DAY-23

-------

example:

--------

class Test

{

void add(int x, float y)

{

int z;

z=x+y;

System.out.println(z);

}

float add (int x, float y)

{

return x+y;

}

}

class Demo

{

public static void main(String[] args)

{

Test t = new Test();

int a = 10;

float b = 1.1f;

t.add(a,b);

}

}

output: --> error

note: In method overloading compiler will never considered the returntype of a method.

NUMERIC TYPE PROMOTION IN METHOD OVERLOADING.

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

During the method overloading if the datatype of the parameter doesnot match then before displaying the error the compiler try to promote the datatype

to the next available type and try to reslove the call.

// Example for NUMERIC TYPE PROMOTION

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

class Test

{

long add(int x, long y)

{

return x+y;

}

double add (double x, double y)

{

return x+y;

}

}

class Demo1

{

public static void main(String[] args)

{

Test t = new Test();

int a=10, b=20;

float c=1.1f, d=2.1f;

System.out.println(t.add(a,b));

System.out.println(t.add(c,d));

}

}

output:

-------

30

3.1999999284744263

PROBLEM WITH RESPECT TO NUMERIC TYPE PROMOTIONS

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

// Example 2 for NUMERIC TYPE PROMOTION

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

class Test

{

long add(int x, long y)

{

return x+y;

}

long add (long x, int y)

{

return x+y;

}

}

class Demo1

{

public static void main(String[] args)

{

Test t = new Test();

int a=10, b=20;

System.out.println(t.add(a,b));

}

}

OUTPUT:

-------

error: reference to add is ambiguous

NOTE: During the numeric type promotion if method call matches with multiple methods then compilation error will be displayed.

example:

--------

// returning multiple value from the method is not permitted in java.

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

class Test

{

long add(int x, long y)

{

return x+y, x-y, x\*y, x/y;

}

}

class Demo1

{

public static void main(String[] args)

{

Test t = new Test();

int a=10, b=20;

System.out.println(t.add(a,b));

}

}

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

VAR-ARGS METHOD

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

example:

--------

class Test

{

int add(int a, int b)

{

return a+b;

}

int add(int a, int b, int c)

{

return a+b+c;

}

int add(int a, int b, int c, int d)

{

return a+b+c+d;

}

int add(int a, int b, int c, int d, int e)

{

return a+b+c+d+e;

}

}

class Demo1

{

public static void main(String[] args)

{

Test t = new Test();

System.out.println(t.add(10,20));

System.out.println(t.add(10,20,30));

System.out.println(t.add(10,20,30,40));

System.out.println(t.add(10,20,30,40,50));

}

}

NOTE: The same above program which as overloaded method (add) can be simplified using VAR-ARGS parameter

// EXAMPLE FOR VAR-ARG METHOD

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

class Test

{

int add(int... a)

{

int sum =0;

for (int i : a )

{

sum=sum + i;

}

return sum;

}

}

class Demo1

{

public static void main(String[] args)

{

Test t = new Test();

System.out.println(t.add(10));

System.out.println(t.add(10,20));

System.out.println(t.add(10,20,30));

System.out.println(t.add(10,20,30,40));

System.out.println(t.add(10,20,30,40,50));

System.out.println(t.add(10,20,30,40,50,60));

}

}

Var-args allowed the method to accept 0 or multiple arguments(inputs) if we dont know how many arguments we are going to pass to the parameter then

we can go for var-args parameter.

The advantage is we do not have to provide over loaded methods.

NOTE:1

------

vara-args support any primitive datatype:

int... a; --> int[] a;

byte... a; --> byte[] a;

NOTE:2

------

valid syntax of var-args

int... a;

int ...a;

int...a;

NOTE:3

------

var-args parameter internally converts into single dimension array.

int... a ---> int []a;

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

DAY-24

-------

NOTE:4

-------

NUMERIC TYPE PROMOTION IN VAR-ARG METHOD.

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

// EXAMPLE FOR NUMERIC TYPE PROMOTION IN VAR-AR METHOD

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

class Test

{

float add(float... a)

{

float sum = 0.0f;

for (float i : a )

{

sum = sum + i;

}

return sum;

}

}

class Demo2

{

public static void main(String[] args)

{

Test t = new Test();

System.out.println(t.add()); // 0.0

System.out.println(t.add(1.1f,2.4f)); // 3.5

System.out.println(t.add(1.5f)); // 1.5

System.out.println(t.add(10,20)); // 30.0 --> NUMERIC TYPE PROMOTION

System.out.println(t.add(10,22.5f)); // 32.5 -->NUMERIC TYPE PROMOTION

System.out.println(t.add(10,24.5)); // COMPILATION ERROR

}

}

NOTE:5

-------

In a method along with var-arg parameter we can have other paramerter also.

// EXAMPLE FOR NUMERIC TYPE PROMOTION IN VAR-AR METHOD

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

class Test

{

void display(int num,String... s)

{

System.out.println("number value is :" + num);

for (String i : s)

{

System.out.println("String value is: " + i);

}

}

}

class Demo2

{

public static void main(String[] args)

{

Test t = new Test();

//t.display(10);

//t.display(10,"hello");

//t.display(10,20,"hello"); --> compilation error

//t.display(10,"hello",20); --> compilation error

t.display(10,"hello","WORLD");

}

}

NOTE:6

-------

Whenever var-arg paramter is used with other parameter then var-arg parameter should be present as last parameter.

example:

---------

class Test

{

void display(String... s,int num) --> COMPILATION ERROR

{

System.out.println("number value is :" + num);

for (String i : s)

{

System.out.println("String value is: " + i);

}

}

}

void display(String...s,int num) --> invalid

void display(int num,String...s) --> valid

void display(String...s) --> valid

NOTE:7

------

Only one var-arg paramter is allowed in the method signature.

example:

--------

class Test

{

void display(String... s,int... a) --> COMPILATION ERROR

{

System.out.println("number value is :" + num);

for (String i : s)

{

System.out.println("String value is: " + i);

}

}

}

NOTE:8

-------

Whenever the method call mathches with var-arg method and normal method then the execution will happen by executing the normal method because normal

method as highest priority when compared with var-arg method.

EXAMPLE:

--------

class Test

{

void fun(int... a)

{

System.out.println("I am inside the var-arg method");

for (int i : a)

{

//System.out.println("The value is: " + i);

}

}

void fun(int a)

{

System.out.println("I am inside the normal method");

System.out.println(a);

}

}

class Demo2

{

public static void main(String[] args)

{

Test t = new Test();

//t.fun();// --> var-arg method

//t.fun(10); --> normal method

//t.fun(10,20); --> var-arg method

}

}

NOTE:9

------

Within a class we cannot declare var-arg and single dimension array of the same datattype.

EXAMPLE:

--------

class Test

{

void fun(int... a) --> void fun(float... a) This is valid

{

System.out.println("I am inside the var-arg method");

}

void fun(int[] a)

{

System.out.println("I am inside the normal method");

}

}

NOTE:10

-------

void fun(int... a) --> fun(int[] a) --> valid

void fun(int[]... a) --> fun(int[][] a) --> valid

void fun(int[][]... a) --> fun(int[][][] a) --> valid

example:

--------

public static void main(String[] args)

static public void main(String... args)

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