<https://java2blog.com/java-interview-questions-for-5-years-experience/>

1.

**package** org.arpit.java2blog;

**class** A

{

**void** m1()

{

System.***out***.println("In m1 A");

}

}

**class** B **extends** A

{

**void** m1()

{

System.***out***.println("In m1 B");

}

**void** m2()

{

System.***out***.println("In m2 B");

}

}

**public** **class** Test {

**public** **static** **void** main(String[] args) {

A a=**new** B();

a.m2();

}

}

**What will be the output?**  
**A. In m2 B**  
**B. Compile time error**  
**C. Runtime error**

Answer:

B. Compile time error  
There will be compile-time error. Even though we are assigning B’s object to A’s reference we can call only methods which are in A from A’s reference.

2.

**package** org.arpit.java2blog;

**class** A

{

**void** m1() **throws** ArrayIndexOutOfBoundsException

{

System.***out***.println("In m1 A");

}

}

**class** B **extends** A

{

**void** m1() **throws** IndexOutOfBoundsException

{

System.***out***.println("In m1 B");

}

}

**public** **class** Test {

**public** **static** **void** main(String[] args) {

A a=**new** B();

a.m1();

}

}

**What will be the output?**  
**A. In m1 B**  
**B. Compile time error**  
**C. Runtime error**

Answer:

A. In m1 B  
This will work fine as ArrayIndexOutOfBoundsException and IndexOutOfBoundsException are Runtime exceptions and there is no rule for runtime exceptions while [**method overriding**](https://java2blog.com/method-overriding-in-java/).

3.

**package** org.arpit.java2blog;

**import** java.io.IOException;

**class** A

{

**void** m1() **throws** IOException

{

System.***out***.println("In m1 A");

}

}

**class** B **extends** A

{

**void** m1() **throws** Exception

{

System.***out***.println("In m1 B");

}

}

**public** **class** Test {

**public** **static** **void** main(String[] args) {

A a=**new** B();

**try** {

a.m1();

} **catch** (IOException e) {

e.printStackTrace();

}

}

}

**What will be the output?**  
**A. In m1 B**  
**B. Compile-time error**  
**C. Runtime error**

Answer:

B. Compile-time error  
As IOException and [**Exception**](https://java2blog.com/exception-handling-java/) are checked exception, so you can not broaden the scope of Exception while [**method overriding**](https://java2blog.com/method-overriding-in-java/).

4.

**class** A

{

**synchronized** **void** m1()

{

System.***out***.println("In m1 A");

}

**void** m2()

{

System.***out***.println("In m2 A");

}

}

There are two threads T1 and T2. T1 is accessing m1 method. Will T2 be able to access m2 method on the same instance at the same time?

Answer:

Yes, T2 will be able to access m2 as it does not require a lock to access m2 method.

5.

**class** A

{

**synchronized** **void** m1()

{

System.***out***.println("In m1 A");

}

**synchronized** **void** m2()

{

System.***out***.println("In m2 A");

}

}

There are two threads T1 and T2. T1 is accessing m1 method. Will T2 be able to access m2 method on the same instance at the same time?

Answer:

Yes, T2 will be able to access m2

6.

**class** A

{

**synchronized** **static** **void** m1()

{

System.***out***.println("In m1 A");

}

**synchronized** **void** m2()

{

System.***out***.println("In m2 A");

}

}

There are two threads T1 and T2. T1 is accessing m1 method. Will T2 be able to access m2 method on the same instance at the same time?

Answer:

Yes, T2 will be able to access m2 as it requires object level lock to access m2 method and T1 thread has taken class level lock.

7.

**package** org.arpit.java2blog;

**import** java.util.HashSet;

**public** **class** Customer {

String name;

**int** age;

Customer(String name,**int** age)

{

**this**.name=name;

**this**.age=age;

}

**public** **static** **void** main(String[] args) {

Customer c1= **new** Customer("John",20);

Customer c2= **new** Customer("John",20);

HashSet<Customer> customerSet=**new** HashSet<>();

customerSet.add(c1);

customerSet.add(c2);

System.***out***.println(customerSet.size());

}

// getters and setters

}

Answer:

Output will be 2 as we did not implement hashcode and equals method in Customer class.

8.

**package** org.arpit.java2blog;

**public** **class** Employee {

String name;

**int** age;

**public** Employee(String name,**int** age)

{

**this**.name=name;

**this**.age=age;

}

@Override

**public** **int** hashCode() {

**final** **int** prime = 31;

**int** result = 1;

result = prime \* result + age;

result = prime \* result + ((name == **null**) ? 0 : name.hashCode());

**return** result;

}

@Override

**public** **boolean** equals(Object obj) {

**if** (**this** == obj)

**return** **true**;

**if** (obj == **null**)

**return** **false**;

**if** (getClass() != obj.getClass())

**return** **false**;

Employee other = (Employee) obj;

**if** (age != other.age)

**return** **false**;

**if** (name == **null**) {

**if** (other.name != **null**)

**return** **false**;

} **else** **if** (!name.equals(other.name))

**return** **false**;

**return** **true**;

}

**public** String getName() {

**return** name;

}

**public** **void** setName(String name) {

**this**.name = name;

}

**public** **int** getAge() {

**return** age;

}

**public** **void** setAge(**int** age) {

**this**.age = age;

}

}

**package** org.arpit.java2blog;

**import** java.util.HashMap;

**import** java.util.Map;

**public** **class** HashMapMain {

**public** **static** **void** main(String[] args) {

Employee emp1=**new** Employee("Martin",27);

Map<Employee,String> hm=**new** HashMap<Employee,String>();

hm.put(emp1, "Verified");

emp1.setName("John");

System.out.println(hm.get(emp1));

}

}

Answer:

**Output will be null.**  
We have implemented Employee’s [**hashcode and equals method**](https://java2blog.com/hashcode-and-equals-method-in-java/) using name and age attributes, so when you put emp1 as key in hashmap, it will use hashcode and equals method and will be put in HashMap.  
After putting emp1 in HashMop, we have changed name of the employee, so when you will try to retrieve element from [**HashMap**](https://java2blog.com/hashmap-in-java-with-examples/) using hm.get(emp1), you won’t be able to get object which we have put earlier and it will return null.

9.

**package** org.arpit.java2blog;

**import** java.util.HashMap;

**import** java.util.Iterator;

**public** **class** IterateMapMain {

**public** **static** **void** main(String args[])

{

// HashMap with Country as key and capital as value

HashMap<String,String> map=**new** HashMap<String,String>();

map.put("India","Delhi");

map.put("Japan","Tokyo");

map.put("France","Paris");

map.put("Russia","Moscow");

// Iterating java iterator

System.***out***.println("Iterating java Iterator");

Iterator<String> countryKeySetIterator=map.keySet().iterator();

**while**(countryKeySetIterator.hasNext()){

String countryKey=countryKeySetIterator.next();

map.put("Nepal", "KathMandu");

System.***out***.println(countryKey);

}

System.***out***.println("-----------------------------");

}

}

Answer:

You will get below output  
Iterating java IteratorException in thread “main” java.util.ConcurrentModificationException  
at java.base/java.util.HashMap$HashIterator.nextNode(HashMap.java:1489)  
at java.base/java.util.HashMap$KeyIterator.next(HashMap.java:1512)  
at org.arpit.java2blog.IterateMapMain.main(IterateMapMain.java:24)Japan  
Whenever you try to change count of [**Hashmap**](https://java2blog.com/hashmap-in-java-with-examples/) while iterating, it will throw java.util.ConcurrentModificationException because we call iterator.next,it checks for modCount and initial count, if it does not match it will throw ConcurrentModificationException.

10.