Method Overloading and Overridding in Java

**Method Overloading**

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(int a,int 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[]){

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(int a,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

**Que)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);

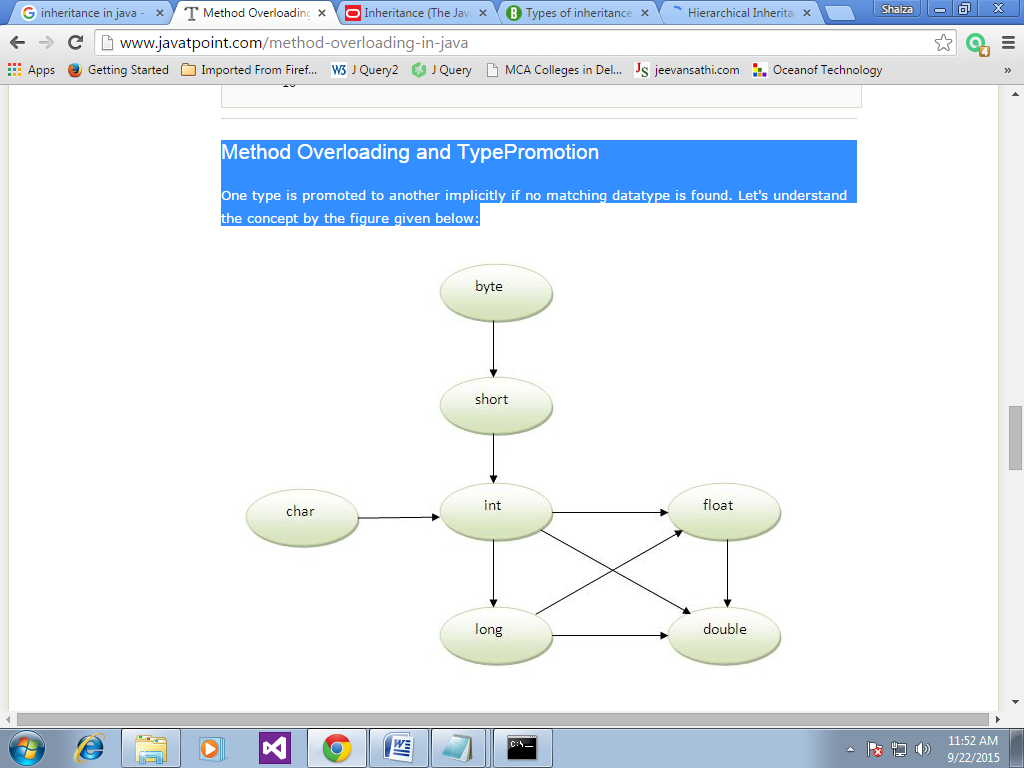
} }

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);

}

}

Test it Now

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("int arg 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**:int arg 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(int a,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

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

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

**Method Overriding**

If subclass (child class) has the same method as declared in the parent class, it is known as method overriding in java.

In other words, If subclass provides the specific implementation of the method that has been provided by one of its parent class, it is known as method overriding.

Q1) Can we override the static method

Q2) method overloading vs method overriding

If subclass (child class) has the same method as declared in the parent class, it is known as method overriding in java.

**Usage of Java Method Overriding**

1) Method overriding is used to provide specific implementation of a method that is already provided by its super class.

2) Method overriding is used for runtime polymorphism

**Rules for Java Method Overriding**

1) method must have same name as in the parent class

2) method must have same parameter as in the parent class.

3) must be IS-A relationship (inheritance).

**Understanding the problem without method overriding**

Let's understand the problem that we may face in the program if we don't use method overriding.

class Vehicle{

void run(){System.out.println("Vehicle is running");}

}

class Bike extends Vehicle{

public static void main(String args[]){

Bike obj = new Bike();

obj.run();

} }

Output:Vehicle is running

**Problem is** that I have to provide a specific implementation of run() method in subclass that is why we use method overriding.

**Example of method overriding**

class Vehicle{

void run(){System.out.println("Vehicle is running");}

}

class Bike2 extends Vehicle{

void run(){System.out.println("Bike is running safely");}

public static void main(String args[]){

Bike2 obj = new Bike2();

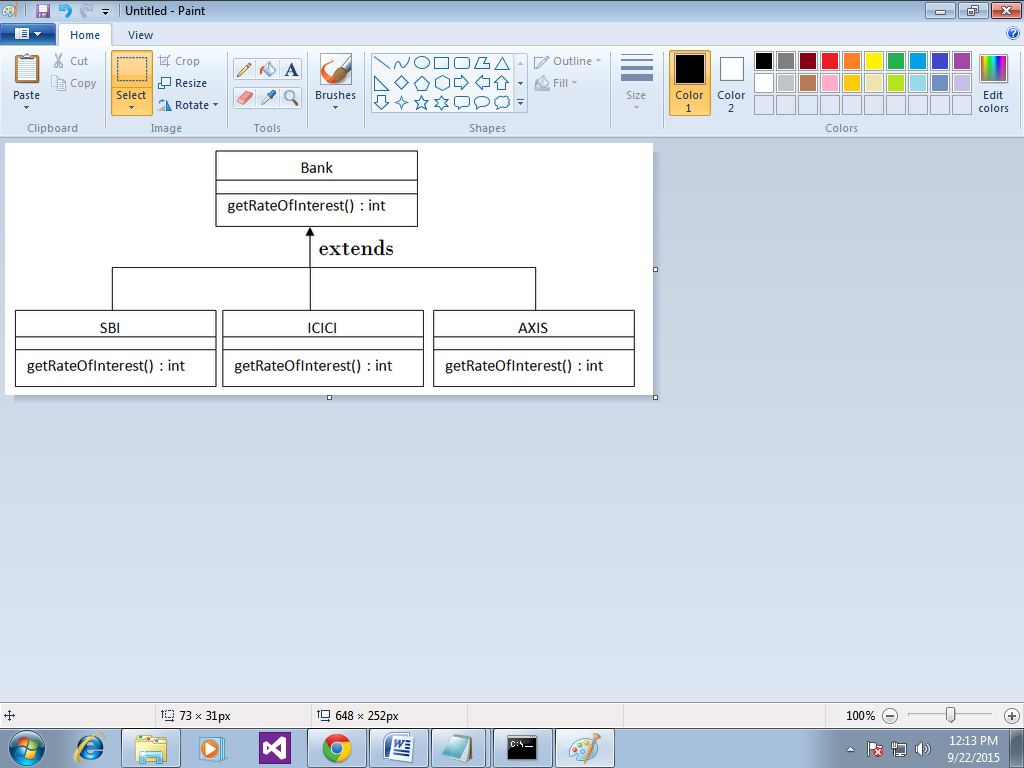
obj.run();

}

Output:Bike is running safely

**Real example of Java Method Overriding**

Consider a scenario, Bank is a class that provides functionality to get rate of interest. But, rate of interest varies according to banks. For example, SBI, ICICI and AXIS banks could provide 8%, 7% and 9% rate of interest.



class Bank{

int getRateOfInterest(){return 0;}

}

class SBI extends Bank{

int getRateOfInterest(){return 8;}

}

class ICICI extends Bank{

int getRateOfInterest(){return 7;}

}

class AXIS extends Bank{

int getRateOfInterest(){return 9;}

}

class Test2{

public static void main(String args[]){

SBI s=new SBI();

ICICI i=new ICICI();

AXIS a=new AXIS();

System.out.println("SBI Rate of Interest: "+s.getRateOfInterest());

System.out.println("ICICI Rate of Interest: "+i.getRateOfInterest());

System.out.println("AXIS Rate of Interest: "+a.getRateOfInterest());

} }

Output:

SBI Rate of Interest: 8

ICICI Rate of Interest: 7

AXIS Rate of Interest: 9

=======

**Q) Can we override static method?**

No, static method cannot be overridden. It can be proved by runtime polymorphism, so we will learn it later.

**Q) Why we cannot override static method?**

because static method is bound with class whereas instance method is bound with object. Static belongs to class area and instance belongs to heap area.

**Q) Can we override java main method?**

No, because main is a static method.

**Difference between method Overloading and Method Overriding in java**

|  |  |  |
| --- | --- | --- |
| **No.** | **Method Overloading** | **Method Overriding** |
| 1) | Method overloading is used *to increase the readability* of the program. | Method overriding is used *to provide the specific implementation* of the method that is already provided by its super class. |
| 2) | Method overloading is performed *within class*. | Method overriding occurs *in two classes* that have IS-A (inheritance) relationship. |
| 3) | In case of method overloading, *parameter must be different*. | In case of method overriding,*parameter must be same*. |
| 4) | Method overloading is the example of *compile time polymorphism*. | Method overriding is the example of *run time polymorphism*. |
| 5) | In java, method overloading can't be performed by changing return type of the method only. *Return type can be same or different* in method overloading. But you must have to change the parameter. | *Return type must be same or covariant* in method overriding. |

**Method Overloading example**

class OverloadingExample{

static int add(int a,int b){return a+b;}

static int add(int a,int b,int c){return a+b+c;}

}

**Method Overriding example**

class Animal{

void eat(){System.out.println("eating...");}

}

class Dog extends Animal{

void eat(){System.out.println("eating bread...");}

}