HowToDoInJava

Overloading vs Overriding in Java

Last Updated: January 29, 2022 By: Lokesh Gupta Java Object Oriented Programming Inheritance, Java OOP

Method overloading and overriding (in other words, polymorphism in java) is neither a very difficult concept and nor it's one of very unknown topics. Yet, I am bringing this topic here in this post, because at the same time it is very easy to make mistakes when such concepts are tested in java **interviews** using multiple code examples. I am not giving any new concept here, but I intend to revise your existing knowledge regarding rules of **method overloading and overriding in java**.

Method Overloading Rules

Here are the rules which you keep in mind while overloading any method in java:

1) First and important rule to overload a method in java is to **change method signature**. Method signature is made of **number of arguments, type of arguments and order of arguments** if they are of different types.

```
public class DemoClass {
  // Overloaded method
  public Integer sum(Integer a, Integer b) {
    return a + b;
  }

  // Overloading method
  public Integer sum(Float a, Integer b) {  //Valid
    return null;
  }
}
```

2) Return type of method is never part of method signature, so only **changing the return type of method does not amount to method overloading**.

```
public class DemoClass {
  // Overloaded method
  public Integer sum(Integer a, Integer b) {
    return a + b;
  }

  // Overloading method
  public Float sum(Integer a, Integer b) {     //Not valid; Compile time error
    return null;
  }
}
```

3) Thrown exceptions from methods are also not considered when overloading a method. So your overloaded method throws the same exception, a different exception or it simply does no throw any exception; **no effect at all on method loading**.

```
public class DemoClass {
  // Overloaded method
  public Integer sum(Integer a, Integer b) throws NullPointerException{
    return a + b;
  }

  // Overloading method
  public Integer sum(Integer a, Integer b) throws Exception{  //Not valid; Compil
    return null;
  }
}
```

Read More: What is polymorphism in java

Method Overriding Rules

We read above the rules for method overloading, now its time to list down the rules which you should keep remember while overriding a method in java.

1) The method **argument list in overridden and overriding methods must be exactly same** If they don't match, you will end up with an overloaded method.

2) The **return type of overriding method can be child class of return type declared in overridden method**.

```
public class SuperClass {
  //Overriden method
  public Number sum(Integer a, Integer b) {
    return a + b;
  }
}

class SubClass extends SuperClass {
  //Overriding method
  @Override
  public Integer sum(Integer a, Integer b) {    //Integer extends Number; so it's
    return a + b;
  }
}
```

3) Above all rules, **private, static and final methods can not be overridden** in java in any way. As simple as that !!

```
public class SuperClass {
  private Integer sum(Integer a, Integer b) {   //private method; overriding not
    return a + b;
  }
}

class SubClass extends SuperClass {
  //Overriding method
  public Integer sum(Integer a, Integer b) {
    return a + b;
  }
}
```

4) Overriding method can not throw checked Exception higher in hierarchy than thrown by overridden method. Let's say for example overridden method in parent class throws `FileNotFoundException`, the overriding method in child class can throw `FileNotFoundException`; but it is not allowed to throw `IOException` or `Exception`, because `IOException` or `Exception` are higher in hierarchy i.e. super classes of `FileNotFoundException`.

More to it, you can omit the exception declaration from overriding method. It's allowed and perfectly valid. Also overriding method can throw any unchecked (runtime) exception, regardless of whether the overridden method declares the exception.

```
public class SuperClass {
  //Overriden method
  public Integer sum(Integer a, Integer b) throws FileNotFoundException {
    return a + b;
  }
}

class SubClass extends SuperClass {
  //Overriding method
  public Integer sum(Integer a, Integer b) throws IOException {     //Not valid;
    return a + b;
  }
  //Exception IOException is not compatible with throws clause in SuperClass.sum(
  public Integer sum(Integer a, Integer b)  {            //It's valid; Don't decla
    return a + b;
  }
}
```

5) Also note that **overriding method can not reduce the access scope of overridden method**. Put in simple words, if overridden method in parent class is protected, then overriding method in child class can not be private. It must be either protected (same access) or public (wider access).

```
public class SuperClass {
  //Overriden method
  protected Integer sum(Integer a, Integer b) {
    return a + b;
  }
}

class SubClass extends SuperClass {
  //Overriding method
  //Not valid; Compile time error &quot;Cannot reduce the visibility of the inher
  private Integer sum(Integer a, Integer b)  {
    return a + b;
  }
}
```

Not to repeat again that method overriding is legal when talking in terms on parent classes and child classes. It does not happen within same class.

To verify that you are correctly overriding a method or not, simply use the annotation **@Override** on overriding method. It will verify all the method overriding rules for you. If there is any issue, it will result in compile time error.

Read More: Java Interview Questions

That's all for this simple yet important concept to brush your basics in core java and object oriented programming.

Happy Learning !!

Ref: Oracle Blog

Was this post helpful?

Let us know if you liked the post. That's the only way we can improve.

Yes

No

Recommended Reading:

1. Overriding final static method in Java
2. TypeScript Function Overloading
3. Guide to Inheritance
4. Interface vs Abstract Class in Java
5. Encapsulation vs Abstraction in Java
6. Java Access Modifiers
7. Multiple Inheritance in Java
8. Constructors in Java
9. Java extends vs implements Keywords
0. Java Instance Initializer Blocks

17 thoughts on "Overloading vs Overriding in Java"

gourav

May 2, 2018 at 7:49 am

why final,static can not be overriding?

Reply

priya

November 3, 2017 at 12:06 pm

Method overloading is performed within class. Method overriding occurs in two classes that have IS-A (inheritance) relationship. Both are very important in java.

Reply

Mayur kohli

October 7, 2017 at 1:49 pm

Never read a better article elsewhere. Simple description and I will never forget overloading and overriding hereafter.can you please explain java polymorphism it hasnt been very clear to me all time although partially i understand still not clear.
Thanks a lot for sharing this!

Reply

Navin

February 2, 2016 at 7:35 am

Super

Reply

rb

July 20, 2015 at 8:29 pm

Thanks!Crystal clear!

Reply

Asgar

April 7, 2015 at 11:39 am

Hi,
why I am not getting any compile time exception in this example? coz in super class we are using "FileNotFoundException" and in subclass while overriding we have "IOException"

package Overriding;

import java.io.FileNotFoundException;
import java.io.IOException;

class SuperClass1{
public Number sum(int a,int b)throws FileNotFoundException{
System.out.println("super");
return a+b;

}
}
public class SubClass1 {
/*
public static void main(String[] args)throws IOException {
SubClass1 sc= new SubClass1();
sc.sum(10, 30);
}*/
public Integer sum(int a,int b)throws IOException{

System.out.println("sub");

return a+b;

}

}

Reply

Vishal Mali

December 28, 2016 at 2:32 pm

subclass should extend super class first….:)

Reply

Shrikant

November 27, 2014 at 12:56 pm

The below mentioned code works perfectly in JDK1.6 . I Just want to know is it overloading ??

public class TTTT {

public void display(){

System.out.println("Inside display of TTTT class");

}

public int display(int i){

System.out.println("Inside display of TTTT class values is "+ i);

return 0;

}

}

Reply

Lokesh Gupta

November 27, 2014 at 3:05 pm

Yes it is.

Reply

Nitish

September 24, 2014 at 5:54 am

Hi Lokesh,
which is best API to use in coding google's Lists or ImmutableList

Reply

Lokesh Gupta

September 24, 2014 at 7:13 am

Nitish, both seems to be different things to me. Lists provide static methods for creating lists in many ways including ImmutableList. ImmutableList is what it says that it is immutable. They are not alternative to one another.

Reply

Anil

August 26, 2014 at 6:25 am

why String literal and String object return same hashcode? what is the internal process

Reply

Lokesh Gupta

August 26, 2014 at 6:56 am

String literals are short-hand representation of string objects only. They are used for performance gain using String pool concept.
Read More : https://howtodoinjava.com/java/string/interview-stuff-about-string-class-in-java/

Reply

Rajendra

July 25, 2014 at 6:53 am

Hi Lokesh,

Thank you for posting about rules for overriding a method in java.

I have a doubt If super class method is default and sub class method is in another package. Its tries to override the super class method what will happen ?

As we discussed method overriding in same package but we missed that if sub class is in another package..

Thanks in advance..

Reply

Lokesh Gupta

July 25, 2014 at 8:18 am

default methods are always like private methods in different package. As you can not override private methods (even in same package), so you can not override default method (only in different package). Thanks for pointing out.

Reply

Bhaskar

July 14, 2014 at 5:50 am

Hi Lokesh,

Thanks For keeping us on learning track and for sharing the great post on regular basis !!

Here i do have some doubts on method overloading :

Overloading Of Method is a static binding on the basis of which we can say that compiler plays important role in defining the overloading rule.Now, as we know exception get caught during runtime , by considering this will it be valid to say that overloaded method get affected due to throwing of any sort of exception

Reply

Lokesh Gupta

July 14, 2014 at 9:10 am

I already said that "your overloaded method throws the same exception, a different exception or it simply does no throw any exception; no effect at all on method loading."

Reply

Leave a Comment

Name *

Email *

Website

☐ Add me to your newsletter and keep me updated whenever you publish new blog posts

Post Comment

Search …

HowToDoInJava

A blog about Java and related technologies, the best practices, algorithms, and interview questions.

Meta Links

- About Me
- Contact Us
- Privacy policy
- Advertise
- Guest Posts

Blogs

REST API Tutorial