# Nikos' Java blog

quick & easy Java tutorials

# SCJP Mock exam for Generics

i
72 Votes

A collection of structured questions for Generics, under the scope of Sun Certified Java Programmer for Java SE 6.

If you are busy, just go through questions 1-40.

| questions | topic |
|---|---|
| 1-24 | assignment and hierarchy |
| 25-40 | add and get elements from a generic collection |
| 41-50 | overloading and overriding |
| 51-60 | various details |
| 61-72 | declaration of generic classes |
| | () |

1. Will this code compile successfully? (1 correct answer)

```
List<Number> list1 = null;
List<Integer> list2 = null;
list1 = list2;
```

   1. Yes.
   2. No.

2. Will this code compile successfully? (1 correct answer)

```
List<Number> list1 = null;
List<Integer> list2 = null;
list2 = list1;
```

   1. Yes.
   2. No.

3. Will this code compile successfully? (1 correct answer)

```
List<? extends Number> list1 = null;
List<Integer> list2 = null;
list1 = list2;
```

   1. Yes.
   2. No.

4. Will this code compile successfully? (1 correct answer)

```
List<? extends Number> list1 = null;
List<Integer> list2 = null;
list2 = list1;
```

   1. Yes.
   2. No.

5. Will this code compile successfully? (1 correct answer)

```
List<Number> list1 = null;
List<? super Integer> list2 = null;
list1 = list2;
```

   1. Yes.
   2. No.

6. Will this code compile successfully? (1 correct answer)

```
List<Number> list1 = null;
List<? super Integer> list2 = null;
list2 = list1;
```

   1. Yes.
   2. No.

7. Will this code compile successfully? (1 correct answer)

```
SortedSet<? super Number> set1 = null;
SortedSet<Integer> set2 = null;
set1 = set2;
```

1. Yes.
2. No.

8. Will this code compile successfully? (1 correct answer)

```
SortedSet<? super Number> set1 = null;
SortedSet<Integer> set2 = null;
set2 = set1;
```

1. Yes.
2. No.

9. Will this code compile successfully? (1 correct answer)

```
SortedSet<Number> set1 = null;
SortedSet<? extends Integer> set2 = null;
set1 = set2;
```

1. Yes.
2. No.

10. Will this code compile successfully? (1 correct answer)

```
SortedSet<Number> set1 = null;
SortedSet<? extends Integer> set2 = null;
set2 = set1;
```

1. Yes.
2. No.

11. Will this code compile successfully? (1 correct answer)

```
Queue<? extends Number> q1 = null;
Queue<? super Integer> q2 = null;
q1 = q2;
```

1. Yes.
2. No.

12. Will this code compile successfully? (1 correct answer)

```
Queue<? extends Number> q1 = null;
Queue<? super Integer> q2 = null;
q2 = q1;
```

1. Yes.
2. No.

13. Will this code compile successfully? (1 correct answer)

```
Queue<? super Number> q1 = null;
Queue<? extends Integer> q2 = null;
q1 = q2;
```

1. Yes.
2. No.

14. Will this code compile successfully? (1 correct answer)

```
Queue<? super Number> q1 = null;
Queue<? extends Integer> q2 = null;
q2 = q1;
```

1. Yes.
2. No.

15. Will this code compile successfully? (1 correct answer)

```
Queue<?> q1 = null;
Queue<Integer> q2 = null;
q1 = q2;
```

1. Yes.
2. No.

16. Will this code compile successfully? (1 correct answer)

```
LinkedList<?> list1 = null;
LinkedList<Integer> list2 = null;
list2 = list1;
```

1. Yes.
2. No.

17. Will this code compile successfully? (1 correct answer)

```
LinkedList<?> list1 = null;
LinkedList<? extends Integer> list2 = null;
list1 = list2;
```

1. Yes.
2. No.

18. Will this code compile successfully? (1 correct answer)

```
LinkedList<?> list1 = null;
LinkedList<? extends Integer> list2 = null;
list2 = list1;
```

1. Yes.

2. No.

19. Will this code compile successfully? (1 correct answer)

```
LinkedList<?> list1 = null;
LinkedList<? super Integer> list2 = null;
list1 = list2;
```

1. Yes.
2. No.

20. Will this code compile successfully? (1 correct answer)

```
LinkedList<?> list1 = null;
LinkedList<? super Integer> list2 = null;
list2 = list1;
```

1. Yes.
2. No.

21. Will this code compile successfully? (1 correct answer)

```
PriorityQueue queue1 = null;
PriorityQueue<Integer> queue2 = null;
queue1 = queue2;
```

1. Yes, without warnings.
2. Yes, with a warning.
3. No.

22. Will this code compile successfully? (1 correct answer)

```
PriorityQueue queue1 = null;
PriorityQueue<Integer> queue2 = null;
queue2 = queue1;
```

1. Yes, without warnings.
2. Yes, with a warning.
3. No.

23. Will this code compile successfully? (1 correct answer)

```
PriorityQueue<?> queue1 = null;
PriorityQueue queue2 = null;
queue1 = queue2;
```

1. Yes, without warnings.
2. Yes, with a warning.
3. No.

24. Will this code compile successfully? (1 correct answer)

```
PriorityQueue<?> queue1 = null;
PriorityQueue queue2 = null;
queue1 = queue2;
```

    1. Yes, without warnings.
    2. Yes, with a warning.
    3. No.

25. Will this code compile successfully? (1 correct answer)

```
Set<Integer> set = new TreeSet<Integer>();
set.add(10);
```

    1. Yes.
    2. No.

26. Will this code compile successfully? (1 correct answer)

```
Set<Integer> set = new TreeSet<Integer>();
set.add((int)1.0f);
```

    1. Yes.
    2. No.

27. Will this code compile successfully? (1 correct answer)

```
Set<Integer> set = new TreeSet<Integer>();
set.add(10L);
```

    1. Yes.
    2. No.

28. Will this code compile successfully? (1 correct answer)

```
Set<Integer> set = new TreeSet<Integer>();
int number = (short)10;
set.add(number);
```

    1. Yes.
    2. No.

29. Will this code compile successfully? (1 correct answer)

```
Set<Number> set = new TreeSet<Integer>();
set.add(10L);
```

    1. Yes.
    2. No.

30. Will this code compile successfully? (1 correct answer)

```
NavigableSet<?> set = new TreeSet<Object>();
set.add(new Object());
```

    1. Yes.
    2. No.

31. Will this code compile successfully? (1 correct answer)

```
NavigableSet<? super Object> set = new TreeSet<Object>();
set.add(new Object());
```

    1. Yes.
    2. No.

32. Will this code compile successfully? (1 correct answer)

```
NavigableSet<? extends Object> set = new TreeSet<Object>();
set.add(new Object());
```

    1. Yes.
    2. No.

33. Will this code compile successfully? (1 correct answer)

```
NavigableSet<? extends String> set = new TreeSet<String>();
set.add("string");
```

    1. Yes.
    2. No.

34. Will this code compile successfully? (1 correct answer)

```
NavigableSet<? super String> set = new TreeSet<String>();
set.add("string");
```

    1. Yes.
    2. No.

35. Will this code compile successfully? (1 correct answer)

```
NavigableSet<? super String> set = new TreeSet<String>();
set.add(new Object());
```

    1. Yes.
    2. No.

36. Will this code compile successfully? (1 correct answer)

```
List<? extends Integer> list = new ArrayList<Integer>();
for (Integer element : list) {
        System.out.println(element);
}
```

1. Yes.
2. No.

37. Will this code compile successfully? (1 correct answer)

```
List<? extends Integer> list = new ArrayList<Integer>();
Integer first = list.get(0);
```

1. Yes.
2. No.

38. Will this code compile successfully? (1 correct answer)

```
List<? super Integer> list = new ArrayList<Integer>();
for (Integer element : list) {
        System.out.println(element);
}
```

1. Yes.
2. No.

39. Will this code compile successfully? (1 correct answer)

```
List<? super Integer> list = new ArrayList<Integer>();
Integer first = list.get(0);
```

1. Yes.
2. No.

40. Will this code compile successfully? (1 correct answer)

```
List<? super Integer> list = new ArrayList<Integer>();
Object first = list.get(0);
```

1. Yes.
2. No.

41. Will this code compile successfully? (1 correct answer)

```
import java.util.*;

class Test {
    void say(Set<Double> set) {
    }
    void say(SortedSet<Double> set) {
    }
}
```

1. Yes.
2. No.

42. Will this code compile successfully? (1 correct answer)

```
import java.util.*;

class Test {
    void say(Set<Double> set) {
    }
    void say(Set<Boolean> set) {
    }
}
```

1. Yes.
2. No.

43. Will this code compile successfully? (1 correct answer)

```
import java.util.*;

class Test {
    void say(Set<Double> set) {
    }
    void say(Set<Double>... set) {
    }
}
```

1. Yes.
2. No.

44. Consider these classes.

```
import java.util.*;

class Parent {
        void say(List<String> list) {
                System.out.println("parent");
        }
}
class Child extends Parent {
        void say(List list) {
                System.out.println("child");
        }
}
```

What happens when this code is compiled and executed? (1 correct answer)

```
public static void main(String[] java) {
        Child c = new Child();
        c.say(new LinkedList<String>());
}
```

1. It prints “child”.
2. It prints “parent”.
3. Compilation fails.

45. Consider these classes.

```
import java.util.*;

class Parent {
        void say(List<String> list) {
                System.out.println("parent");
        }
}
class Child extends Parent {
        void say(List list) {
                System.out.println("child");
        }
}
```

What happens when this code is compiled and executed? (1 correct answer)

```
public static void main(String[] java) {
        Child c = new Child();
        c.say(new LinkedList<List<Boolean>>());
}
```

1. It prints “child”.
2. It prints “parent”.

3. Compilation fails.

46. Consider these classes.

```
import java.util.*;

class Parent {
        void say(List<String> list) {
                System.out.println("parent");
        }
}
class Child extends Parent {
        void say(List list) {
                System.out.println("child");
        }
}
```

What happens when this code is compiled and executed? (1 correct answer)

```
public static void main(String[] java) {
        Parent c = new Child();
        c.say(new LinkedList<String>());
}
```

1. It prints “child”.
2. It prints “parent”.
3. Compilation fails.

47. Consider these classes.

```
import java.util.*;

class Parent {
        void say(List<String> list) {
                System.out.println("parent");
        }
}
class Child extends Parent {
        void say(List list) {
                System.out.println("child");
        }
}
```

What happens when this code is compiled and executed? (1 correct answer)

```
public static void main(String[] java) {
        Parent c = new Child();
        c.say(new LinkedList<Long>());
}
```

1. It prints “child”.
2. It prints “parent”.
3. Compilation fails.

48. Will this code compile successfully? (1 correct answer)

```
import java.util.*;

class Parent {
        void say(List<String> list) {
                System.out.println("parent");
        }
}
class Child extends Parent {
        void say(List<Integer> list) {
                System.out.println("child");
        }
}
```

1. Yes.
2. No.

49. Will this code compile successfully? (1 correct answer)

```
import java.util.*;

class Parent {
        void say(List<? extends Number> list) {
                System.out.println("parent");
        }
}
class Child extends Parent {
        void say(List<Integer> list) {
                System.out.println("child");
        }
}
```

1. Yes.
2. No.

50. Will this code compile successfully? (1 correct answer)

```
import java.util.*;

class Parent {
        void say(List list) {
                System.out.println("parent");
        }
}
class Child extends Parent {
        void say(List<Integer> list) {
                System.out.println("child");
        }
}
```

1. Yes.
2. No.

51. Will this code compile successfully? (1 correct answer)

```
Object set = new TreeSet<Integer>();
boolean flag = set instanceof NavigableSet<Integer>;
```

1. Yes.
2. No.

52. What is the output of the following code? (1 correct answer)

```
List<? extends String> list1 = new ArrayList<String>();
List<? super String> list2 = new ArrayList<String>();
List<Integer> list3 = new ArrayList<Integer>();
List list4 = new ArrayList();

if (list1 instanceof List &&
    list2 instanceof List &&
    list3 instanceof List &&
    list4 instanceof List) {
    System.out.println("yes");
}
```

1. It prints “yes”.
2. It prints nothing.

53. Will this code compile successfully? (1 correct answer)

```
Class c = ArrayList<Integer>.class;
```

1. Yes.
2. No.

54. Will this code compile successfully? (1 correct answer)

```
Class c = new ArrayList<Integer>().getClass();
```

1. Yes.
2. No.

55. Will this code compile successfully? (1 correct answer)

```
new ArrayList<?>();
```

1. Yes.
2. No.

56. Will this code compile successfully? (1 correct answer)

```
new TreeMap<String, ? super Integer>();
```

1. Yes.
2. No.

57. Will this code compile successfully? (1 correct answer)

```
new ArrayList<Set<?>>();
```

1. Yes.
2. No.

58. Will this code compile successfully? (1 correct answer)

```
class Test extends ArrayList<? extends Number> {
}
```

1. Yes.
2. No.

59. Will this code compile successfully? (1 correct answer)

```
class Test implements Comparable<?> {
    public int compareTo(Comparable<?> object) {
        return 0;
    }
}
```

1. Yes.
2. No.

60. Will this code compile successfully? (1 correct answer)

```
class Test implements Comparable<Comparable<?>> {
    public int compareTo(Comparable<?> object) {
        return 0;
    }
}
```

1. Yes.
2. No.

61. Will this code compile successfully? (1 correct answer)

```
class Test {
<T> T getFirst(List<T> list) {
    return list.get(0);
}
}
```

1. Yes.
2. No.

62. Will this code compile successfully? (1 correct answer)

```
class Test {
static <T> T getFirst(List<T> list) {
    return list.get(0);
}
}
```

1. Yes.
2. No.

63. Will this code compile successfully? (1 correct answer)

```
class Test <T> {
T getFirst(List<T> list) {
    return list.get(0);
}
}
```

1. Yes.
2. No.

64. Will this code compile successfully? (1 correct answer)

```
class Test <T> {
static T getFirst(List<T> list) {
    return list.get(0);
}
}
```

1. Yes.
2. No.

65. Will this code compile successfully? (1 correct answer)

```
class Test <T> {
T instance;
Test(T instance) {
    this.instance = instance;
}
}
```

1. Yes.
2. No.

66. Will this code compile successfully? (1 correct answer)

```
class Test <T> {
Test(T my) {
    boolean b = (my instanceof T);
}
}
```

1. Yes.
2. No.

67. Will this code compile successfully? (1 correct answer)

```
class Test <T> {
void test(T method) {
    Object my = (T)method;
}
}
```

1. Yes.
2. No.

68. Will this code compile successfully? (1 correct answer)

```
class Test <T> {
void test() {
    NavigableMap map = new TreeMap<String, T>();
}
}
```

1. Yes.
2. No.

69. Will this code compile successfully? (1 correct answer)

```
class Test <T> {
private T[] array = null;
}
```

1. Yes.
2. No.

70. Will this code compile successfully? (1 correct answer)

```
class Test <T> {
private T[] array = new T[7];
}
```

1. Yes.
2. No.

71. Will this code compile successfully? (1 correct answer)

```
class Test<String> {
    String my = "Hello!";
}
```

1. Yes.
2. No.

72. Will this code compile successfully? (1 correct answer)

```
class Test<String> {
    String my;
    public Test(String my) {
        this.my = my;
    }
    public String get() {
        return my;
    }
}

public class RunTest {
    public static void main(String[] args) {
        Integer i = new Test<Integer>(1).get();
        System.out.println(i.getClass());
    }
}
```

1. Yes.
2. No.

## Answers

1. b
2. b
3. a
4. b
5. b
6. a
7. b
8. b
9. b
10. b
11. b
12. b
13. b
14. b
15. a
16. b
17. a
18. b
19. a
20. b
21. a
22. b
23. a
24. a
25. a
26. a
27. b
28. a
29. b
30. b
31. a
32. b
33. b
34. a
35. b
36. a
37. a
38. b
39. b
40. a
41. a

42. b
43. a
44. a
45. a
46. a
47. c
48. b
49. b
50. b
51. b
52. a
53. b
54. a
55. b
56. b
57. a
58. b
59. b
60. a
61. a
62. a
63. a
64. b
65. a
66. b
67. a
68. a
69. a
70. b
71. b
72. a

This entry was posted on Thursday, October 9th, 2008 at 8:32 pm and is filed under SCJP 6, Training. You can follow any responses to this entry through the RSS 2.0 feed. You can leave a response, or trackback from your own site.

## 26 Responses to *SCJP Mock exam for Generics*

**Free SCJP Mock exams « Niko's java blog says:**
30 December 2008 at 7:47 pm
[…] java blog quick & easy Java tutorials « SCJP Mock exam for Generics JBoss integration with JDeveloper 11g […]

**Aruna says:**

2 February 2009 at 11:39 am
Hii Niko This an Excellent Practice Stuff.it's helped me a lot in grasping the Concepts of Generics fully.
Thnaks A Lot.

**JacQ says:**
9 March 2009 at 4:02 pm
Thx ! ☺ nice one

**himalay says:**
11 March 2009 at 3:50 pm
kudos to you

**Ahsan Jamshaid says:**
23 March 2009 at 1:24 pm
Hi Niko,
this is really excellent stuff. i liked it and i really appreciate your effort.

Thanks Buddy!..

**Paolo says:**
10 April 2009 at 11:06 am
Hi, thanks for this!

Questions 23 and 24 and respective answers are identical. Is it intended?

**ajith says:**
12 April 2009 at 8:27 am
hi Niko,,,
thanks a lot …. i really enjoyed with these questions …
well done

**unexpected compilation error with Generics : java cms says:**
15 April 2009 at 6:11 am
[…] This is from the practice exam for Generics on > NikoJava […]

**The true about the question 72?? says:**
24 October 2009 at 1:55 am
Hi everyone

In question 72 the correct answer is b), i don't know the exact reason; I supposed is a name colide because i tried to compile and the result is:

non-static class String cannot be referenced from a static context

Thanks for your time

**Shan says:**
23 February 2010 at 8:33 pm

Hi Nikos,
Thanks for this wonderful effort.
To add cream on top of this, if you could display the rules/logics/traps that you are testing in your questions, it would help us a lot in revising the concepts.

**Tatiana says:**
5 May 2010 at 8:35 am
48, 49 and 50 the answer is a! They compile just fine!

and about question 72, it compiles fine, so, it's correct.

**Mohamed Farouk says:**
18 May 2010 at 5:39 pm
Hello Guys
Finally cracked this all confusing generic collection type assignement. If you use this logic you can crack nicko questions on generics (1-24) without any problem as well understanding the concepts of generics assignments.

Solutions:
Look at the assignments L1 = L2
Read like this
I accept L1 = You have L2
Question No Left hand side (I accept) Right Hand side is I have Can I accept Answer
1 I accept Number I have Integer No No
2 I accept Integer I have Number No No
3 I accept subclasses of Number I have Integer Yes Yes
4 I accept Integer I have any subclasses of Number No No
5 I accept Number I have any super class of Integer (Number/Object) No No
6 Any super classes of Integer (Number or Object) I have Number Yes Yes
7 Any super classes of Number(Object) I have Integer No No
8 Integer I have any super classes of Number(Object) No No
9 Number Any subclasses of Integer No No
10 Any subclass on Integer Number No No
11 Any subclass of Number Super classes of Integer(Object/Number) No No
12 Any super class of Integer(Number/Object) Subclasses of Number No No
13 Only super classes of Number(Object) Integer No No
14 Subclasses of Integer Any Super class of Number (Object) No No
15 Anything Integer Yes Yes
16 Integer Anything No No
17 Anything Subclasses of Integer Yes Yes
18 Subclasses of Integer Anything No No
19 Anything Super class of Integer(Number/Object) Yes Yes
20 Any super class of Integer(Number/Object) Anything No No
21 PriorityQueue(non param read as anything) Integer Yes Yes
22 Integer Anything No No
23 Anything Anything Yes Yes
24 Anything Anything Yes Yes

**arjun says:**
10 September 2010 at 12:41 am
48,49,50
are compiler errors
please if you could some explanation kind of thing or reasons for the answers,it will be great for some weird questions here.

**Jean says:**
3 August 2011 at 7:42 pm
Q64: it compiles fine

**Shakthi balaji says:**
8 August 2011 at 3:09 pm
Hi Nick,

This is simply great. You must have spent a lot of time over this to bring the questions in more organised fashion. Great work.

**Balaji says:**
9 August 2011 at 5:49 am
Hi Niko,
One small question. The generics are used only during the compile time and not during the run time. Am i right? Ex – List will be just List during the runtime.

**vipul reddy says:**
16 December 2011 at 8:00 am
Hi Nikos Pougounias.

I am vipul Kumar. Thanks for providing the stuff on generics. I am preparing for scjp 1.6. The questions you have kept awared me of some topics i am not covered..

Thanks once again,
Vipul Kumar

**RaviTeja says:**
3 January 2012 at 1:51 pm
whats wrong with 30th question??

Can anyone xplain why it can't compile successfully

plz reply asap

**RaviTeja says:**
3 January 2012 at 2:01 pm
List list = new ArrayList();
for (Integer element : list) {
System.out.println(element);
}

this should not compile. right??

but in 36th question the answer is YES..

can anyone explain please..

**Hiral Jhaveri says:**
5 January 2012 at 2:21 pm
This is good stuff.

Nice concepts.

**Madhukar Gunda says:**
1 July 2012 at 10:27 pm
Nick
This is a super stuff

**srikanth ganta says:**
23 September 2012 at 11:36 am
Reblogged this on Srikanth's Blog.

**Gvesh says:**
1 February 2013 at 2:57 pm
Mohamed Farouk thanks a lot for clearing the concept ☺

**webcane says:**
6 April 2013 at 8:26 pm
31 the answer is b! (ClassCastException: java.lang.Object cannot be cast to java.lang.Comparable)

**Paulo Moreira Mendes says:**
9 December 2013 at 8:25 am
Thank you very much Nikos!
Mohamed Farouk, thank you for your help indeed.

**nour says:**
23 February 2014 at 1:58 pm
Thanks a lot Mohamed Farouk for sharing and clearing the concept 😉