**3.What are the other names for method overloading?**

* Method overloading also known as static polymorphism or compile time polymorphism because at compile time itself we can tell which method going to get executed based on method arguments.
* So method overloading also called as static binding.

Q=

1. package com.instanceofjava;
2. class sample{
4. int a;
5. int b;
7. **System.out.println("instance of java");**// compiler throws an error.
9. }
11. package com.instanceofjava;
12. class sample{
14. static int a=10;
16. **public static void main(String args[]){**
17. **System.out.println(a); // works fine,prints a value:10**
19. **}**
20. }

Q=**4. Can we change the return type of overridden method in sub class?**

* No. Return type must be same in super class and sub class.
* package MethodOverridingExamplePrograms;
* public class Super{
* void add(){
* System.out.println("Super class add method");
* }
* }
* package MethodOverridingInterviewPrograms;
* public class Sub extends Super{
* int add(){    //Compiler Error: The return type is incompatible with Super.add()
* System.out.println("Sub class add method");
* return 0;
* }
* }

**5.Can we change accessibility modifier in sub class overridden method?**

* Yes we can change accessibility modifier in sub class overridden method but should increase the accessibility if we decrease compiler will throw an error message.

| **Super class method** | **Subclass method** |
| --- | --- |
| Protected | protected, public |
| public | public |
| default | default , public |

**6.What happens if we try to decrease accessibility from super class to sub class?**

* Compile time error will come.

1. package MethodOverridingExamplePrograms;
2. public class Super{
4. public void add(){
5. System.out.println("Super class add method");
6. }
7. }
8. package MethodOverridingInterviewPrograms;
9. public class Sub extends Super{
11. void add(){ //Compiler Error: Cannot reduce the visibility of the inherited method
12. from Super
14. System.out.println("Sub class add method");
15. }
16. }

**7.Can we override a super class method without throws clause to with throws clause in the sub class?**

* Yes if super class method throws unchecked exceptions.
* No need if super class method throws checked exceptions. But it is recommended to add in sub class method in order to maintain exception messages.

1. **Rule:**

**1.Super class method  Not throwing any exceptions.**

* When super class method not throws ant exception,
* We can add throws unchecked exception in sub class overridden method.
* We can not add throws checked exception.

**2.Super class method  throws exceptions.** 

* If super class method throws checked exceptions sub class overridden method can throw same exception , sub class exception or no exception but can not declare parent exception.
* If super class method throws unchecked exceptions then no rules.

**Program #4: When super class method  throws unchecked exception then we can not add throws  checked exception in subclass overridden method.**

1. package exceptionhandlingmethodoverridingjava;
2. public class Super {
4. /\*\*
5. \* @author www.instanceofjava.com
6. \* @category throws in method overriding java
7. \*/

10. public void show() throws ArithmeticException{
12. System.out.println("Super class show() method");
14. }
16. }

1. package exceptionhandlingmethodoverridingjava;
2. public class Super {
4. /\*\*
5. \* @author www.instanceofjava.com
6. \* @category throws in method overriding java
7. \*/

10. public void show() throws IOException{ //compilation error
12. System.out.println("Super class show() method");
14. }
16. }

**9.What happens if we not follow these rules if super class method throws some exception.**

* Compile time error will come.

1. package MethodOverridingExamplePrograms;
2. public class Super{
4. public void add(){
5. System.out.println("Super class add method");
6. }
7. }
8. package MethodOverridingInterviewPrograms;
9. public class Sub extends Super{
11. void add() throws Exception{ //Compiler Error: Exception Exception is not compatible with
12. throws clause in Super.add()
13. System.out.println("Sub class add method");
14. }
15. }
16. package MethodOverridingExamplePrograms;
17. public class Super{
19. public void add(){
20. System.out.println("Super class add method");
21. }
22. }
23. package MethodOverridingInterviewPrograms;
24. public class Sub extends Super{
26. void add() throws NullPointerException{ // this method throws unchecked exception so no
27. isuues
28. System.out.println("Sub class add method");
29. }
30. }

**10.Can we change an exception of a method with throws clause from unchecked to checked while overriding it?**

* No. As mentioned above already
* If super class method throws exceptions in sub class if you want to mention throws  then use  same class  or its  sub class exception.
* So we can not change from unchecked to checked

**3. Can we overload**[**static**](https://www.java2blog.com/2013/12/static-keyword-in-java.html)**methods in java?**

**Answer:**

Yes, we can overload static methods in java but we can not override them.

**5. Can we change only return type while method overloading?**

**Answer:**  
You can not.If we change only return type, it will become ambiguous for compiler to figure out which method to call.That is why you can not change only return type.

**7. What are rules of method overriding?**

**Rules for method overriding:**

|  |  |
| --- | --- |
| **Arguments** | Must not change |
| **Return type** | Can’t change except for covariant (subtype) returns |
| **Access Modifier** | Must not be more restrictive. Can be less restrictive. |
| **Exceptions** | Can reduce or eliminate but must not throw new/broader checked exceptions |
| **Contructor** | Can not be overridden |
| **Static method** | Can not be overridden |
| **final method** | Can not be overridden |
| **Private method** | Can not be overridden |

**8.  Can you override**[**static**](https://www.java2blog.com/2013/12/static-keyword-in-java.html)**methods in java?**

**Answer:**  
No, you can not override static methods in java. Static methods belongs to class level not at object level.You can create static method with same name in child class and it won’t give you compilation error but it is called method hiding. You won’t get overriding behaviour with it.

But If you will try to override then**:**

* It will be method hiding.
* If super class have any static method then we can not define instance method with same name.
* If we override a static method then it will me method hiding.

Super sup = new Sub();

sup.add(); //If add is static method // it will call Super class add static method

sup.add(); // If add is instance method // It will call sub class add method.

**9. Can you override private methods in java?**

**Answer:**  
No, you can not override private methods in java. Private methods are not visible to child class, hence you can not override it , you can only hide it.

But If you will try to override then you does not get any exception.

**10. Can you override**[**final**](https://www.java2blog.com/2017/03/final-keyword-java-example.html)**methods?**

**Answer:**  
Because final methods are meant to be not overridden.You declare a method final because you don’t want it to be overridden in subclass.

But If you will try to override then you got a compile time error.’ Cannot override the final method from A’

**13. What are Covariant return type in java?**

Covariant return type means if subclass overrides any method, return type of this overriding method can be subclass of return type of base class method.

public class BaseClass {

public A m1() {

System.out.println("In BaseClass method");

return new A();

}

public static void main(String args[])

{

BaseClass b=new SubClass();

b.m1();

}

}

class SubClass extends BaseClass {

public B m1() {

System.out.println("In SubClass method");

return new B();

}

}

class A {

}

class B extends A {

}

**14. Predict output of below program:**



|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20 | public class MethodOverloadingExample {    public void methodTest(Object object)  {    System.out.println("Calling object method");  }    public void methodTest(String object)  {    System.out.println("Calling String method");  }    public static void main(String args[])  {    MethodOverloadingExample moe=new MethodOverloadingExample();    moe.methodTest(null);  }  } |

**Output:**



|  |  |
| --- | --- |
| 1  2  3 | Calling String method |

**Explanation:**  
When we have two overloaded version of same method, JVM will always call most specific method.

**3) In the below class, is ‘method’ overloaded or duplicated?**

[?](http://javaconceptoftheday.com/java-practice-questions-on-method-overloading-and-overriding/)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12 | public class MainClass  {      void method(int ... a)      {          System.out.println(1);      }        void method(int[] a)      {          System.out.println(2);      }  } |

[View Answer](http://javaconceptoftheday.com/java-practice-questions-on-method-overloading-and-overriding/#collapse3)

**Answer :**  
Duplicated. Because, var args (int … a) are nothing but the arrays. So here, (int … a) and (int[] a) are the same.

**4) Method signature consists of**

a) Method Name, Return Type and Number Of Arguments  
b) Access Modifier, Method Name and Types Of Arguments  
c) Method Name, Number Of Arguments, Types Of Arguments and Order Of Arguments  
d) Return Type, Access Modifier and Order Of Arguments

[View Answer](http://javaconceptoftheday.com/java-practice-questions-on-method-overloading-and-overriding/#collapse4)

**Answer :**  
c) Method Name, Number Of Arguments, Types Of Arguments and Order Of Arguments

**7) What will be the outcome of the below program?**

[?](http://javaconceptoftheday.com/java-practice-questions-on-method-overloading-and-overriding/)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24 | public class MainClass  {      double overloadedMethod(double d)      {          return d \*= d;      }        int overloadedMethod(int i)      {          return overloadedMethod(i \*= i);      }        float overloadedMethod(float f)      {          return overloadedMethod(f \*= f);      }        public static void main(String[] args)      {          MainClass main = new MainClass();            System.out.println(main.overloadedMethod(100));      }  } |

[View Answer](http://javaconceptoftheday.com/java-practice-questions-on-method-overloading-and-overriding/#collapse7)

**Answer :**  
It will throw java.lang.StackOverflowError at run time. Because, overloadedMethod(int) keeps calling itself.