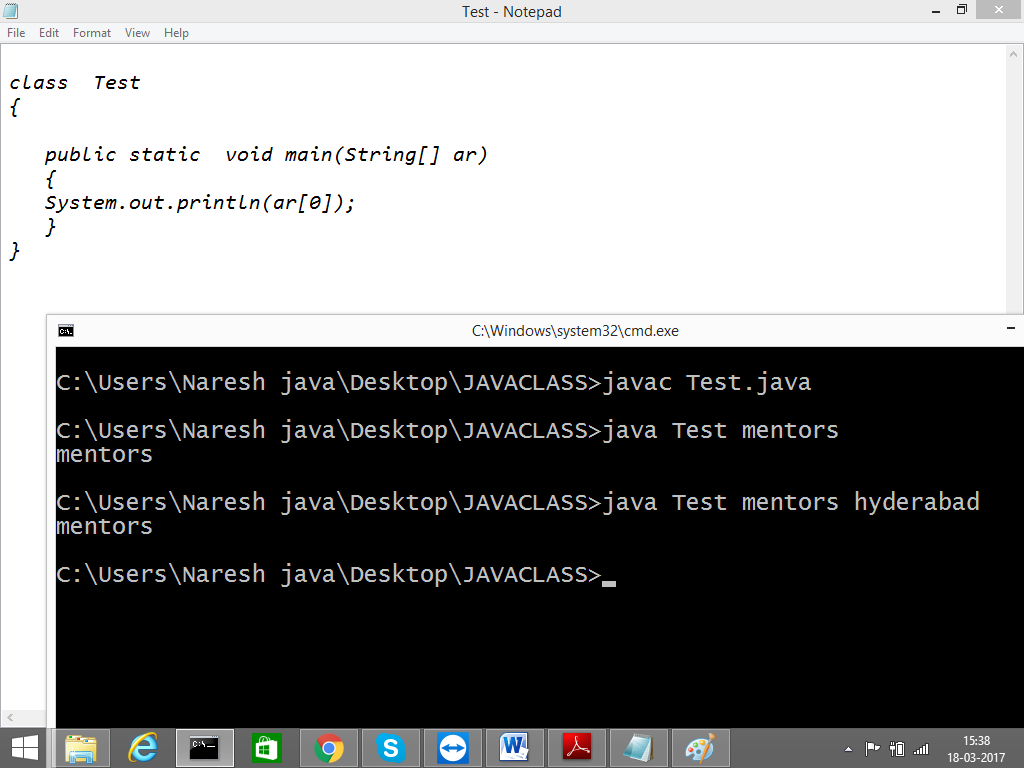
public static final void main(String arg[]){}

**Command line arguments**

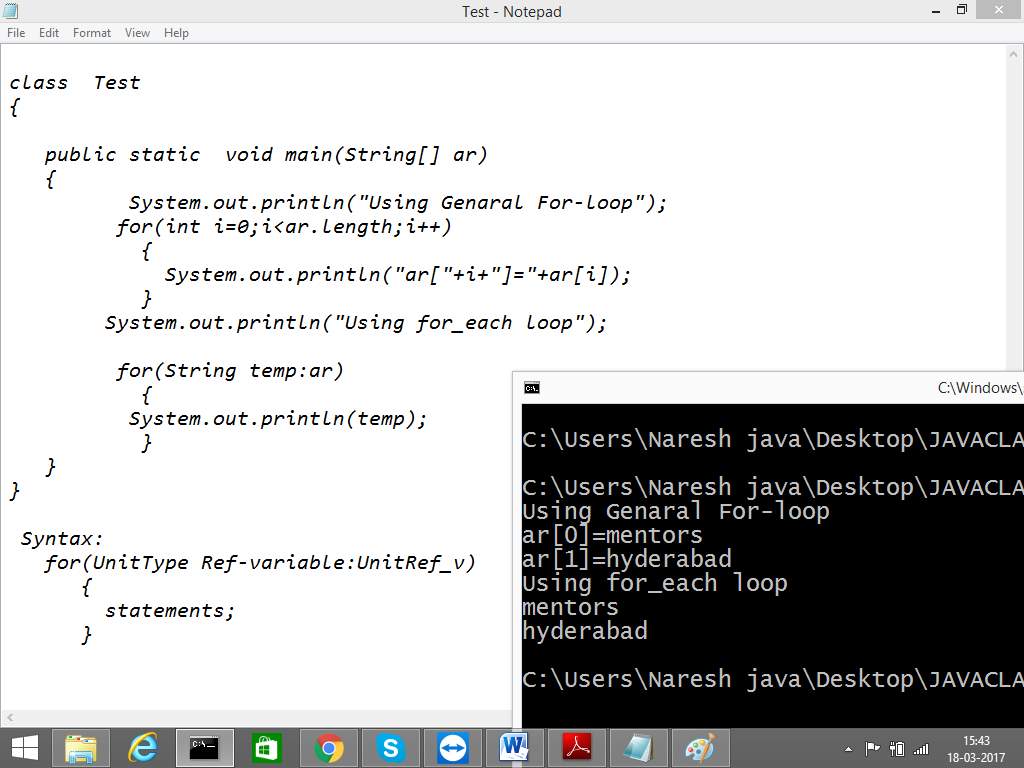
We can send command line argument to the main() by using with the reference of String[] .



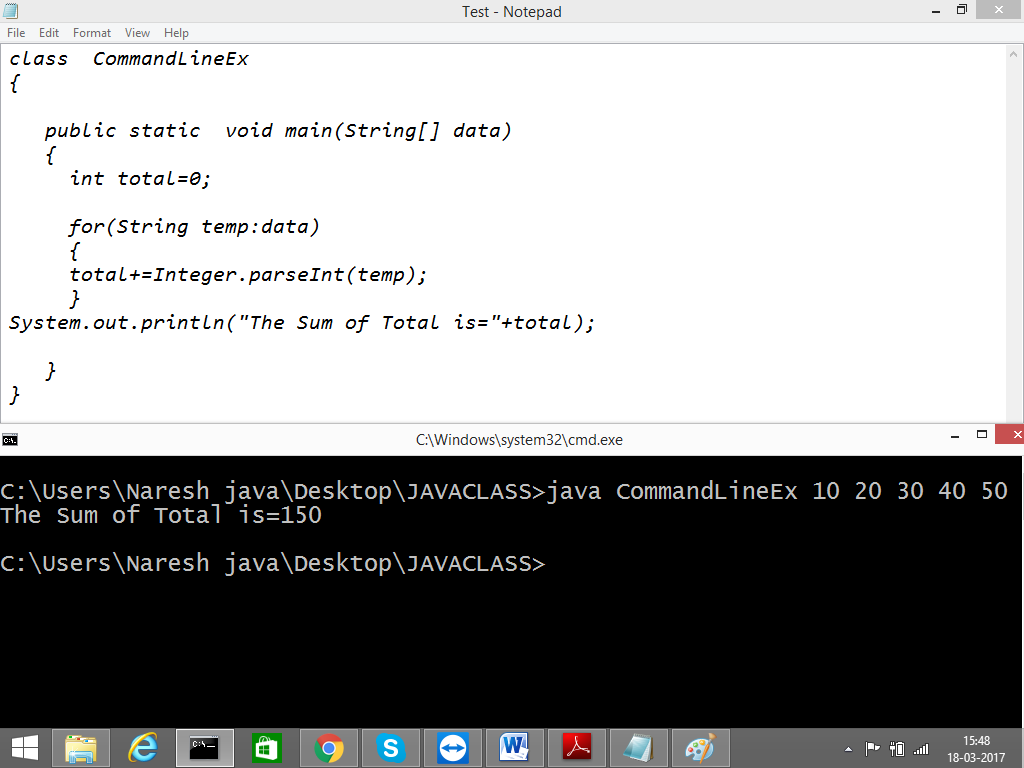
Example:



To Read More Elements You can use for\_each lopp or general for loop…



Q) WAP to printSum of Total by using CommandLine Arguments?



**Java coding Standards**

**Coding standards for classes**

Usually class name should be noun. Should starts with upper case letter and if it contain multiple

words every inner words also should start with capital letters.

Ex:

String

StringBuffer

NumberFormat

CustomerInformation

**Coding standards for Interfaces**

Usually interface named should be adjective, starts with capital letters and if it contains multiple

words, every inner word also should starts with capital letter.

Ex:

Runnable

Serializable

Clonable

Movable

Transferable

Workable

**Coding standards with methods**

Values should be either verbs or verb + noun combination.

Starts with lower case and every inner words starts with upper case(this convention is also called

camel case convention).

Ex:

getName(), getMessage(), toString(), show(), display().

**Coding standards for variables**

Usually the variable starts with noun and every inner word should start with upper case i.e camel

case convention.

Ex:

name, rollno, bandwidth, totalNumber.

**Coding standards for constants**

It should be noun, it should contain only upper case letters and words are separated with underscores.

Ex:

MAX\_SIZE, MIN\_PRIORITY, COLLEGE\_NAME.

**Java Been Coding Conventions**

A java bean is a normal java class with private properties & public getter and setter methods.

Ex:

public class StudentBeen

{

private String name;

public String getName()

{

return name;

}

public void setName(String name)

{

this.name = name;

}

}