



Java Method Overriding Interview Questions

📅 December 17, 2017 👤 Posted by Abhi Andhariya ✍️ [Core Java](#), [Core Java Interview Questions](#), [Interview Questions](#)

Follow

Like 1.8K

Share

In our previous article, we have discussed tricky programming questions based on [method overloading](#). However, Method overriding programming questions are more interesting and requires depth understanding of the concept than that of [method overloading](#). Before moving to programming questions, let's understand the basic concepts of method overriding.

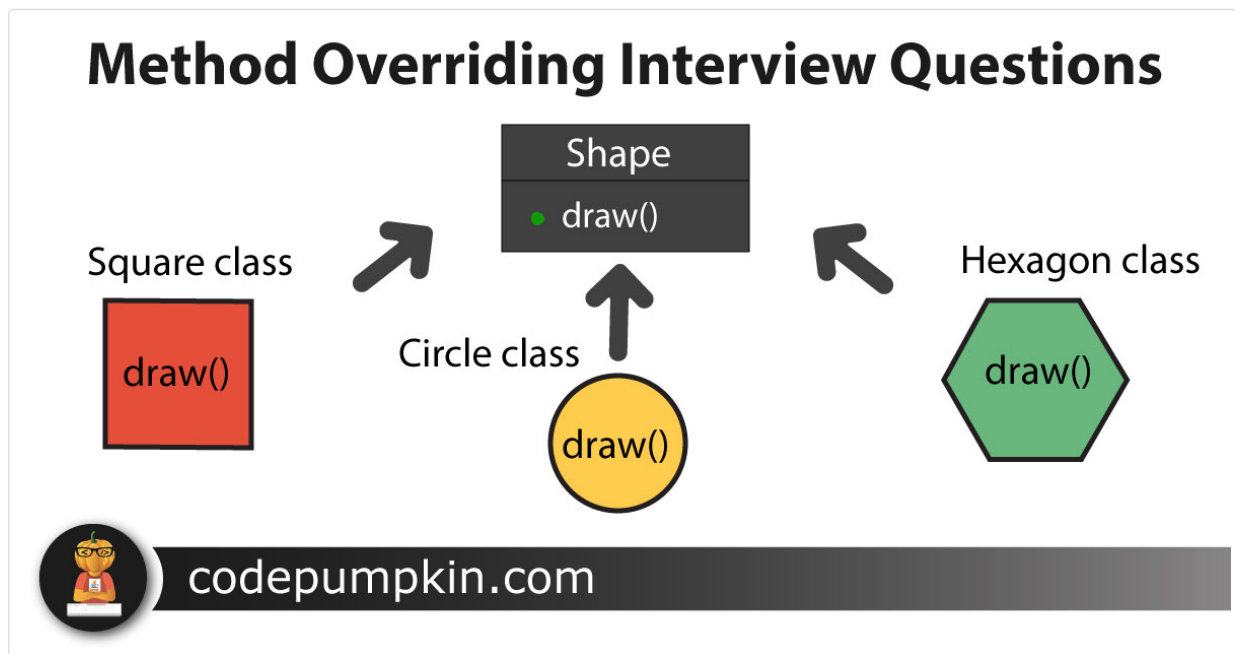
Method Overriding

Using method overriding, child class can provide its own implementation of the method which is already present in the parent class or declared in parent interface.

In other words, when method in the sub class has the same name, same parameters and same return type (or co-variant return type) as parent class or interface, then we can say that child class method has overridden the parent class method.

For Example,

Let's understand this using simple example.



We have two classes : parent class `Shape` and child class `Circle` which extends `Shape` class.

Both the class has common method `draw()`. `Circle` class has provided its own implementation of `draw()` method. In other words, it has overridden `draw()` method of `Shape` class.

Along with `draw()` method, `Shape` class also contains `fill()` method which has not been overridden by `Circle` class. But this method will be inherited to `Circle` class with default implementation.

Purpose of method overriding is very clear here. `Circle` class wants to provide its own implementation of `draw()` method so that when it calls this method, it will print '**Circle**' instead of '**Shape**'.



```
6         s.draw();
7         s.fill();
8     }
9 }
10
11 class Shape{
12     public void draw()
13     {
14         System.out.println("Shape");
15     }
16
17     public void fill()
18     {
19         System.out.println("Shape Filled with color");
20     }
21 }
22
23
24 class Circle extends Shape{
25     public void draw()
26     {
27         System.out.println("Circle");
28     }
29 }
30
31 class Square extends Shape{
32     public void draw()
33     {
34         System.out.println("Square");
35     }
36 }
37
38 class Hexagon extends Shape{
39     public void draw()
40     {
41         System.out.println("Hexagon");
42     }
43 }
```

output

```
1 Circle
2 Shape Filled with color
```

Why Is It Known As Runtime Polymorphism?

In above example, we have created reference of type `Shape`, but object of type `Circle`. When we call method `draw()`, jvm decides method of which class needs to be called at runtime.

In short, if we create a object of child class and if child class has overridden the method, then child class method will be called e.g. `draw()`. If method has not been overridden in the child class, then parent class method will be called e.g. `fill()`.

You can also refer our article on [method overloading](#) to know more about Compile-Time Polymorphism or static binding.

Programming Interview Questions

We have divided the questions into six categories so that it would be easier for you to understand the concepts.

1. [Access Modifiers](#)
2. [static methods](#)
3. [Member Variables](#)
4. [Exception Handling](#)
5. [Return types](#)
6. [Method Parameters](#)

We have listed down all such questions and their answers with explanation. But we suggest you guys to first try solving and guessing the answer of each question and then read its answer.

Access Modifiers

Question 1) What would be the output of below code?



```

6      }
7    }
8
9    class Shape{
10     protected void draw()
11     {
12         System.out.println("Shape");
13     }
14 }
15
16 class Circle extends Shape{
17     public void draw()
18     {
19         System.out.println("Circle");
20     }
21 }

```

Here method in parent class has `protected` scope, but in child class it is `public`. Will method overriding work here? Will it print **Shape** or **Circle**?

Answer: Method overriding has nothing to do with [access modifier scopes](#). It will print **Circle** in above code.

Question 2) As a follow up question, interviewer may ask you : What will happen if I change scope of `draw()` method from `protected` to `private` in `Shape` class?

Answer : It will give you a compile time error in `main()` method where you are calling `s.draw()` as we can not call the `private` method from outside the class.

Question 3) What would be the output of below code?

```

1 public class PumpkinDemo {
2
3     public static void main(String[] args) {
4         Shape s = new Circle();
5         s.draw();
6     }
7 }
8
9 class Shape{
10     public void draw()
11     {
12         System.out.println("Shape");
13     }
14 }
15
16 class Circle extends Shape{
17     private void draw()
18     {
19         System.out.println("Circle");
20     }
21 }

```

Answer:

Above code will give you compile time error as we cannot reduce the visibility or scope of the inherited method from parent class i.e. `public void draw()` to `private void draw()`.

Static Methods

Question 4) This question is not related to inheritance or method overriding, but it is one of our favorite question. Many programmer gives confused look when we ask them this one. Have a look at below code and guess the output.

```

1 public class PumpkinDemo {
2
3     public static void main(String[] args) {
4         Shape s = null;
5         s.draw();
6     }
7 }
8
9 class Shape{
10     public static void draw()
11     {
12         System.out.println("Shape");
13     }
14 }

```

Will above code give `NullPointerException`? Or it will print **Shape** as output?

Answer : It won't give `NullPointerException` as `draw()` is a static method and compiler will replace reference variable with class name i.e. `Shape.draw()`

Question 5) What would be the output of below code?



```

6      }
7    }
8
9    class Shape{
10      public static void draw()
11      {
12        System.out.println("Shape");
13      }
14    }
15
16    class Circle extends Shape{
17      public static void draw()
18      {
19        System.out.println("Circle");
20      }
21    }

```

Answer:

It will print **Shape** as output. Method overriding happens only with instance methods. Static methods are attached to class and compiler converts reference variable to class name i.g. Shape.draw()

Question 6) What will happen if we will remove `static` keyword from the `draw()` method of `Circle` class. `Shape` class still contains the `public static void draw()` method.

Answer: It will give compile-time error saying 'This instance method cannot override the static method from Shape'.

Member Variables

Question 7) Guess the output of below code

```

1  public class PumpkinDemo {
2
3      public static void main(String[] args) {
4          Shape s = new Circle();
5          System.out.println(s.name);
6      }
7  }
8
9  class Shape{
10     String name = "Shape";
11 }
12
13 class Circle extends Shape{
14     String name = "Circle";
15 }

```

Output:

```
1 Shape
```

Answer : Member variables cannot be overridden. In other words, Variables are resolved at compile-time and methods at run-time.

Exception Handling

Question 8) Can overridden method throw different exception than the one being thrown in parent class method. For Example, Will below code compile successfully?

```

1  import java.io.FileNotFoundException;
2  import java.io.IOException;
3
4  public class PumpkinDemo {
5
6      public static void main(String[] args) throws IOException{
7          Shape s = new Circle();
8          s.draw();
9      }
10 }
11
12 class Shape{
13     public void draw() throws IOException
14     {
15         System.out.println("Shape");
16     }
17 }
18
19 class Circle extends Shape{
20     public void draw() throws FileNotFoundException
21     {
22         System.out.println("Circle");
23     }
24 }

```

Answer:

While overriding a method, you can compress the scope of checked exception but you cannot widen it. Also you can not throw any other checked exception which is not being thrown in parent class method.

Here, `FileNotFoundException` is a child class of `IOException`. So, above code will compile successfully and it will give **Circle** as output.



Code Pumpkin

Get your idea online.

Bring your business online. Our 24/7 guides can help you.

in.godaddy.com

Shc

If we change `FileNotFoundException` to generic `Exception` in above code, then it will give compile time error saying '`Exception` `Exception` is not compatible with throws clause in `Shape.draw()`'.

Question 9) Will below code compile successfully?

```

1 public class PumpkinDemo {
2
3     public static void main(String[] args){
4         Shape s = new Circle();
5         s.draw();
6     }
7 }
8
9 class Shape{
10     public void draw() throws ArithmeticException
11     {
12         System.out.println("Shape");
13     }
14 }
15
16 class Circle extends Shape{
17     public void draw() throws RuntimeException
18     {
19         System.out.println("Circle");
20     }
21 }
```

Answer: Yes. Overridden methods can throw any `RuntimeException` irrespective of its scope unlike checked exception.

Return Type

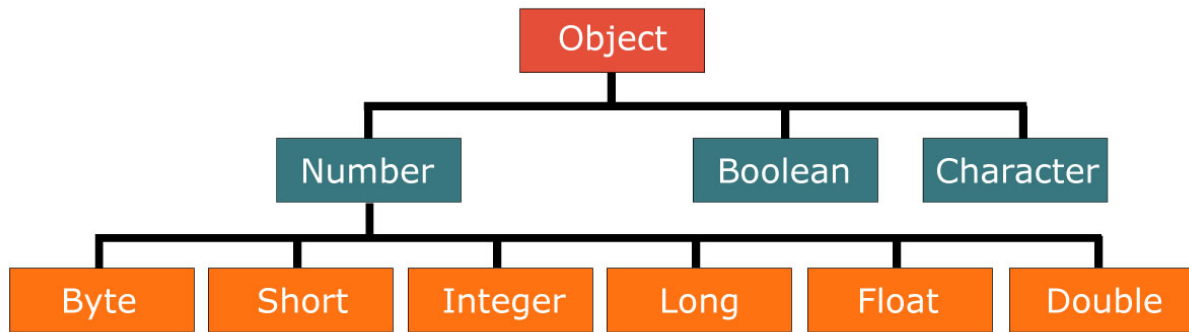
Question 10) Can a return type be different in overridden method? Guess the output of below code.

```

1 public class PumpkinDemo {
2
3     public static void main(String[] args){
4         Parent p = new Child();
5         p.testMethod();
6     }
7 }
8
9 class Parent{
10     public Number testMethod()
11     {
12         System.out.println("Parent");
13         return 0;
14     }
15 }
16
17 class Child extends Parent{
18     public Integer testMethod()
19     {
20         System.out.println("Child");
21         return 0;
22     }
23 }
```

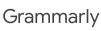
Answer: Above code will compile successfully and prints **Child** as output. So is different return type allowed in method overriding?

Well `Number` is a parent class of `Integer Wrapper class`, and that is why above code compiled successfully. They are called covariant return types. You can check Wrapper classes Hierarchy in below image.



The **covariant return types** are newly introduced since Java 5.0, and used during **method overriding**. **Covariant return type** allows us to change the **return type** of the **overriding method** in the subclass; however this **return type** in subclass **method** must be a subtype of super class **method return type**.

Below two combinations will give you compile time errors:



Free Intelligent Writing Tool

Write confidently knowing that Grammarly is there you share your perspective

[Learn](#)

- 1) Parent class method return type : Integer,
Child class method return type : Number or Long
- 2) Parent class method return type : String,
Child class method return type : Number or Long

Method Parameters

Question 11) Here are the last two questions of this article. Guess the output of below code:

```

1 public class PumpkinDemo {
2
3     public static void main(String[] args){
4         Parent p = new Child();
5         p.testMethod(0);
6     }
7 }
8
9 class Parent{
10     public void testMethod(Number n)
11     {
12         System.out.println("Parent");
13     }
14 }
15
16 class Child extends Parent{
17     public void testMethod(Integer n)
18     {
19         System.out.println("Child");
20     }
21 }
  
```

Confused? Does java allows covariant method parameters? Is this method overriding?

Answer:



Question 12) What will be the output if we change main() method as below

```
1 public static void main(String[] args){
2     Child p = new Child();
3     p.testMethod(0);
4 }
```

Output:

1 Child

Interesting, isn't it?

That's all for this topic. If you guys have any suggestions or queries, feel free to drop a comment. We would be happy to add that in our post. You can also contribute your articles by creating contributor account [here](#).

Happy Learning 😊

If you like the content on CodePumpkin and if you wish to do something for the community and the planet Earth, you can donate to our campaign for planting more trees at [CodePumpkin Cauvery Calling Campaign](#).

We may not get time to plant a tree, but we can definitely donate ₹42 per Tree.

About The Author



Abhi Andhariya

[Visit Full Profile](#)

Surviving Java Developer, Passionate Blogger, Table Tennis Lover, Bookworm, Occasional illustrator and a big fan of Joey Tribbiani, The Walking Dead and Game of Thrones....!!



Tags: [Core Java](#), [Java](#), [Java keywords](#), [OOPs](#), [Overriding](#), [Polymorphism](#)

Comments And Queries

If you want someone to read your code, please put the code inside `<pre><code>` and `</code></pre>` tags. For example:

```
<pre><code class="java">
String foo = "bar";
</code></pre>
```

🔒 ✕

Free Intelligent Writing Tool

Write confidently knowing that Grammarly is there you share your perspective

Grammarly

[Learn](#)

For more information on supported HTML tags in Disqus comment, [click here](#).



Code Pumpkin



Join the discussion...

LOG IN WITH

OR SIGN UP WITH DISQUS [?](#)

Name

[Rahul Jangra](#) • 3 years ago

Hi, I am not able to understand answer to 11th question. Integer class is subtype of Number class, so shouldn't the output be 'Child' ?

^ | v • Reply • Share ›

[Pumpkin](#) Mod → [Rahul Jangra](#) • 3 years ago

Java considers methods with co-variant return types as method overloading and not overriding and as we have specified

Search ...

Total Posts : 124

Like Share 1.8K people like this.
[Sign Up](#) to see what

Follow 1,060 followers

YouTube 120

[Follow](#)

[Contribute Your Articles](#)

Categories

- [Algorithms](#) (10)
- [C / C++](#) (2)
- [Core Java](#) (48)
 - [Java MultiThreading](#) (11)
- [Data Structure](#) (10)
- [Design Patterns](#) (15)
- [HTML](#) (1)
- [Javascript](#) (8)
- [Pumpkin Practice](#) (10)
- [Python](#) (15)
- [SQL](#) (2)

Tag Cloud

[Algorithms](#) [Collection](#) [Framework](#) [Collections](#) [Concurrency](#) [Core Java](#) [Creational](#) [Design Patterns](#) [DataStructure](#) [Design Patterns](#) [HashMap](#) [Java](#) [Java8](#) [Java keywords](#) [javascript](#) [JVM](#) [internals](#) [Multithreading](#) [python](#) [Python](#) [Tutorials](#) [Synchronization](#) [Thread](#) [x vs y](#)

Interview Questions

- [Core Java Interview Questions](#) (19)
- [Multithreading Interview Questions](#) (7)
- [Programming](#) (7)
- [SQL Interview Questions](#) (2)

Popular Posts



Code Pumpkin

- > Snakes N Ladders | Java Program Implementation
- > Program to find Unique Array Element

Interview Experiences

- > CGI (1)
- > High Radius Technologies (1)
- > Morgan Stanley (6)
- > Sapient Global Markets (3)
- > ServiceNow (1)
- > Synechron (1)
- > Wissen Infotech (2)

Recent Posts

- > HTML5 Autocomplete Attribute
- > CGI Interview Questions - Set 1
- > File I/O | Python
- > Exception Handling | Python
- > while loop | Python Flow Control

Cauvery Calling

The banner features a dark blue background with a white border. On the left, there is a logo for 'RALLY FOR RIVERS INDIA'S LIFELINES' and 'Action Now'. Below this, the text 'Cauvery Calling' is written in large, bold, white letters. At the bottom left, the 'isha' logo is visible. On the right side, the text 'ACTION NOW to Save Cauvery' is displayed in white, followed by '₹42 per tree' in a larger font. Below that, 'CauveryCalling.org' and '#CauveryCalling' are written in white.

We may not get time to plant a tree, but we can definitely donate ₹42 per Tree.

Join us to save the planet Earth by donating at [CodePumpkin Cauvery Calling Campaign](#).

Like Us On Facebook



Like Share 1.8K people like this. [Sign Up](#) to see what your friends like.



Categories

Algorithms (10)
C / C++ (2)
Core Java (48)
 Java MultiThreading (11)
Data Structure (10)
Design Patterns (15)
HTML (1)
Interview Experience (15)
 CGI (1)
 High Radius Technologies (1)
 Morgan Stanley (6)
 Sapient Global Markets (3)
 ServiceNow (1)
 Synchron (1)
 Wissen Infotech (2)
Javascript (8)
Pumpkin Practice (10)
Python (15)
SQL (2)

Interview Questions

Core Java Interview Questions (19)
Multithreading Interview Questions (7)
Programming (7)
SQL Interview Questions (2)

Popular Posts

Tic Tac Toe | Java Program Implementation
Synchron Interview Questions – Set 1
How is HashSet implemented internally in Java?
Hashtable vs SynchronizedMap vs ConcurrentHashMap
Program to find Unique Array Element

Privacy Policy

Terms And Condtions

Disclaimer

No part of this blog maybe copied or reproduced or reused or republished in any way, except to share either using the share feature of LinkedIn, Facebook, Twitter or any other Social Media Sites or posting a direct link to this blog - An excerpt from the blog may be quoted while sharing it in the above mentioned manner. Any other form of reuse, must be only after explicit written consent of the CodePumpkin.

