What are things java consider for method signature?

Method name

Argument types

Example public static int m1(int i, float f) {}

Method signature: m1(int i, float f)

Return type is not part of method signature in Java.

Invalid signatures:

**public int** m1() {**return** 0;}

**public void** m1() {}

**public int** m1(**int** i) {**return** 0;}

**public void** m1(**int** i) {}

What is object type casting and what are rules applied?

When doing

A b = (C) d;

Compiler is going to do 2 things

1. Is the conversion is legal or not

2. It is the assignment is legal or not

JVM having 1 checking

Compiler Rule 1:

The type of `d` and `C` must have some relation either of child to parent or parent to child or same type. Otherwise we will compile time error saying inconvertiable types.

Object o = new String(“durga”);

StringBuffer sb = (StringBuffer) o; //Valid during compile time

Compiler Rule 2:

`C` must be either same or derived type of `A`, otherwise we will get `incompatiable types` error.

JVM(Runtime Checking) – Runtime object type of `d` must be either same or derived type of `C`.

What is overriding?

In overriding method resolution always takes care by JVM based on runtime object and hence overriding is also considered as runtime polymorphism or dynamic polymorphism or late binding

What are rules for oveririding?

Rule 1:

1. Method signature must be matched(method name and argument types)

2. Return types must be same, but this rule is applicable until 1.4 version. From 1.5 version we can take co-variant return types.

**class** Person1 {

Object m1() {

System.***out***.println(**"Person method"**);

**return null**;

}

}

**class** Employee1 **extends** Person1 {

@Override

String m1() {

System.***out***.println(**"Employee method"**);

**return null**;

}

}

According to 1.4v compiler above code is invalid, after 1.5v this code will works.

Above rule is not applicable for primitive types.

Rule 2:

Parent class private methods are not available to the child and hence overriding concept is not applicable for private methods.

Rule 3:

Final methods are not available to the child and hence overriding concepts are not possible.

We will compile time error

Refer OverridingTest1.java

Rule 4:

Parent class abstract methods, we should override in child class to provide implementation.

Refer OverridingTest2.java

Rule 5:

We can override non-abstract method as abstract

Refer OverridingTest3.java

Rule 6:

In overriding

final to non-final is not possible but non-final to final is possible

But following modifiers like synchronized, native, strictfp up and down change is possible.

Rule 7:

While overiding we can’t reduce the scope. But opposite is possible.

Refer: OverridingScopeTest.java

Rule 8:

If child class throws any checked exception compulsary parent class method should throw the same check exception or it parent. Otherwise we will get compile time error.

Valid:

1. public void m1() throws Exception

public void m1()

2. public void m1() throws Exception

public void m1() throws IOException

3. public void m1() throws IOException

public void m1() throws FileNotFoundException, EOFException

4. public void m1() throws IOException

public void m1() throw AE, NPE

Invalid:

1. public void m1()

public void m1() throws Exception

2. public void m1() throws IOException

public void m1() throws Exception

3. public void m1() throws IOException

public void m1() throws EOFException, InterruptedException

Rule 9:

Overiding with respect to static method:

We can override static method

Similarly we can’t override non-static as static.

Method hiding:

If both parent and child class methods are static then we won’t get compile time error. This concept is call method hiding

Rule 10:

Overriding w.r.t var-args methods

We can override var-arg method with anther var-arg method only. If we are trying to override with normal method, then it is not overriding, it is overloading

Rule 11:

Overriding w.r.t to variable names

Overriding concepts only applicable to method not for variable

What are difference between Overriding and Overloading?

|  |  |  |
| --- | --- | --- |
| Property | Overload | Override |
| Method name | Same | Same |
| Return type | No Restriction | Should be same or co-variants |
| Arguments | Must be different | Must be same |
| Method Signature | Must be different atleast the argument type | Must be same |
| Private, Static, final methods | Can be overloaded | Can’t able to override |
| Throw clause | No restriction | If child class method throws any checked exception, compulsary parent class method should throws same checked exception or it parent. |
| Access modifier | No restriction | The scope of access modifier cannot be reduced but we can increase the scope. |
| Method resolution | Compiler based | JVM based on Runtime |
| Also knowns as | Compile time Polymorphism | Runtime Polymorphism |