WAP to show that if we change only the return type of the method then it is not an overloaded method

MOL6.java

```
void main()
{
}

void accept(int x, int y)
{
}

int accept(int x, int y) //error (it is not an overloaded method)
{
    return 0;
}
```

WAP in java that describes static method can be overloaded as well as type promotion :

* With following data type the promotion of data type is as follows :

a) char -> int -> long -> float -> double
b) byte -> short -> int -> long -> float -> double

MOL6.java

```
void main()
{
    accept('a'); //calling the static method
}

public static void accept(char c)
{
    IO.println("char");
}

public static void accept(int i)
{
    IO.println("int");
}

public static void accept(long l)
{
    IO.println("long");
}

public static void accept(float f)
{
    IO.println("float");
}

public static void accept(double d)
{
    IO.println("double");
}
```

---

for each loop :

It was introduced from JDK 1.5d.

It is also known as enhanced for loop.

It is mainly used to retrieve the data one by one from the collection like array.

Internally java compiler will convert this for each loop into ordinary for loop.


```
int []arr = {10, 20, 30};   //Array in java

for(int  x : arr)
{
    IO.println(x);
}
```

Note : x is an ordinary variable so it can hold only **one value but arr is an array variable so it can hold multiple values.**

loop.java

```
int []arr = {10, 20, 30};

for(int  x : arr)
{
    IO.println(x);
}
```

loop.class

```
int []arr = {10, 20, 30};

for(int x = 0; x<arr.length; x++)
{
    IO.println(arr[x]);
}
```

Java compiler generated code to convert for each loop into ordinary for loop

//Programs :

ForEach1.java

```
void main()
{
    int values[] = {10,20,30,40, 50 ,60,70};

    for(int value : values)
    {
        IO.println(value);
    }
}
```

WAP to retrieve the array values using for each loop for String type :

ForEach2.java

```
void main()
{
    String cities[] = {"Hyd", "Pune", "Blgr", "Goa", "Amaravati"};

    for(String city : cities)
    {
        IO.println(city);
    }
}
```

WAP in java where array can hold hetrogeneous types of data and print the data by using for each loop

(in java, An array can hold hetrogeneous types of data also by using Object array (Object array)

ForEach3.java

```
void main()
{
    Object []values = {12, 34.99, 'A', false, "BET"};

    for(Object value : values)
    {
        IO.println(value);
    }
}
```

Var args in Java :

* It is known as Variable Argument.
* It was introduced from JDK 1.5V.
* It is represented by exactly 3 dots.
* Internally it is an array.
* Var args we can only use as a method OR Constructor parameter.

Example :

```
void main()
{
    int ...arr = {10 = {10,20,30};
}
```

* Var args must be only one and last parameter.

//Programs

WAP to show that var args can accept 0 to n number of arguments.

VarArgsDemo1.java

```
void main()
{
    input();
    input(1);
    input(10,20);
    input(100,200,300);
    input(100,200,300, 400);
}

void input(int ...x)
{
    IO.println("Executed !!");
}
```

WAP to show that Var args can accept hetrogeneous types of data

VarArgsDemo2.java

```
void main()
{
    input(1,2,3, 'A', true, "BET");
}

void input(Object ...values)
{
    for(Object value : values)
    {
        IO.println(value);
    }
}
```

WAP in java to add the argument values by using var args

VarArgsDemo3.java

```
void input(int ...values)
{
    int sum = 0;

    for(int value : values)
    {
        sum = sum + value;
    }

    IO.println("Sum of parameter is :"+sum);
}

void main()
{
    input(10, 20, 30, 40 , 50, 50);
}
```

WAP in java to show varargs must be only one and last parameter

VarArgsDemo4.java

```
void main()
{
    accept(100, 10,20,30,90);
}

public void accept(int a, int ...values)
{
    IO.println("x value is :"+x);

    for(int value : values)
    {
        IO.println(value);
    }
}

//All the commented code -> Invalid
/*
public void accept(double ...a, int ...y)
{
}

public void accept(int ...a, int y)
{
}

public void accept(int ...a, int ...y)
{
}
*/
```

===
=============