**Var- arg methods (variable no of argument methods) (1.5)**

** Until 1.4v we can't declared a method with variable no. Of arguments.**

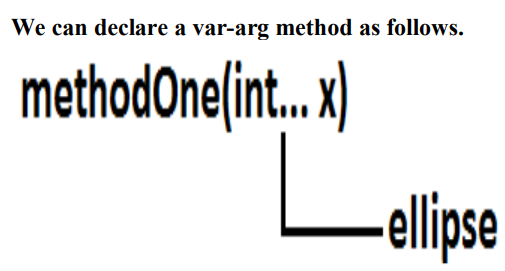
** If there is a change in no of arguments compulsory we have to define a new**

**method.**

** This approach increases length of the code and reduces readability.**

** But from 1.5 version onwards we can declare a method with variable no. Of**

**arguments such type of methods are called var-arg methods.**

****

**We can call or invoke this method by passing any no. Of int values including zero**

**number also.**

**Example:**

class Test

{

public static void methodOne(int... x)

{

System.out.println("var-arg method");

}

public static void main(String[] args)

{

methodOne();

methodOne(10);

methodOne(10,20,30);

}

}

Output:

var-arg method

var-arg method

var-arg method

**Internally var-arg parameter implemented by using single dimensional array hence**

**within the var-arg method we can differentiate arguments by using index.**

**Example:**

class Test

{

public static void sum(int... x)

{

int total=0;

for(int i=0;i<x.length;i++)

{total=total+x[i];

}

System.out.println("The sum :"+total);

}

public static void main(String[] args)

{

sum();

sum(10);

sum(10,20);

sum(10,20,30,40);

}

}

Output:

The sum: 0

The sum: 10

The sum: 30

The sum: 100

The sum: 30

The sum: 100

Case 1:

**Which of the following var-arg method declarations are valid?**

**1. methodOne(int... x) (valid)**

**2. methodOne(int ...x) (valid)**

**3. methodOne(int...x) (valid)**

**4. methodOne(int x...) (invalid)**

**5. methodOne(int. ..x) (invalid)**

**6. methodOne(int .x..) (invalid)**

**Case 2:**

**We can mix var-arg parameter with general parameters also.**

**Example:**

**methodOne(int a,int... b) //valid**

**methodOne(String s,int... x) //valid**

**Case 3:**

**if we mix var-arg parameter with general parameter then var-arg parameter should be**

**the last parameter.**

**Example:**

**methodOne(int a,int... b) //valid**

**methodOne(int... a,int b) //(invalid)**

**Case 4:**

**With in the var-arg method we can take only one var-arg parameter. i.e., if we are**

**trying to more than one var-arg parameter we will get CE.**

**Example:**

**methodOne(int... a,int... b) //(invalid)**

**Case 5:**

class Test

{public static void methodOne(int i)

{

System.out.println("general method");

}

public static void methodOne(int... i)

{

System.out.println("var-arg method");

}

public static void main(String[] args)

{

methodOne();//var-arg method

methodOne(10,20);//var-arg method

methodOne(10);//general method

}

}

**In general var-arg method will get least priority that is if no other method matched then**

**only var-arg method will get the chance this is exactly same as default case inside a**

**switch.**

**Case 6:**

For the var-arg methods we can provide the corresponding type array as argument.

Example:

class Test

{

{

System.out.println("var-arg method");

}

public static void main(String[] args)

{

methodOne(new int[]{10,20,30});//var-arg method

}

}

Case 7:

class Test

{

public void methodOne(int[] i){}

public void methodOne(int... i){}

}

Output:

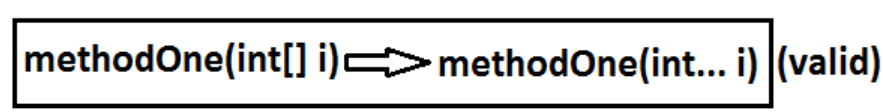
Compile time error.

**Cannot declare both methodOne(int...) and methodOne(int[]) in Test**

**Single Dimensional Array Vs Var-Arg Method:**

**Case 1:**

**Wherever single dimensional array present we can replace with var-arg parameter.**



class Test

{

public static void main(String... args)

{

System.out.println("var-arg main method");//var-arg main

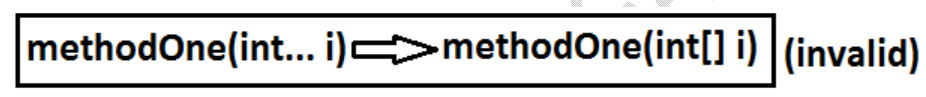
method

}

}

Case 2:

Wherever var-arg parameter present we can't replace with single dimensional array.



**Note :**

**1. methodOne(int... x)**

**we can call this method by passing a group of int values and x will become 1D**

**array. (i.e., int[] x)**

**2. methodOne(int[]... x)**

**we can call this method by passing a group of 1D int[] and x will become 2D**

**array. ( i.e., int[][] x)**

**Above reasons this case 2 is invalid.**

Example:

class Test

{

public static void methodOne(int[]... x)

{

for(int[] a:x)

{ System.out.println(a[0]);

}

}

public static void main(String[] args)

{

int[] l={10,20,30};

int[] m={40,50};

methodOne(l,m);

}

}

Output:

10

40

Analysis:

