Method Overloading in Java

If a class have multiple methods by same name but different parameters, it is known as Method Overloading.

If we have to perform only one operation, having same name of the methods increases the readability of the program.

**Different ways to overload the method**

1) By changing the no. of arguments

2) By changing the datatype

Q 1) Why method overloading is not possible by changing the return type

Q 2) Can we overload the main method

Q3) Method overloading with Type Promotion

**Advantage of method overloading?**

Method overloading increases the readability of the program.

**Different ways to overload the method**

1) By changing number of arguments

2) By changing the data type

**1)Example of Method Overloading by changing the no. of arguments**

class Calculation{

void sum(inta,int b){System.out.println(a+b);}

void sum(inta,intb,int c){System.out.println(a+b+c);}

public static void main(String args[]){

Calculation obj=new Calculation();

obj.sum(10,10,10);

obj.sum(20,20);

} }

Output:30

40

**2)Example of Method Overloading by changing data type of argument**

class Calculation2{

void sum(inta,int b){System.out.println(a+b);}

void sum(double a,double b){System.out.println(a+b);}

public static void main(String args[]){

Calculation2 obj=new Calculation2();

obj.sum(10.5,10.5);

obj.sum(20,20);

} }

**Output**:21.0

40

**Que) Why Method Overloaing is not possible by changing the return type of method?**

In java, method overloading is not possible by changing the return type of the method because there may occur ambiguity. Let's see how ambiguity may occur:

**because there was problem:**

class Calculation3{

int sum(int a,int b){System.out.println(a+b);}

double sum(int a,int b){System.out.println(a+b);}

public static void main(String args[]){

Calculation3 obj=new Calculation3();

int result=obj.sum(20,20); //Compile Time Error

} }

**Output:**

int result=obj.sum(20,20); //Here how can java determine which sum() method should be called

**Can we overload main() method?**

Yes, by method overloading. You can have any number of main methods in a class by method overloading. Let's see the simple **example:**

class Overloading1{

public static void main(int a){

System.out.println(a);

}

public static void main(String args[]){

System.out.println("main() method invoked");

main(10);

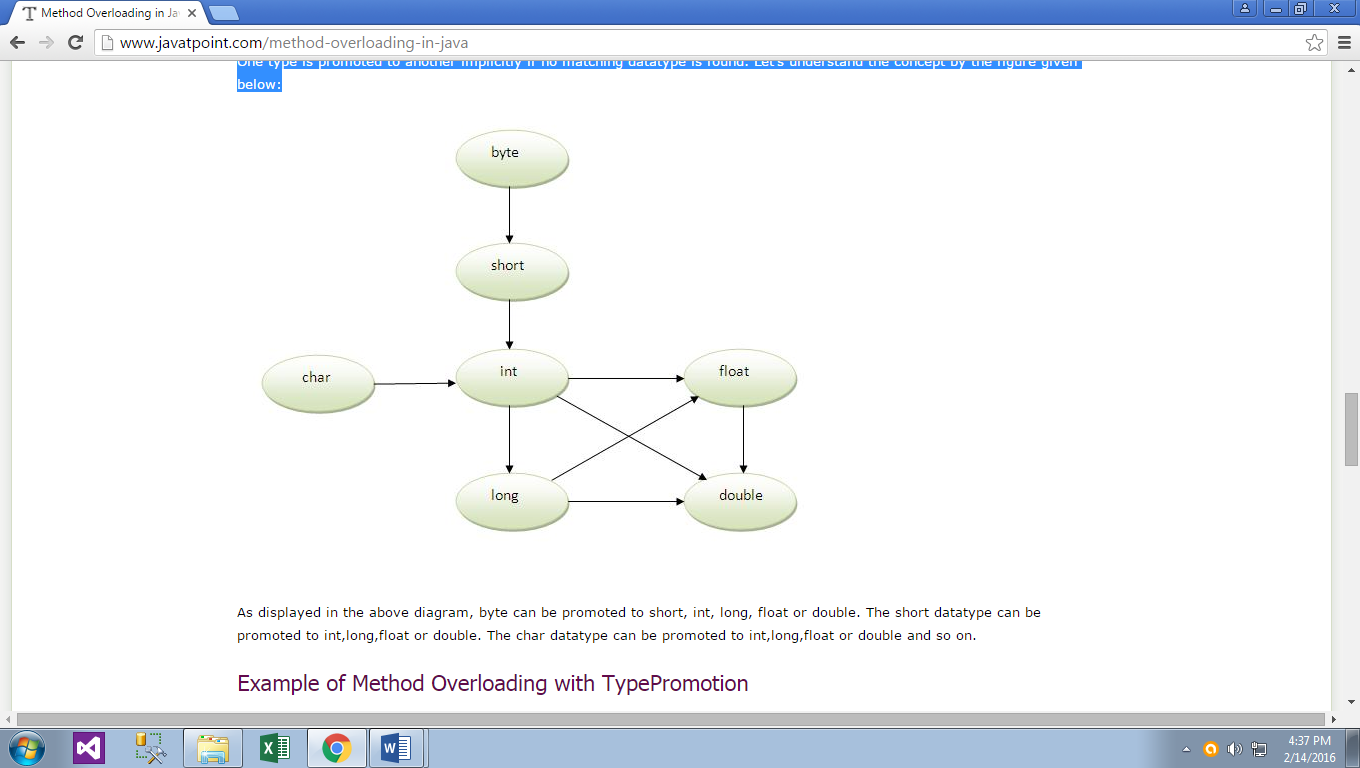
} }

**Output**:main() method invoked

10

**Method Overloading and TypePromotion**

One type is promoted to another implicitly if no matching datatype is found. Let's understand the concept by the figure given below:



As displayed in the above diagram, byte can be promoted to short, int, long, float or double. The short datatype can be promoted to int,long,float or double. The char datatype can be promoted to int,long,float or double and so on.

**Example of Method Overloading with TypePromotion**

class OverloadingCalculation1{

void sum(int a,long b){System.out.println(a+b);}

void sum(int a,int b,int c){System.out.println(a+b+c);}

public static void main(String args[]){

OverloadingCalculation1 obj=new OverloadingCalculation1();

obj.sum(20,20);//now second int literal will be promoted to long

obj.sum(20,20,20);

} }

**Output**:40

60

**Example of Method Overloading with TypePromotion if matching found**

If there are matching type arguments in the method, type promotion is not performed.

class OverloadingCalculation2{

void sum(int a,int b){System.out.println("intarg method invoked");}

void sum(long a,long b){System.out.println("long arg method invoked");}

public static void main(String args[]){

OverloadingCalculation2 obj=new OverloadingCalculation2();

obj.sum(20,20);//now int arg sum() method gets invoked

} }

**Output**:intarg method invoked

**Example of Method Overloading with TypePromotion in case ambiguity**

If there are no matching type arguments in the method, and each method promotes similar number of arguments, there will be ambiguity.

class OverloadingCalculation3{

void sum(inta,long b){System.out.println("a method invoked");}

void sum(long a,int b){System.out.println("b method invoked");}

public static void main(String args[]){

OverloadingCalculation3 obj=new OverloadingCalculation3();

obj.sum(20,20);//now ambiguity

} }

**Output**:Compile Time Error

**Important:**One type is not de-promoted implicitly for example double cannot be depromoted to any type implicitely.

Operator Overloading……

[Click Here](Why%20Java%20does%20not%20support%20Operator%20overloading.docx)