* [Java Tutorial](https://www.journaldev.com/java-tutorial-java-ee-tutorials)
* [Index Posts](https://www.journaldev.com/2167/java-exception-interview-questions-and-answers)
* [Interview Questions](https://www.journaldev.com/2167/java-exception-interview-questions-and-answers)
* [Resources](https://www.journaldev.com/resources)

*nights\_staysearch*

* [*keyboard\_arrow\_left*Previous](https://www.journaldev.com/2110/jsp-interview-questions-and-answers)

[Next*keyboard\_arrow\_right*](https://www.journaldev.com/2244/eclipse-log4j-xml-log4j-dtd-cannot-validated-xml-definition)

[Share onFacebook](https://www.facebook.com/sharer/sharer.php?u=https%3A%2F%2Fwww.journaldev.com%2F2167%2Fjava-exception-interview-questions-and-answers)[Share onTwitter](https://twitter.com/intent/tweet?text=Java%20Exception%20Interview%20Questions%20and%20Answers&url=https%3A%2F%2Fwww.journaldev.com%2F2167%2Fjava-exception-interview-questions-and-answers&via=JournalDev&related=JournalDev)[Share onWhatsApp](https://api.whatsapp.com/send?text=Java%20Exception%20Interview%20Questions%20and%20Answers%20%E2%80%94%20https%3A%2F%2Fwww.journaldev.com%2F2167%2Fjava-exception-interview-questions-and-answers)

*nights\_staysearch*

**Java Exception Interview Questions and Answers**

[51 Comments](https://www.journaldev.com/2167/java-exception-interview-questions-and-answers#comments)

Filed Under: [Interview Questions](https://www.journaldev.com/interview-questions)

* [Home](https://www.journaldev.com/) » [Interview Questions](https://www.journaldev.com/interview-questions) » Java Exception Interview Questions and Answers

**Core Java Tutorial**

* [0. Java 15 Features](https://www.journaldev.com/44681/java-15-features)
* [1. Java 14 Features](https://www.journaldev.com/37273/java-14-features)
* [2. Java 13 Features](https://www.journaldev.com/33204/java-13-features)
* [3. Java 12 Features](https://www.journaldev.com/28666/java-12-features)
* [4. Java 11 Features](https://www.journaldev.com/24601/java-11-features)
* [5. Java 10 Features](https://www.journaldev.com/20395/java-10-features)
* [6. Java 9 Features](https://www.journaldev.com/13121/java-9-features-with-examples)
* [7. Java 8 Features](https://www.journaldev.com/2389/java-8-features-with-examples)
* [8. Java 9 Modules](https://www.journaldev.com/13106/java-9-modules)
* [9. Lambda Expressions](https://www.journaldev.com/2763/java-8-functional-interfaces)
* [10. Stream API](https://www.journaldev.com/2774/java-8-stream)
* [11. Date Time API](https://www.journaldev.com/2800/java-8-date-localdate-localdatetime-instant)
* [12. Java Access Modifiers](https://www.journaldev.com/2345/java-access-modifiers)
* [13. Abstract Class in Java](https://www.journaldev.com/1582/abstract-class-in-java)
* [14. Interface in Java](https://www.journaldev.com/1601/interface-in-java)
* [15. Composition in Java](https://www.journaldev.com/1325/composition-in-java-example)
* [16. Inheritance in Java](https://www.journaldev.com/644/inheritance-java-example)
* [17. Java Nested Classes](https://www.journaldev.com/996/java-inner-class)
* [18. Java String](https://www.journaldev.com/16928/java-string)
* [19. Java Annotations](https://www.journaldev.com/721/java-annotations)
* [20. Java Enum](https://www.journaldev.com/716/java-enum)
* [21. Collections in Java](https://www.journaldev.com/1260/collections-in-java-tutorial)
* [22. Java Generics](https://www.journaldev.com/1663/java-generics-example-method-class-interface)
* [23. Java Scanner Class](https://www.journaldev.com/872/java-scanner-class-java-util-scanner)
* [24. Exception Handling](https://www.journaldev.com/1696/exception-handling-in-java)
* [25. Threads in Java](https://www.journaldev.com/1016/java-thread-example)
* [26. Synchronization](https://www.journaldev.com/1061/thread-safety-in-java)
* [27. Java ThreadLocal](https://www.journaldev.com/1076/java-threadlocal-example)
* [28. Java Timer](https://www.journaldev.com/1050/java-timer-timertask-example)
* [29. Java Thread Pool](https://www.journaldev.com/1069/threadpoolexecutor-java-thread-pool-example-executorservice)
* [30. Java Callable Future](https://www.journaldev.com/1090/java-callable-future-example)
* [31. Java FutureTask](https://www.journaldev.com/1650/java-futuretask-example-program)
* [32. Java Lock](https://www.journaldev.com/2377/java-lock-example-reentrantlock)
* [33. Regular Expressions](https://www.journaldev.com/634/regular-expression-in-java-regex-example)
* [34. Java Heap Memory](https://www.journaldev.com/4098/java-heap-space-vs-stack-memory)
* [35. JVM Memory Model](https://www.journaldev.com/2856/java-jvm-memory-model-memory-management-in-java)
* [36. Java Reflection API](https://www.journaldev.com/1789/java-reflection-example-tutorial)
* [37. Serialization in Java](https://www.journaldev.com/2452/serialization-in-java)
* [38. Java System Class](https://www.journaldev.com/1847/system-class-java-properties-out-println)
* [39. Internationalization](https://www.journaldev.com/1370/java-i18n-internationalization-in-java)
* [40. Thread Dump](https://www.journaldev.com/1053/java-thread-dump-visualvm-jstack-kill-3-jcmd)
* [41. Deadlocks in Java](https://www.journaldev.com/1058/deadlock-in-java-example)
* [42. JDK, JRE and JVM](https://www.journaldev.com/546/difference-jdk-vs-jre-vs-jvm)
* [43. Java Classloader](https://www.journaldev.com/349/java-classloader)

**Important Interview Questions**

* [1. Core Java Interview Questions](https://www.journaldev.com/2366/core-java-interview-questions-and-answers)
* [2. String Interview Questions](https://www.journaldev.com/1321/java-string-interview-questions-and-answers)
* [3. Multithreading Interview Questions](https://www.journaldev.com/1162/java-multithreading-concurrency-interview-questions-answers)
* [4. Collections Interview Questions](https://www.journaldev.com/1330/java-collections-interview-questions-and-answers)
* [5. Exception Interview Questions](https://www.journaldev.com/2167/java-exception-interview-questions-and-answers)
* [6. Java Programming Interview Questions](https://www.journaldev.com/370/java-programming-interview-questions)
* [7. Java 8 Interview Questions Part 1](https://www.journaldev.com/8697/javase8-interview-questions-part1)
* [8. Java 8 Interview Questions Part 2](https://www.journaldev.com/10081/javase8-interview-questions-part2)
* [9. Servlet Interview Questions](https://www.journaldev.com/2015/servlet-interview-questions-and-answers)
* [10. JSP Interview Questions](https://www.journaldev.com/2110/jsp-interview-questions-and-answers)
* [11. Struts 2 Interview Questions](https://www.journaldev.com/2354/struts2-interview-questions-and-answers)
* [12. JDBC Interview Questions](https://www.journaldev.com/2529/jdbc-interview-questions-and-answers)
* [13. Spring Interview Questions](https://www.journaldev.com/2696/spring-interview-questions-and-answers)
* [14. Hibernate Interview Questions](https://www.journaldev.com/3633/hibernate-interview-questions-and-answers)
* [15. JSF Interview Questions](https://www.journaldev.com/7261/jsf-interview-questions-and-answers)
* [16. Web Services Interview Questions](https://www.journaldev.com/9193/web-services-interview-questions-soap-restful)
* [17. Scala Basic Interview Questions](https://www.journaldev.com/8958/scala-basic-interview-questions)
* [18. Scala Intermediate Interview Questions](https://www.journaldev.com/9663/scala-intermediate-interview-questions)
* [19. Scala Advanced Interview Questions](https://www.journaldev.com/8960/scala-advanced-interview-questions)
* [20. Scala Interview Questions Summary](https://www.journaldev.com/10302/scala-intermediate-and-advanced-interview-questions-and-answers)
* [21. Common Job Interview Questions](https://www.journaldev.com/3923/common-job-interview-questions-and-answer-tips)
* [22. Java ClassLoader](https://www.journaldev.com/349/java-classloader)
* [23. String StringBuffer StringBuilder](https://www.journaldev.com/538/java-interview-question-string-vs-stringbuffer-vs-stringbuilder)
* [24. Java is Pass By Value](https://www.journaldev.com/3884/java-is-pass-by-value-and-not-pass-by-reference)
* [25. Java Heap vs Stack Memory](https://www.journaldev.com/4098/java-heap-space-vs-stack-memory)
* [26. Java Interview Questions](https://www.journaldev.com/java-interview-questions)

Java provides a robust and object-oriented approach to handle exception scenarios known as **Java Exception Handling**.

Sometime back I wrote a long post on [Exception Handling in Java](https://www.journaldev.com/1696/exception-handling-in-java) and today I am listing some important **Java Exceptions Questions with Answers** to help you in interviews.

1. [What is an Exception in Java?](https://www.journaldev.com/2167/java-exception-interview-questions-and-answers#java-exception)
2. [What are the Exception Handling Keywords in Java?](https://www.journaldev.com/2167/java-exception-interview-questions-and-answers#java-exception-keywords)
3. [Explain Java Exception Hierarchy?](https://www.journaldev.com/2167/java-exception-interview-questions-and-answers#java-exception-hierarchy)
4. [What are the important methods of Java Exception Class?](https://www.journaldev.com/2167/java-exception-interview-questions-and-answers#java-exception-methods)
5. [Explain Java 7 ARM Feature and multi-catch block?](https://www.journaldev.com/2167/java-exception-interview-questions-and-answers#java-7-arm-multi-catch)
6. [What is the difference between Checked and Unchecked Exceptions in Java?](https://www.journaldev.com/2167/java-exception-interview-questions-and-answers#checked-vs-unchecked-excepion)
7. [What is the difference between the throw and throws keyword in Java?](https://www.journaldev.com/2167/java-exception-interview-questions-and-answers#throw-vs-throws)
8. [How to write custom exceptions in Java?](https://www.journaldev.com/2167/java-exception-interview-questions-and-answers#java-custom-exception)
9. [What is OutOfMemoryError in Java?](https://www.journaldev.com/2167/java-exception-interview-questions-and-answers#java-outofmemoryerror)
10. [What are different scenarios causing “Exception in thread main”?](https://www.journaldev.com/2167/java-exception-interview-questions-and-answers#java-exception-in-thread-main)
11. [What is the difference between final, finally, and finalize in Java?](https://www.journaldev.com/2167/java-exception-interview-questions-and-answers#java-final-finally-finalize)
12. [What happens when an exception is thrown by the main method?](https://www.journaldev.com/2167/java-exception-interview-questions-and-answers#exception-main-method)
13. [Can we have an empty catch block?](https://www.journaldev.com/2167/java-exception-interview-questions-and-answers#empty-catch-block)
14. [Provide some Java Exception Handling Best Practices?](https://www.journaldev.com/2167/java-exception-interview-questions-and-answers#java-exception-handling-best-practices)
15. [What is the problem with the below programs and how do we fix it?](https://www.journaldev.com/2167/java-exception-interview-questions-and-answers#java-exception-programming-questions)

**1. What is an Exception in Java?**

An exception is an error event that can happen during the execution of a program and disrupts its normal flow. The exception can arise from different kinds of situations such as wrong data entered by the user, hardware failure, network connection failure, etc.

Whenever any error occurs while executing a java statement, an exception object is created, and then [**JRE**](https://www.journaldev.com/546/difference-jdk-vs-jre-vs-jvm) tries to find an exception handler to handle the exception. If a suitable exception handler is found then the exception object is passed to the handler code to process the exception, known as **catching the exception**. If no handler is found then the application throws the exception to the runtime environment and JRE terminates the program.

**Java Exception handling** framework is used to handle runtime errors only, compile-time errors are not handled by exception handling framework.

**2. What are the Exception Handling Keywords in Java?**

There are four keywords used in java exception handling.

1. **throw**: Sometimes we explicitly want to create an exception object and then throw it to halt the normal processing of the program. The **throw** keyword is used to throw exceptions to the runtime to handle it.
2. **throws**: When we are throwing any checked exception in a method and not handling it, then we need to use the throws keyword in the method signature to let the caller program know the exceptions that might be thrown by the method. The caller method might handle these exceptions or propagate them to its caller method using the throws keyword. We can provide multiple exceptions in the throws clause and it can be used with the **main()** method also.
3. **try-catch**: We use try-catch block for exception handling in our code. try is the start of the block and catch is at the end of the try block to handle the exceptions. We can have multiple catch blocks with a try and try-catch blocks can be nested also. catch block requires a parameter that should be of type Exception.
4. **finally**: The finally block is optional and can be used only with a try-catch block. Since exception halts the process of execution, we might have some resources open that will not get closed, so we can use the finally block. The finally block gets executed always, whether an exception occurs or not.

**3. Explain Java Exception Hierarchy?**

Java Exceptions are hierarchical and [inheritance](https://www.journaldev.com/644/inheritance-java-example) is used to categorize different types of exceptions. Throwable is the parent class of Java Exceptions Hierarchy and it has two child objects – Error and Exception. Exceptions are further divided into checked exceptions and runtime exceptions.

**Errors** are exceptional scenarios that are out of the scope of application and it’s not possible to anticipate and recover from them, for example, hardware failure, JVM crash, or out-of-memory error.

**Checked Exceptions** are exceptional scenarios that we can anticipate in a program and try to recover from it, for example, FileNotFoundException. We should catch this exception and provide a useful message to the user and log it properly for debugging purposes. Exception is the parent class of all Checked Exceptions.

**Runtime Exceptions** are caused by bad programming, for example, trying to retrieve an element from the Array. We should check the length of the array first before trying to retrieve the element otherwise it might throw ArrayIndexOutOfBoundException at runtime. RuntimeException is the parent class of all runtime exceptions.

**4. What are the important methods of Java Exception Class?**

Exception and all of its subclasses don’t provide any specific methods and all of the methods are defined in the base class Throwable.

1. **String getMessage()** – This method returns the message String of Throwable and the message can be provided while creating the exception through its constructor.
2. **String getLocalizedMessage()** – This method is provided so that subclasses can override it to provide the locale-specific messages to the calling program. Throwable class implementation of this method simply use getMessage() method to return the exception message.
3. **synchronized Throwable getCause()** – This method returns the cause of the exception or null if the cause is unknown.
4. **String toString()** – This method returns the information about Throwable in String format, the returned String contains the name of Throwable class and localized message.
5. **void printStackTrace()** – This method prints the stack trace information to the standard error stream, this method is overloaded and we can pass PrintStream or PrintWriter as an argument to write the stack trace information to the file or stream.

**5. Explain Java 7 ARM Feature and multi-catch block?**

If you are catching a lot of exceptions in a single try block, you will notice that catch block code looks very ugly and mostly consists of redundant code to log the error, keeping this in mind Java 7 one of the features was the multi-catch block where we can catch multiple exceptions in a single catch block. The catch block with this feature looks like below:

catch(IOException | SQLException | Exception ex){

logger.error(ex);

throw new MyException(ex.getMessage());

}

Most of the time, we use finally block just to close the resources and sometimes we forget to close them and get runtime exceptions when the resources are exhausted. These exceptions are hard to debug and we might need to look into each place where we are using that type of resource to make sure we are closing it. So java 7 one of the improvements was **try-with-resources** where we can create a resource in the try statement itself and use it inside the try-catch block. When the execution comes out of the try-catch block, the runtime environment automatically closes these resources. Sample of try-catch block with this improvement is:

try (MyResource mr = new MyResource()) {

System.out.println("MyResource created in try-with-resources");

} catch (Exception e) {

e.printStackTrace();

}

Read more about this at [**Java 7 ARM**](https://www.journaldev.com/592/java-try-with-resources).

**6. What is the difference between Checked and Unchecked Exceptions in Java?**

1. Checked Exceptions should be handled in the code using try-catch block or else the method should use the throws keyword to let the caller know about the checked exceptions that might be thrown from the method. Unchecked Exceptions are not required to be handled in the program or to mention them in the throws clause of the method.
2. Exception is the superclass of all checked exceptions whereas RuntimeException is the superclass of all unchecked exceptions. Note that RuntimeException is the child class of Exception.
3. Checked exceptions are error scenarios that require to be handled in the code, or else you will get compile time error. For example, if you use FileReader to read a file, it throws FileNotFoundException and we must catch it in the try-catch block or throw it again to the caller method. Unchecked exceptions are mostly caused by poor programming, for example, NullPointerException when invoking a method on an object reference without making sure that it’s not null. For example, I can write a method to remove all the vowels from the string. It’s the caller’s responsibility to make sure not to pass a null string. I might change the method to handle these scenarios but ideally, the caller should take care of this.

**7. What is the difference between the throw and throws keyword in Java?**

throws keyword is used with method signature to declare the exceptions that the method might throw whereas throw keyword is used to disrupt the flow of the program and handing over the exception object to runtime to handle it.

**8. How to write custom exceptions in Java?**

We can extend Exception class or any of its subclasses to create our custom exception class. The custom exception class can have its own variables and methods that we can use to pass error codes or other exception-related information to the exception handler.

A simple example of a custom exception is shown below.

package com.journaldev.exceptions;

import java.io.IOException;

public class MyException extends IOException {

private static final long serialVersionUID = 4664456874499611218L;

private String errorCode="Unknown\_Exception";

public MyException(String message, String errorCode){

super(message);

this.errorCode=errorCode;

}

public String getErrorCode(){

return this.errorCode;

}

}

**9. What is OutOfMemoryError in Java?**

OutOfMemoryError in Java is a subclass of java.lang.VirtualMachineError and it’s thrown by JVM when it ran out of heap memory. We can fix this error by providing more memory to run the java application through java options.

$>java MyProgram -Xms1024m -Xmx1024m -XX:PermSize=64M -XX:MaxPermSize=256m

**10. What are different scenarios causing “Exception in thread main”?**

Some of the common main thread exception scenarios are:

* **Exception in thread main java.lang.UnsupportedClassVersionError**: This exception comes when your java class is compiled from another JDK version and you are trying to run it from another java version.
* **Exception in thread main java.lang.NoClassDefFoundError**: There are two variants of this exception. The first one is where you provide the class full name with .class extension. The second scenario is when Class is not found.
* **Exception in thread main java.lang.NoSuchMethodError: main**: This exception comes when you are trying to run a class that doesn’t have the main method.
* **Exception in thread “main” java.lang.ArithmeticException**: Whenever an exception is thrown from the main method, it prints the exception in the console. The first part explains that an exception is thrown from the main method, the second part prints the exception class name and then after a colon, it prints the exception message.

Read more about these at [Java Exception in thread main](https://www.journaldev.com/611/exception-in-thread-main-java).

**11. What is the difference between final, finally, and finalize in Java?**

final and finally are keywords in java whereas finalize is a method.

final keyword can be used with class variables so that they can’t be reassigned, with the class to avoid extending by classes and with methods to avoid overriding by subclasses, finally keyword is used with try-catch block to provide statements that will always get executed even if some exception arises, usually finally is used to close resources. finalize() method is executed by Garbage Collector before the object is destroyed, it’s a great way to make sure all the global resources are closed.

Out of the three, only finally is related to java exception handling.

**12. What happens when an exception is thrown by the main method?**

When an exception is thrown by a main() method, Java Runtime terminates the program and prints the exception message and stack trace in the system console.

**13. Can we have an empty catch block?**

We can have an empty catch block but it’s an example of bad programming. We should never have an empty catch block because if the exception is caught by that block, we will have no information about the exception and it wil be a nightmare to debug it. There should be at least a logging statement to log the exception details in console or log files.

**14. Provide some Java Exception Handling Best Practices?**

Some of the best practices related to Java Exception Handling are:

* Use Specific Exceptions for ease of debugging.
* Throw Exceptions Early (Fail-Fast) in the program.
* Catch Exceptions late in the program, let the caller handle the exception.
* Use Java 7 ARM feature to make sure resources are closed or use finally block to close them properly.
* Always log exception messages for debugging purposes.
* Use multi-catch block for cleaner close.
* Use custom exceptions to throw a single type of exception from your application API.
* Follow naming convention, always end with Exception.
* Document the Exceptions Thrown by a method using @throws in javadoc.
* Exceptions are costly, so throw it only when it makes sense. Else you can catch them and provide a null or empty response.

Read more about them in detail at [Java Exception Handling Best Practices](https://www.journaldev.com/1696/exception-handling-in-java#exception-best-practices).

**15. What is the problem with the below programs and how do we fix it?**

In this section, we will look into some programming questions related to java exceptions.

1. **What is the problem with the below program?**
2. package com.journaldev.exceptions;
3. import java.io.FileNotFoundException;
4. import java.io.IOException;
5. public class TestException {
6. public static void main(String[] args) {
7. try {
8. testExceptions();
9. } catch (FileNotFoundException | IOException e) {
10. e.printStackTrace();
11. }
12. }


16. public static void testExceptions() throws IOException, FileNotFoundException{
18. }
19. }

The above program won’t compile and you will get an error message as “The exception FileNotFoundException is already caught by the alternative IOException”. This is because FileNotFoundException is a subclass of IOException, there are two ways to solve this problem.

The first way is to use a single catch block for both the exceptions.

try {

testExceptions();

}catch(FileNotFoundException e){

e.printStackTrace();

}catch (IOException e) {

e.printStackTrace();

}

Another way is to remove the FileNotFoundException from the multi-catch block.

try {

testExceptions();

}catch (IOException e) {

e.printStackTrace();

}

You can chose any of these approach based on your catch block code.

1. **What is the problem with the below program?**
2. package com.journaldev.exceptions;
3. import java.io.FileNotFoundException;
4. import java.io.IOException;
5. import javax.xml.bind.JAXBException;
6. public class TestException1 {
7. public static void main(String[] args) {
8. try {
9. go();
10. } catch (IOException e) {
11. e.printStackTrace();
12. } catch (FileNotFoundException e) {
13. e.printStackTrace();
14. } catch (JAXBException e) {
15. e.printStackTrace();
16. }
17. }
18. public static void go() throws IOException, JAXBException, FileNotFoundException{
20. }
21. }

The program won’t compile because FileNotFoundException is a subclass of IOException, so the catch block of FileNotFoundException is unreachable and you will get an error message as “Unreachable catch block for FileNotFoundException. It is already handled by the catch block for IOException”.

You need to fix the catch block order to solve this issue.

try {

go();

} catch (FileNotFoundException e) {

e.printStackTrace();

} catch (IOException e) {

e.printStackTrace();

} catch (JAXBException e) {

e.printStackTrace();

}

Notice that JAXBException is not related to IOException or FileNotFoundException and can be put anywhere in the above catch block hierarchy.

1. **What is the problem with the below program?**
2. package com.journaldev.exceptions;
3. import java.io.IOException;
4. import javax.xml.bind.JAXBException;
5. public class TestException2 {
6. public static void main(String[] args) {
7. try {
8. foo();
9. } catch (IOException e) {
10. e.printStackTrace();
11. }catch(JAXBException e){
12. e.printStackTrace();
13. }catch(NullPointerException e){
14. e.printStackTrace();
15. }catch(Exception e){
16. e.printStackTrace();
17. }
18. }
19. public static void foo() throws IOException{
21. }
22. }

The program won’t compile because JAXBException is a checked exception and foo() method should throw this exception to catch in the calling method. You will get an error message as “Unreachable catch block for JAXBException. This exception is never thrown from the try statement body”.

To solve this issue, you will have to remove the catch block of JAXBException.

Notice that catching NullPointerException is valid because it’s an unchecked exception.

1. **What is the problem with the below program?**
2. package com.journaldev.exceptions;
3. public class TestException3 {
4. public static void main(String[] args) {
5. try{
6. bar();
7. }catch(NullPointerException e){
8. e.printStackTrace();
9. }catch(Exception e){
10. e.printStackTrace();
11. }
13. foo();
14. }
15. public static void bar(){
17. }
19. public static void foo() throws NullPointerException{
21. }
22. }

This is a trick question, there is no problem with the code and it will compile successfully. We can always catch an Exception or any unchecked exception even if it’s not in the throws clause of the method.

Similarly, if a method (foo) declares an unchecked exception in the throws clause, it is not mandatory to handle that in the program.

1. **What is the problem with the below program?**
2. package com.journaldev.exceptions;
3. import java.io.IOException;
4. public class TestException4 {
5. public void start() throws IOException{
6. }
8. public void foo() throws NullPointerException{
10. }
11. }
12. class TestException5 extends TestException4{
14. public void start() throws Exception{
15. }
17. public void foo() throws RuntimeException{
19. }
20. }

The above program won’t compile because the start() method signature is not the same in the subclass. To fix this issue, we can either change the method singnature in the subclass to be exactly the same as the superclass or we can remove the throws clause from the subclass method as shown below.

@Override

public void start(){

}

1. **What is the problem with the below program?**
2. package com.journaldev.exceptions;
3. import java.io.IOException;
4. import javax.xml.bind.JAXBException;
5. public class TestException6 {
6. public static void main(String[] args) {
7. try {
8. foo();
9. } catch (IOException | JAXBException e) {
10. e = new Exception("");
11. e.printStackTrace();
12. }catch(Exception e){
13. e = new Exception("");
14. e.printStackTrace();
15. }
16. }
17. public static void foo() throws IOException, JAXBException{
19. }
20. }

The above program won’t compile because the exception object in the multi-catch block is final and we can’t change its value. You will get compile time error as “The parameter e of a multi-catch block cannot be assigned”.

We have to remove the assignment of “e” to a new exception object to solve this error.

Read more at [Java 7 multi-catch block](https://www.journaldev.com/629/java-catch-multiple-exceptions-rethrow-exception).

Thats all for the java exception interview questions, I hope you will like them. I will be adding more to the list in the future, make sure you bookmark it for future use.

[Share onFacebook](https://www.facebook.com/sharer/sharer.php?u=https%3A%2F%2Fwww.journaldev.com%2F2167%2Fjava-exception-interview-questions-and-answers)[Share onTwitter](https://twitter.com/intent/tweet?text=Java%20Exception%20Interview%20Questions%20and%20Answers&url=https%3A%2F%2Fwww.journaldev.com%2F2167%2Fjava-exception-interview-questions-and-answers&via=JournalDev&related=JournalDev)[Share onWhatsApp](https://api.whatsapp.com/send?text=Java%20Exception%20Interview%20Questions%20and%20Answers%20%E2%80%94%20https%3A%2F%2Fwww.journaldev.com%2F2167%2Fjava-exception-interview-questions-and-answers)[Share onReddit](https://www.reddit.com/submit?url=https%3A%2F%2Fwww.journaldev.com%2F2167%2Fjava-exception-interview-questions-and-answers)[Share onLinkedIn](https://www.linkedin.com/shareArticle?mini=1&url=https%3A%2F%2Fwww.journaldev.com%2F2167%2Fjava-exception-interview-questions-and-answers&title=Java%20Exception%20Interview%20Questions%20and%20Answers&source=https%3A%2F%2Fwww.journaldev.com)[Share onEmail](mailto:?body=I%20read%20this%20post%20and%20wanted%20to%20share%20it%20with%20you.%20Here%27s%20the%20link%3A%20https%3A%2F%2Fwww.journaldev.com%2F2167%2Fjava-exception-interview-questions-and-answers&subject=A%20post%20worth%20sharing%3A%20Java%20Exception%20Interview%20Questions%20and%20Answers)

[PREV](https://www.journaldev.com/2110/jsp-interview-questions-and-answers)

[JSP Interview Questions and Answers](https://www.journaldev.com/2110/jsp-interview-questions-and-answers)

[NEXT](https://www.journaldev.com/2244/eclipse-log4j-xml-log4j-dtd-cannot-validated-xml-definition)

[Eclipse log4j.xml - log4j.dtd cannot be validated as the XML definition](https://www.journaldev.com/2244/eclipse-log4j-xml-log4j-dtd-cannot-validated-xml-definition)

[Pankaj](https://www.journaldev.com/author/pankaj)

I love Open Source technologies and writing about my experience about them is my passion.

Follow Author

**Comments**

1. rajasays:

[February 4, 2021 at 12:56 pm](https://www.journaldev.com/2167/java-exception-interview-questions-and-answers#comment-51467)

Thank You for the post pankaj,  
could you please update the indentation/numbering.  
Thank You.

[Reply](https://www.journaldev.com/2167/java-exception-interview-questions-and-answers#comment-51467)

1. Jai Kumarsays:

[August 9, 2020 at 3:05 am](https://www.journaldev.com/2167/java-exception-interview-questions-and-answers#comment-50530)

May i know all these questions are enough for Interview

[Reply](https://www.journaldev.com/2167/java-exception-interview-questions-and-answers#comment-50530)

1. Ramsays:

[July 31, 2019 at 12:20 am](https://www.journaldev.com/2167/java-exception-interview-questions-and-answers#comment-47257)

Hi Sir,

Thanks for advance,

This is your statement, Java Exception handling framework is used to handle runtime errors only, compile time errors are not handled by exception handling framework.

Could you please share me who will handle this compile time errors(compiletime exceptions).

[Reply](https://www.journaldev.com/2167/java-exception-interview-questions-and-answers#comment-47257)

* 1. [Pankaj](https://www.journaldev.com/)says:

[July 31, 2019 at 5:24 am](https://www.journaldev.com/2167/java-exception-interview-questions-and-answers#comment-47261)

Compile-time errors are programmer responsibility. For example, you typed “STRING” rather than “String” and getting a compile-time error. Do you think this is something Java can fix? NO.

[Reply](https://www.journaldev.com/2167/java-exception-interview-questions-and-answers#comment-47261)

* 1. vijaysays:

[April 2, 2021 at 9:56 am](https://www.journaldev.com/2167/java-exception-interview-questions-and-answers#comment-51746)

system admin and admin server is responsible to handle the exception.rise heap  
memory.use this concept we can handile the exception  
any qes send me in email=vijaybhan@gmail.com

[Reply](https://www.journaldev.com/2167/java-exception-interview-questions-and-answers#comment-51746)

1. Chetansays:

[July 14, 2019 at 1:19 am](https://www.journaldev.com/2167/java-exception-interview-questions-and-answers#comment-47128)

consider following code snipped  
try{  
int a = 10/0; // statement 1  
}catch(ArithmeticException ae){  
// handle code  
}  
My question is – exception arise at statement 1 & Exception object created,but how java internally identify that exception is ArithmeticException only, not other one ?

[Reply](https://www.journaldev.com/2167/java-exception-interview-questions-and-answers#comment-47128)

1. Murali Psays:

[February 25, 2019 at 12:52 pm](https://www.journaldev.com/2167/java-exception-interview-questions-and-answers#comment-46082)

Good set of questions Pankaj. Thanks for the effort you put in.

[Reply](https://www.journaldev.com/2167/java-exception-interview-questions-and-answers#comment-46082)

1. Dashsays:

[September 16, 2018 at 10:24 pm](https://www.journaldev.com/2167/java-exception-interview-questions-and-answers#comment-44487)

HI Pankaj,

Thanks for providing this very useful tutorial, i learned a lot from it.

It is mentioned that catching “NullpointerException” is perfectly fine.  
But i think it’s a bad practice to catch such Runtime Exceptions, they should be avoided with proper programming logics rather catching.

what do you think?

Thanks,  
-Dash

[Reply](https://www.journaldev.com/2167/java-exception-interview-questions-and-answers#comment-44487)

* 1. [Pankaj](https://www.journaldev.com/)says:

[September 16, 2018 at 11:13 pm](https://www.journaldev.com/2167/java-exception-interview-questions-and-answers#comment-44490)

NullPointerException comes from poor programming. It’s better to have a null check rather than catching NullPointerException. I have never mentioned that it’s a good practice to catch NPE. However, you might see that in some of the sample codes, that is just to explain some concepts and shouldn’t be considered as best practice.

[Reply](https://www.journaldev.com/2167/java-exception-interview-questions-and-answers#comment-44490)

1. Murali Tekumatlasays:

[September 10, 2018 at 6:15 am](https://www.journaldev.com/2167/java-exception-interview-questions-and-answers#comment-44442)

Hi All,  
Question 5 ; One overriding rule w.r.t exception: If child class method throws any checked exception compulsory parent class method should throw the same checked exception or its parent. It needs not to exact same. But there are no restrictions for unchecked exceptions. The code will be fine like below…

package com.journaldev.exceptions;

import java.io.IOException;

public class TestException4 {

public void start() throws Exception{  
}

public void foo() throws NullPointerException{

}  
}

class TestException5 extends TestException4{

public void start() throws IOException{  
}

public void foo() throws RuntimeException{

}  
}

Thanks,  
Murali

[Reply](https://www.journaldev.com/2167/java-exception-interview-questions-and-answers#comment-44442)

1. Shreyasays:

[August 11, 2018 at 10:09 pm](https://www.journaldev.com/2167/java-exception-interview-questions-and-answers#comment-44181)

Please explain 5 question

[Reply](https://www.journaldev.com/2167/java-exception-interview-questions-and-answers#comment-44181)

1. Ozzybilsays:

[August 1, 2018 at 12:50 am](https://www.journaldev.com/2167/java-exception-interview-questions-and-answers#comment-44076)

Hi,  
First sample code in section 5 looked incorrect to me (catch(IOException | SQLException | Exception ex)). Types in multi-catch block must be disjoint. IO and SQL exceptions are subclasses of Exception in this case!

[Reply](https://www.journaldev.com/2167/java-exception-interview-questions-and-answers#comment-44076)

1. Jitendra Singhsays:

[May 15, 2018 at 10:54 am](https://www.journaldev.com/2167/java-exception-interview-questions-and-answers#comment-43394)

Beautiful tutorials.We love to study java from this tutorials all time!! Keep it Up!!

[Reply](https://www.journaldev.com/2167/java-exception-interview-questions-and-answers#comment-43394)

1. Shubham Negisays:

[April 12, 2018 at 2:12 pm](https://www.journaldev.com/2167/java-exception-interview-questions-and-answers#comment-42942)

Hi,

Do you really think than question 6 has the right explanation. [ Rest of the questions I haven’t read ]

This is totally wrong, How you are going to handle “checked exceptions in the try catch block”. Unchecked exceptions not meant to be handled using try-catch.?

Or it’s some kind of typo mistake.

[Reply](https://www.journaldev.com/2167/java-exception-interview-questions-and-answers#comment-42942)

* 1. [Pankaj](https://www.journaldev.com/)says:

[April 13, 2018 at 2:10 am](https://www.journaldev.com/2167/java-exception-interview-questions-and-answers#comment-42946)

I didn’t get your point, let’s say you have a foo() method that is throwing FileNotFoundException and NullPointerException. Now I am calling foo() method in my program, since FileNotFoundException is a checked exception, I will have to either use try-catch block to handle it or use throws clause to throw it back to caller.

For NullPointerException, I don’t have to do anything at all. I am not required to use catch block or add it to throws clause. However better programming would be to make sure that if I am passing any input to foo() method then make sure it’s not null.

[Reply](https://www.journaldev.com/2167/java-exception-interview-questions-and-answers#comment-42946)

1. Anilsays:

[January 9, 2018 at 8:42 pm](https://www.journaldev.com/2167/java-exception-interview-questions-and-answers#comment-41983)

Good explanation, must watch  
<https://www.youtube.com/watch?v=nUWd6242Myw>

[Reply](https://www.journaldev.com/2167/java-exception-interview-questions-and-answers#comment-41983)

1. bharathi gowdsays:

[February 21, 2017 at 2:16 am](https://www.journaldev.com/2167/java-exception-interview-questions-and-answers#comment-37537)

this is the best way of student activity.super good but students not utilise this proper way

[Reply](https://www.journaldev.com/2167/java-exception-interview-questions-and-answers#comment-37537)

1. Kametsays:

[March 10, 2016 at 2:14 am](https://www.journaldev.com/2167/java-exception-interview-questions-and-answers#comment-34273)

Its written that “Exception is the super class of all checked exceptions whereas RuntimeException is the super class of all unchecked exceptions.” According to hierarchy :-

Child of exception class are : IOException ,RuntimeException,ClassNotFoundException, CloneNotSupportedException.

So according to this “Exception is the super class of all checked exceptions” ,all the above are checked ??

[Reply](https://www.journaldev.com/2167/java-exception-interview-questions-and-answers#comment-34273)

* 1. [Pankaj](https://www.journaldev.com/)says:

[June 28, 2016 at 6:18 am](https://www.journaldev.com/2167/java-exception-interview-questions-and-answers#comment-35462)

Think of this like a Tree. Exception is the Root branch and then we have RuntimeException branch. Now anything coming out of RuntimeException branch will become unchecked exception. All others will be checked exceptions.

Exception is the superset. Anything extending RuntimeException is part of a child set and unchecked exceptions, if not then it’s checked exception.

[Reply](https://www.journaldev.com/2167/java-exception-interview-questions-and-answers#comment-35462)

1. Verangasays:

[December 10, 2015 at 9:11 am](https://www.journaldev.com/2167/java-exception-interview-questions-and-answers#comment-33653)

Very important tutorial

[Reply](https://www.journaldev.com/2167/java-exception-interview-questions-and-answers#comment-33653)

* 1. [Pankaj](https://www.journaldev.com/)says:

[June 28, 2016 at 6:19 am](https://www.journaldev.com/2167/java-exception-interview-questions-and-answers#comment-35463)

Thanks a lot Veranga.

[Reply](https://www.journaldev.com/2167/java-exception-interview-questions-and-answers#comment-35463)

1. Urveesays:

[October 7, 2015 at 5:52 am](https://www.journaldev.com/2167/java-exception-interview-questions-and-answers#comment-33322)

Very informative and ease to understand !

However I have a point to add here – It is not mandatory to have catch after every try block, finally can be followed after a try block.  
ie. Each try block must be followed by catch OR / AND finally.

(This was one of the tricky questions in some of the interviews I conducted and most of the candidates failed to answer correctly !)

[Reply](https://www.journaldev.com/2167/java-exception-interview-questions-and-answers#comment-33322)

* 1. Ravi Kumarsays:

[April 29, 2016 at 4:27 pm](https://www.journaldev.com/2167/java-exception-interview-questions-and-answers#comment-34609)

Each try block must be followed by catch OR finally.

[Reply](https://www.journaldev.com/2167/java-exception-interview-questions-and-answers#comment-34609)

* + 1. Andreea Ciocansays:

[November 22, 2017 at 2:30 am](https://www.journaldev.com/2167/java-exception-interview-questions-and-answers#comment-40025)

When we have try with resources we are allowed skip catch and finally blocks.

[Reply](https://www.journaldev.com/2167/java-exception-interview-questions-and-answers#comment-40025)

* 1. [Pankaj](https://www.journaldev.com/)says:

[June 28, 2016 at 6:20 am](https://www.journaldev.com/2167/java-exception-interview-questions-and-answers#comment-35464)

Yes you are right, you can have try-finally block too.

[Reply](https://www.journaldev.com/2167/java-exception-interview-questions-and-answers#comment-35464)

1. Ruchi Guptasays:

[December 16, 2014 at 11:26 pm](https://www.journaldev.com/2167/java-exception-interview-questions-and-answers#comment-31148)

Hi Pankaj,

Could you please explain Progarm F, i couldnt understand it.

[Reply](https://www.journaldev.com/2167/java-exception-interview-questions-and-answers#comment-31148)

* 1. [Pankaj](https://www.journaldev.com/)says:

[June 28, 2016 at 6:22 am](https://www.journaldev.com/2167/java-exception-interview-questions-and-answers#comment-35465)

When you use multi-catch block like catch (IOException | JAXBException e), the exception object implicitly becomes final. So you can’t change it.

That’s why you will get compilation error in e = new Exception(""); statement in the catch block.

It’s a very simple program, you can copy it in Eclipse and check for yourself.

[Reply](https://www.journaldev.com/2167/java-exception-interview-questions-and-answers#comment-35465)

* + 1. Indreshsays:

[January 30, 2018 at 1:25 am](https://www.journaldev.com/2167/java-exception-interview-questions-and-answers#comment-42131)

same is applicable to try block resources, they are also final i.e. you cant re assign new value (object) to resources.

[Reply](https://www.journaldev.com/2167/java-exception-interview-questions-and-answers#comment-42131)

1. Rajeshwarisays:

[November 20, 2014 at 9:49 am](https://www.journaldev.com/2167/java-exception-interview-questions-and-answers#comment-30795)

Very informative.plz add few more questions

[Reply](https://www.journaldev.com/2167/java-exception-interview-questions-and-answers#comment-30795)

* 1. [Pankaj](https://www.journaldev.com/)says:

[June 28, 2016 at 6:23 am](https://www.journaldev.com/2167/java-exception-interview-questions-and-answers#comment-35466)

I will when I get more, but if you got some new and tricky questions then feel free to comment here.

[Reply](https://www.journaldev.com/2167/java-exception-interview-questions-and-answers#comment-35466)

1. prabhatsays:

[November 10, 2014 at 11:24 am](https://www.journaldev.com/2167/java-exception-interview-questions-and-answers#comment-30702)

can u re-explain throw and throws in detail ??  
and also msg me the ans on my id please..

[Reply](https://www.journaldev.com/2167/java-exception-interview-questions-and-answers#comment-30702)

* 1. [Pankaj](https://www.journaldev.com/)says:

[June 28, 2016 at 6:25 am](https://www.journaldev.com/2167/java-exception-interview-questions-and-answers#comment-35467)

It’s simple. ‘throw’ keyword is use to send exception to the caller program. ‘throws’ keyword is used in method declaration to let caller program know that the method may throw these xxx exceptions, so make sure you handle them.

[Reply](https://www.journaldev.com/2167/java-exception-interview-questions-and-answers#comment-35467)

1. Prathapsays:

[August 2, 2014 at 1:05 am](https://www.journaldev.com/2167/java-exception-interview-questions-and-answers#comment-29676)

my question is about QN:E, As per answer start() signature is different in subclass (which is creating the problem) and what about foo() which is also having same issue?

[Reply](https://www.journaldev.com/2167/java-exception-interview-questions-and-answers#comment-29676)

* 1. Adityasays:

[February 25, 2015 at 10:10 pm](https://www.journaldev.com/2167/java-exception-interview-questions-and-answers#comment-31828)

If a method in parent class throws checked exception, child class can not change the signature. The child can ignore it or add more unchecked exceptions.

If parent class method does not throw checked exception, the child class method cannot throw it either.

[Reply](https://www.journaldev.com/2167/java-exception-interview-questions-and-answers#comment-31828)

* 1. [Pankaj](https://www.journaldev.com/)says:

[June 28, 2016 at 6:27 am](https://www.journaldev.com/2167/java-exception-interview-questions-and-answers#comment-35468)

start() method is throwing checked exceptions, so the rule apply. foo() method is throwing un-checked exceptions and the rule doesn’t apply on them.

[Reply](https://www.journaldev.com/2167/java-exception-interview-questions-and-answers#comment-35468)

1. XCESSsays:

[July 31, 2014 at 10:32 am](https://www.journaldev.com/2167/java-exception-interview-questions-and-answers#comment-29650)

Had great Help

[Reply](https://www.journaldev.com/2167/java-exception-interview-questions-and-answers#comment-29650)

1. sampathsays:

[July 10, 2014 at 8:59 am](https://www.journaldev.com/2167/java-exception-interview-questions-and-answers#comment-29455)

Looks like there is mistake in answer for question 11

**final keyword can be used with class variables to make them immutable**

final keyword can used for variables to make variables can’t be re-assigned . it won’t make variables immutable, immutable means when we can’t change the object but we can re-assign a different object to same reference variable.

[Reply](https://www.journaldev.com/2167/java-exception-interview-questions-and-answers#comment-29455)

* 1. [Pankaj](https://www.journaldev.com/)says:

[July 10, 2014 at 11:32 am](https://www.journaldev.com/2167/java-exception-interview-questions-and-answers#comment-29457)

Yes you are right, updated the post. Thanks!!

[Reply](https://www.journaldev.com/2167/java-exception-interview-questions-and-answers#comment-29457)

1. Manjunatha Kurubasays:

[June 29, 2014 at 6:03 am](https://www.journaldev.com/2167/java-exception-interview-questions-and-answers#comment-29369)

try {  
foo();  
} catch (IOException | JAXBException e) {  
e = new Exception(“”);  
e.printStackTrace();  
}catch(Exception e){  
e = new Exception(“”);  
e.printStackTrace();  
}

only the exception object for IOException | JAXBException exceptions is final but exception object for “Exception” exception is not final.

[Reply](https://www.journaldev.com/2167/java-exception-interview-questions-and-answers#comment-29369)

* 1. [Pankaj](https://www.journaldev.com/)says:

[June 29, 2014 at 10:14 pm](https://www.journaldev.com/2167/java-exception-interview-questions-and-answers#comment-29374)

Yes, thats what I meant with multi-catch block.

[Reply](https://www.journaldev.com/2167/java-exception-interview-questions-and-answers#comment-29374)

1. Manjunathsays:

[May 28, 2014 at 3:05 am](https://www.journaldev.com/2167/java-exception-interview-questions-and-answers#comment-29026)

Great man… such a descriptive way of explanation..  
Thanks

[Reply](https://www.journaldev.com/2167/java-exception-interview-questions-and-answers#comment-29026)

1. Kalaisays:

[May 17, 2014 at 10:21 pm](https://www.journaldev.com/2167/java-exception-interview-questions-and-answers#comment-28901)

Hats off to you Pankaj ..Super ..good work….Never stop your blogging habbit…

[Reply](https://www.journaldev.com/2167/java-exception-interview-questions-and-answers#comment-28901)

1. sonalsays:

[May 9, 2014 at 10:31 am](https://www.journaldev.com/2167/java-exception-interview-questions-and-answers#comment-28836)

Great post “Hats Off To YOU”….. I have read many offline/online books n tutorials but ur understanding to  
subject is so deep that it reflects in ur posts….not only this topic but every of ur post …. exceptional knowledge and exceptional writing……. 🙂

Regards

Sonal

[Reply](https://www.journaldev.com/2167/java-exception-interview-questions-and-answers#comment-28836)

* 1. [Pankaj](https://www.journaldev.com/)says:

[May 10, 2014 at 12:03 am](https://www.journaldev.com/2167/java-exception-interview-questions-and-answers#comment-28841)

Thanks Sonal, your appreciation means a lot.

[Reply](https://www.journaldev.com/2167/java-exception-interview-questions-and-answers#comment-28841)

1. Sachidananda Dashsays:

[April 2, 2014 at 5:28 pm](https://www.journaldev.com/2167/java-exception-interview-questions-and-answers#comment-28415)

finally() method is executed by Garbage Collector before the object is destroyed is probably wrong , Ideally it should be finalize()

[Reply](https://www.journaldev.com/2167/java-exception-interview-questions-and-answers#comment-28415)

* 1. [Pankaj](https://www.journaldev.com/)says:

[April 2, 2014 at 7:29 pm](https://www.journaldev.com/2167/java-exception-interview-questions-and-answers#comment-28420)

Yes you are right, it was a typo. Thanks for pointing out, I have updated the post.

[Reply](https://www.journaldev.com/2167/java-exception-interview-questions-and-answers#comment-28420)

1. Rajiv Rahulsays:

[December 14, 2013 at 7:02 am](https://www.journaldev.com/2167/java-exception-interview-questions-and-answers#comment-26712)

Hi,  
Thank you so much for the great work that you are doing for us which is benefiting us in building our career. In fact Its our duty to appreciate people like you because you deserve it!

[Reply](https://www.journaldev.com/2167/java-exception-interview-questions-and-answers#comment-26712)

* 1. [Pankaj](https://www.journaldev.com/)says:

[June 28, 2016 at 6:34 am](https://www.journaldev.com/2167/java-exception-interview-questions-and-answers#comment-35470)

Thanks for the nice comments Rajiv. 🙂

[Reply](https://www.journaldev.com/2167/java-exception-interview-questions-and-answers#comment-35470)

1. Sidramsays:

[September 23, 2013 at 6:02 am](https://www.journaldev.com/2167/java-exception-interview-questions-and-answers#comment-23740)

It very nice explanation. I have one qns, what is the difference between Exception & RuntimeException. When i create custom exception by extedning RuntimeException which is becomes unchecked exception, How it can be when RuntimeException allready extends Exception class. How it will be differentites. Could you explain.

[Reply](https://www.journaldev.com/2167/java-exception-interview-questions-and-answers#comment-23740)

* 1. Irshadsays:

[October 28, 2013 at 7:10 pm](https://www.journaldev.com/2167/java-exception-interview-questions-and-answers#comment-24924)

There is a little bit change in answer of Question E is  
To fix this issue, we can either change the method singnature in subclass to be exact same as superclass or we can remove throws clause from subclass or can define the subclass of Exception in child class method as shown below.  
@Override  
public void start() throws FileNotFoundException{  
}

[Reply](https://www.journaldev.com/2167/java-exception-interview-questions-and-answers#comment-24924)

* + 1. zakirsays:

[July 13, 2014 at 6:02 am](https://www.journaldev.com/2167/java-exception-interview-questions-and-answers#comment-29468)

Good one…very informative post

[Reply](https://www.journaldev.com/2167/java-exception-interview-questions-and-answers#comment-29468)

**Leave a Reply**

Top of Form

Your email address will not be published. Required fields are marked \*

Comment

Name \*

Email \*



Bottom of Form

**[Advantages of VPN For Web Developers](https://www.journaldev.com/50562/advantages-of-vpn-for-web-developers?utm_source=vdoai_stories" \t "_blank)**



**Newsletter for You**

Top of Form



Bottom of Form



JournalDev is one of the most popular websites for Java, Python, Android, and related technical articles. Our tutorials are regularly updated, error-free, and complete. Every month millions of developers like you visit JournalDev to read our tutorials.

JournalDev was founded by Pankaj Kumar in 2010 to share his experience and learnings with the whole world. He loves Open source technologies and writing on JournalDev has become his passion.

**Most Popular**

* [Java / Java EE Tutorials](https://www.journaldev.com/java-tutorial-java-ee-tutorials)
* [Core Java Tutorial](https://www.journaldev.com/7153/core-java-tutorial)
* [Python Tutorials](https://www.journaldev.com/python-tutorial)
* [Java Interview Questions](https://www.journaldev.com/java-interview-questions)
* [Core Java Interview Questions](https://www.journaldev.com/2366/core-java-interview-questions-and-answers)
* [Java Design Patterns](https://www.journaldev.com/1827/java-design-patterns-example-tutorial)
* [Spring Tutorial](https://www.journaldev.com/2888/spring-tutorial-spring-core-tutorial)

**Favorite Sites**

* [AskPython](https://www.askpython.com/)
* [BloggerTipsAndTricks](https://www.bloggertipsandtricks.com/)
* [GoLangDocs](https://golangdocs.com/)
* [LinuxForDevices](https://www.linuxfordevices.com/)
* [Mkyong](https://www.mkyong.com/)
* [VM-Help](https://www.vm-help.com/)
* [WP-Design](https://wp-design.org/)

© 2021 · [Privacy Policy](https://www.journaldev.com/privacy-policy) · [Terms and Conditions](https://www.journaldev.com/terms-and-conditions) · [Contact Us](https://www.journaldev.com/contact) · [About](https://www.journaldev.com/about) · Part of JournalDev IT Services Private Limited

[DZone](https://dzone.com/)

[Java Zone](https://dzone.com/java-jdk-development-tutorials-tools-news)

Over 2 million developers have joined DZone.

[Log In](https://dzone.com/users/login.html) / [Join](https://dzone.com/static/registration.html)

* [**REFCARDZ**](https://dzone.com/refcardz)

* [**RESEARCH**](https://dzone.com/research)

* [**WEBINARS**](https://dzone.com/webinars)

* |

* + [Agile](https://dzone.com/agile-methodology-training-tools-news)

* + [AI](https://dzone.com/artificial-intelligence-tutorials-tools-news)

* + [Big Data](https://dzone.com/big-data-analytics-tutorials-tools-news)

* + [Cloud](https://dzone.com/cloud-computing-tutorials-tools-news)

* + [Database](https://dzone.com/database-sql-nosql-tutorials-tools-news)

* + [DevOps](https://dzone.com/devops-tutorials-tools-news)

* + [Integration](https://dzone.com/enterprise-integration-training-tools-news)

* + [IoT](https://dzone.com/iot-developer-tutorials-tools-news-reviews)

* + [Java](https://dzone.com/java-jdk-development-tutorials-tools-news)

* + [Microservices](https://dzone.com/microservices-news-tutorials-tools)

* + [Open Source](https://dzone.com/open-source-news-tutorials-tools)

* + [Performance](https://dzone.com/apm-tools-performance-monitoring-optimization)

* + [Security](https://dzone.com/application-web-network-security)

* + [Web Dev](https://dzone.com/web-development-programming-tutorials-tools-news)

[DZone](https://dzone.com/) > [Java Zone](https://dzone.com/java-jdk-development-tutorials-tools-news) > [Java Exceptions Handling Interview Questions and Answers](https://dzone.com/articles/java-exception-handling-interview-questions-and-an-1)

**Java Exceptions Handling Interview Questions and Answers**

Let's take a look at key these Java exceptions handling questions and answers that are important for any Java developer preparing for an interview.

[**[](https://dzone.com/users/3439641/ramesh-fadatare.html)**](https://dzone.com/users/3439641/ramesh-fadatare.html)**by**

[**Ramesh Fadatare**](https://dzone.com/users/3439641/ramesh-fadatare.html)

**·**

**Nov. 14, 18 ·**[**Java Zone**](https://dzone.com/java-jdk-development-tutorials-tools-news)**· Presentation**

**Like**[**(15)**](https://dzone.com/articles/java-exception-handling-interview-questions-and-an-1)

**Comment (1)**

**Save**

[**Tweet**](https://dzone.com/articles/java-exception-handling-interview-questions-and-an-1)

**58.91K Views**

Join the DZone community and get the full member experience.

[**JOIN FOR FREE**](https://dzone.com/static/registration.html)

In this article, we will discuss 21 important [Java exception handling](http://www.javaguides.net/p/java-exception-handling-tutorial.html) interview questions with the answers.

**Before we get started, check out this  [complete beginner-to-expert, in-depth exception handling tutorial in Java](http://www.javaguides.net/p/java-exception-handling-tutorial.html" \t "_blank)!**

Let's explore the following questions, which are likely to appear in technical interviews for Java developers regarding exceptions.

1. What is an exception in Java?
2. How does exception handling work in Java?
3. What are exception handling keywords in Java?
4. What is the purpose of the throw and throws keywords?
5. How can you handle an exception?
6. Explain the Java exception hierarchy.
7. How can you catch multiple exceptions?
8. What is the difference between checked and unchecked exceptions in Java?
9. What is the difference between throw and throws keyword in Java?
10. What is the difference between an exception and error?
11. What is the OutOfMemoryError in Java?
12. What are chained exceptions in Java?
13. How do you write a custom exception in Java?
14. What is the difference between final, finally, and finalize in Java?
15. What happens when an exception is thrown by the main method?
16. What is a try-with-resources statement?
17. What is a stacktrace and how does it relate to an exception?
18. What are the advantages of Java exceptions?
19. Can you throw any exception inside a lambda expression’s body?
20. What are the rules that we need to follow when overriding a method that throws an exception?
21. What are some of the exception handling best practices?

**1. What Is an Exception in Java?**

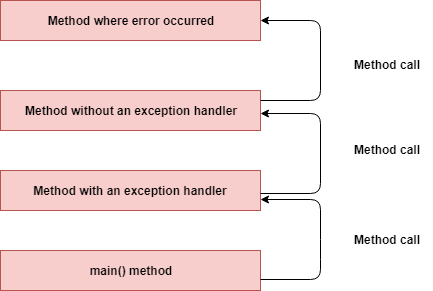
An exception is an event that occurs during the execution of a program and disrupts the normal flow of the program's instructions.

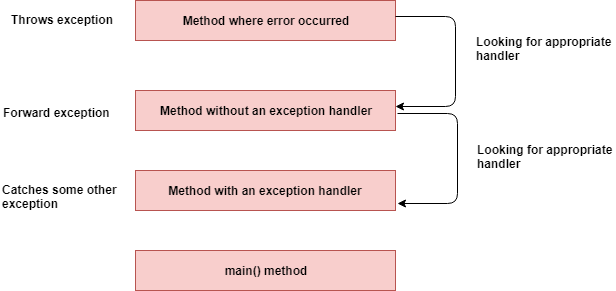
Read more about exceptions in Java in the [Java Exception Handling Guide](http://www.javaguides.net/2018/08/java-exception-handling-guide.html)!

**2. How Does Exception Handling Work in Java?**

Below steps demonstrates how the exception handling works in Java:

**Step 1:** When an error occurs within a method, the method creates an object and hands it off to the runtime system this object is called an *exception object*. The exception object contains information about the error, including its type and the state of the program when the error occurred. Creating an exception object and handing it to the runtime system is called *throwing an exception*.

**Step 2:** After a method throws an exception, the runtime system attempts to find something to handle it. The set of possible "somethings" to handle the exception is the ordered list of methods that had been called to get to the method where the error occurred. The list of methods is known as the call stack. The following diagram shows the call stack of three method calls, where the first method called has the *exception handler.*  
  
[](https://4.bp.blogspot.com/-0Wo56QV6xn4/XCS7VWHL0qI/AAAAAAAAFNo/8O8OQJlQmIkqoMITTOt8T-H37z5flfeTgCLcBGAs/s1600/exception-call+stack.png)

**Step 3:** The runtime system searches the call stack for a method that contains a block of code that can handle the exception. This block of code is called an exception handler. The search begins with the method in which the error occurred and proceeds through the call stack in the reverse order in which the methods were called. When an appropriate handler is found, the runtime system passes the exception to the handler.   
An exception handler is considered appropriate if the type of the exception object thrown matches the type that can be handled by the handler.  
  
**Step 4:**The exception handler chosen is said to catch the exception. If the runtime system exhaustively searches all the methods on the call stack without finding an appropriate exception handler, as shown in the following diagram, the runtime system (and, consequently, the program) terminates.  
[](https://3.bp.blogspot.com/-FNi4uc7THNY/XCS7c7FvTpI/AAAAAAAAFNs/3YB9a811_yEss_ky-ku675sg6OH2HJdHwCLcBGAs/s1600/exception-searching-call+stack.png)

Read more at [How the Exception Handling Works in Java with Example](http://www.javaguides.net/2018/08/how-exception-handling-works-in-java.html).

**3. What Are  Exception Handling Keywords in Java?**

Java exception handling is managed via five keywords:

**1. try:**Enclose the code that might throw an exception within a try block. If an exception occurs within the try block, that exception is handled by an exception handler associated with it. The try block contains at least one catch block or finally block.

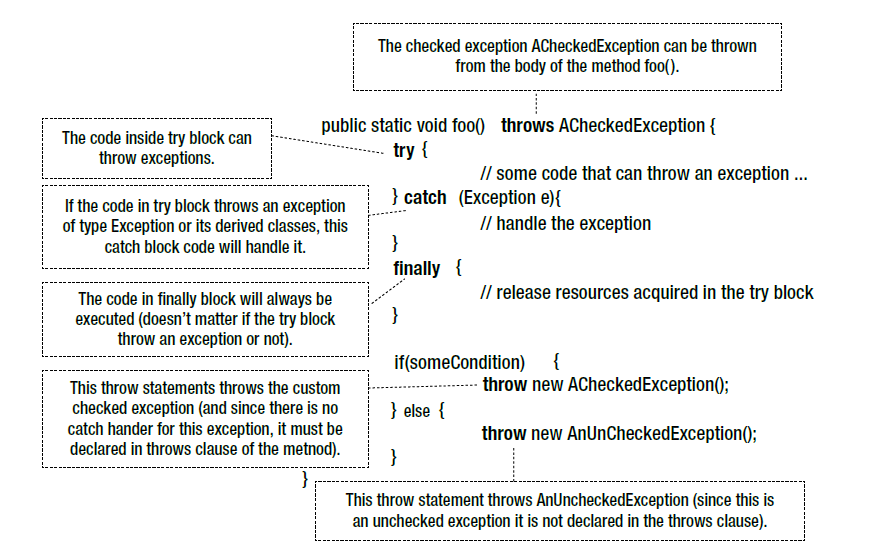
**2. catch:** The Java catch block is used to handle the exception. It must be used after the try block only. You can use multiple catch blocks with a singletry.

**3. throw:** Sometimes, we explicitly want to create an exception object and then throw it to halt the normal processing of the program. The throw keyword is used to throw an exception to the run-time to handle it.

**4. throws:** When we are throwing any checked exception in a method and not handling it, then we need to use the throws keyword in the method signature to let the caller program know that the exceptions might be thrown by the method. The caller method might handle these exceptions or propagate it to its caller method using the throws keyword. We can provide multiple exceptions in the throws clause and it can be used with the main()method also.

**5. finally:**The finally block is optional and can be used only with thetry-catch block. Since exception halts the process of execution, we might have some resources open that will not get closed, so we can use finally block. The finally block gets executed always, whether an exception occurs or not.

This diagram provides a summary of the usage of these keywords.

[](https://1.bp.blogspot.com/-VBriy2lMQrs/W8IlIHxwAMI/AAAAAAAAERU/u0OlZUp30VkB_FV44BE0n3R9LcMkEFTWQCLcBGAs/s1600/exception-handling-keywords.PNG)

Read more about all five keywords at [Exception Handling Keywords in Java with Examples](http://www.javaguides.net/2018/08/exception-handling-keywords-in-java.html).

**4. What Is the Purpose of the Throw and Throws keywords?**

Thethrows keyword is used to specify that a method may raise an exception during its execution. It enforces explicit exception handling when calling a method:

1

public void simpleMethod() throws Exception {

2

// ...

3

}

Thethrow keyword allows us to throw an exception object to interrupt the normal flow of the program. This is most commonly used when a program fails to satisfy a given condition:

1

if (task.isTooComplicated()) {

2

throw new TooComplicatedException("The task is too complicated");

3

}

Read more about the throws and throw keywords here: [Java throw Keyword with Example](http://www.javaguides.net/2018/08/java-throw-keyword.html); [Java throws Keyword with Example](http://www.javaguides.net/2018/08/java-throws-keyword.html).

**5. How Can You Handle an Exception?**

You can handle an exception by using a try-catch-finallystatement:

1

try {

2

// ...

3

} catch (ExceptionType1 ex) {

4

// ...

5

} catch (ExceptionType2 ex) {

6

// ...

7

} finally {

8

// ...

9

}

The block of code in which an exception may occur is enclosed in a try block. This block is also called “protected” or “guarded” code. If an exception occurs, the catch block that matches the exception being thrown is executed. If not, all catch blocks are ignored. The finally block is always executed after the try block exits, whether an exception was thrown inside it or not.

Read more about how to handle exceptions here: [Java Exception Handling Tutorial](http://www.javaguides.net/p/java-exception-handling-tutorial.html).

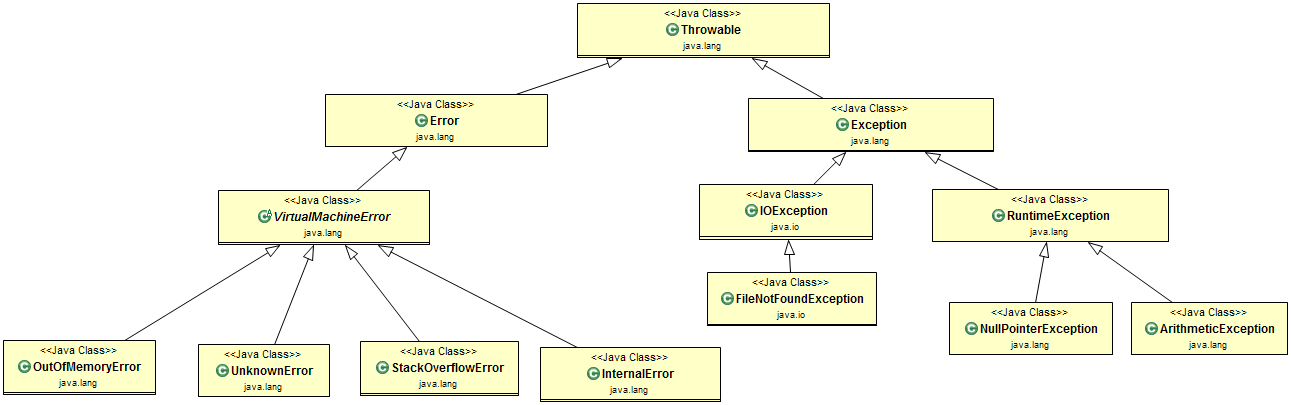
**6. Explain the Java Exception Hierarchy.**

The objects that inherit from the [Throwable](http://www.javaguides.net/2018/08/javalangthrowable-class-in-java.html) class that includes direct descendants (objects that inherit directly from the [Throwable](http://www.javaguides.net/2018/08/javalangthrowable-class-in-java.html) class) and indirect descendants (objects that inherit from children or grandchildren of the [Throwable](http://www.javaguides.net/2018/08/javalangthrowable-class-in-java.html) class).

[Throwable](http://www.javaguides.net/2018/08/javalangthrowable-class-in-java.html) has two direct descendants:

1. Error Class
2. Exception Class

The figure below illustrates the class hierarchy of the [Throwable](http://www.javaguides.net/2018/08/javalangthrowable-class-in-java.html) class and its most significant subclasses.

[](https://2.bp.blogspot.com/--f8P24mVPK0/W3v8Qq2JzrI/AAAAAAAADSA/PR-7hjBxP1QPhzc5UsYyZBgFseJ7QM2XgCLcBGAs/s1600/exception-handling.png)

**Error Class:**When a dynamic linking failure or other hard failures in the Java Virtual Machine occurs, the virtual machine throws an Error.

Examples: VirtualMachineError,  OutOfMemoryError, UnKnownError,  StackOverflowError, etc.

**Exception Class:**Most programs throw and catch objects that derive from the exception class. An exception indicates that a problem occurred, but it is not a serious system problem. For example, when dealing with the FileNotFoundException, we should catch this exception and provide a useful message to the user and log it properly for debugging purposes. The exception is the parent class of all checked exceptions.

**RuntimeException Class:**This provides one exception subclass, RuntimeException, that is reserved for exceptions that indicate an incorrect use of an API. An example of a runtime exception is the NullPointerException, which occurs when a method tries to access a member of an object through a null reference.

Read more about the exception hierarchy in Java here: [Exceptions Hierarchy in Java](http://www.javaguides.net/2018/08/exceptions-hierarchy-in-java.html).

**7. How Can you Catch Multiple Exceptions?**

There are three ways to handle multiple exceptions in a block of code.

The first is to use a catch block that can handle all exception types being thrown:

1

try {

2

// ...

3

} catch (Exception ex) {

4

// ...

5

}

You should keep in mind that the recommended practice is to use exception handlers that are as accurate as possible.

Exception handlers that are too broad can make your code more error-prone, catch exceptions that weren’t anticipated, and cause unexpected behavior in your program.

The second way is implementing multiple catch blocks:

1

try {

2

// ...

3

} catch (FileNotFoundException ex) {

4

// ...

5

} catch (EOFException ex) {

6

// ...

7

}

Note that if the exceptions have an inheritance relationship, the child type must come first and the parent type later. If we fail to do this, it will result in a compilation error.

The third is to use a multi-catch block:

1

try {

2

// ...

3

} catch (FileNotFoundException | EOFException ex) {

4

// ...

5

}

This feature, first introduced in Java 7, reduces code duplication and makes it easier to maintain.

Read more at [Java try/catch Block with Examples](http://www.javaguides.net/2018/08/java-trycatch-block.html).

**8. What Is the Difference Between Checked and Unchecked Exceptions in Java?**

1. Checked exceptions should be handled in the code using a try-catch block, or else, the method should use the throws keyword to let the caller know about the checked exceptions that might be thrown from the method. Unchecked Exceptions are not required to be handled in the program or to mention them in thethrows clause of the method.

2. The exception is the superclass of all checked exceptions, whereas RuntimeException is the superclass of all unchecked exceptions. Note that RuntimeException is the child class ofException.

3. Checked exceptions are error scenarios that require being handled in the code, or else, you will get a compile-time error. For example, if you use FileReader to read a file, it throws the FileNotFoundException and we must catch it in the [try-catch block](http://www.javaguides.net/2018/08/java-trycatch-block.html) or throw it again to the caller method. Unchecked exceptions are mostly caused by poor programming, for example, the NullPointerException when invoking a method on an object reference without making sure that it’s not null. I can write a method to remove all the vowels from the string. It’s the caller's responsibility to make sure not to pass a null string. I might change the method to handle these scenarios, but ideally, the caller should take care of this.

4. Checked and unchecked exceptions are also known as compile-time and run-time exceptions respectively.

**9. What Is the Difference Between Throw and Throws Keywords in Java?**

The throws keyword is used with a method signature to declare exceptions that the method might throw, whereas the throw keyword is used to disrupt the flow of a program and handing over the exception object to run-time to handle it.

Read more at [Difference Between Throw and Throws in Java](https://www.baeldung.com/java-throw-throws).

**10. What Is the Difference Between an Exception and an Error?**

An exception is an event that represents a condition from which it is possible to recover, whereas an error represents an external situation usually impossible to recover from.

All errors thrown by the JVM are instances of Error or one of its subclasses. The more common ones include:

* OutOfMemoryError – thrown when the JVM cannot allocate more objects because it is out memory and the garbage collector was unable to make more available.
* StackOverflowError – occurs when the stack space for a thread has run out. This is typically because an application recurses too deeply.
* ExceptionInInitializerError – signals that an unexpected exception occurred during the evaluation of a static initializer.
* NoClassDefFoundError – is thrown when the classloader tries to load the definition of a class and couldn’t find it, usually because the required class files were not found in the classpath.
* UnsupportedClassVersionError – occurs when the JVM attempts to read a class file and determines that the version in the file is not supported, normally because the file was generated with a newer version of Java

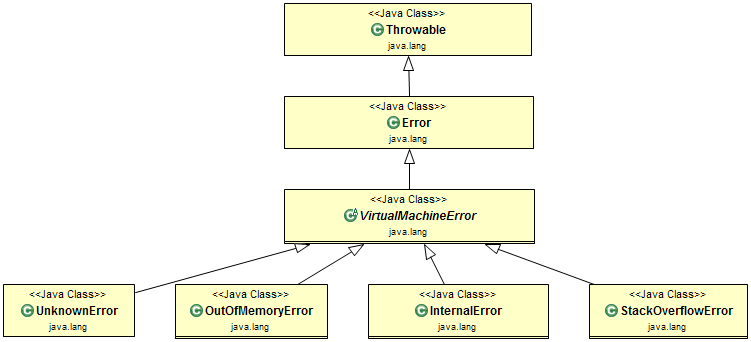
Although an error can be handled with a try statement, this is not a recommended practice since there is no guarantee that the program will be able to do anything reliably after the error was thrown.

Read more on this subject here: [Exceptions Hierarchy in Java](http://www.javaguides.net/2018/08/exceptions-hierarchy-in-java.html).

**11. What Is the OutOfMemoryError in Java?**

The OutOfMemoryError in Java is a subclass of the java.lang. VirtualMachineError and it’s thrown by the JVM when it runs out of heap memory.

The figure below illustrates the class hierarchy of the Error class.

[](https://1.bp.blogspot.com/-UcUc2b2yhdk/W3v8auLcMRI/AAAAAAAADSI/2zPwQZBAgncNI5TVjNam8QEHWhdNAfEOQCLcBGAs/s1600/errorclass-hierachy.png)

We can fix this error by providing more memory to run the Java application through Java options.

1

$>java MyProgram -Xms1024m -Xmx1024m -XX:PermSize=64M -XX:MaxPermSize=256m

**12. What Is a Chained Exception in Java?**

The chained exception feature allows you to associate another exception with an exception. This second exception describes the cause of the first exception.

For example, imagine a situation where a method throws an ArithmeticException because of an attempt to divide by zero. However, the actual cause of the problem was that an I/O error occurred, which caused the divisor to be set improperly. Although the method must certainly throw an ArithmeticExceptionsince that is the error that occurred, you might also want to let the calling code know that the underlying cause was an I/O error. Chained exceptions let you handle this and any other situation in which layers of exceptions exist. This concept was introduced in JDK 1.4.

Read more about chained exception feature here: [Java Chained Exceptions with Example](http://www.javaguides.net/2018/08/java-chained-exceptions-example.html).

**13. How Can We Write a Custom Exception in Java?**

In bigger applications, most of the cases we need custom exceptions for representing business exceptions are at a level higher than technical exceptions defined by the JDK.

Here are the steps that create a custom exception:

* Create a new class whose name should end with an Exception, like the ClassNameException. This is a convention to differentiate an exception class from regular ones.
* Make the class extends one of the exceptions that are subtypes of the  java.lang.Exception class. Generally, a custom exception class always extends directly from the Exception class.
* Create a constructor with a String parameter, which is the detail message of the exception. In this constructor, simply call the super constructor and pass the message. In Java, there are two types of exceptions — checked and unchecked exceptions.

A simple example of a custom exception is shown below.

1

public class ResourceNotFoundException extends Exception {

2

private static final long serialVersionUID = 1L;

3

​

4

public ResourceNotFoundException(Object resourId) {

5

super(resourId != null ? resourId.toString() : null);

6

}

7

}

Read more in detail at [Guide to Create Custom Exceptions](http://www.javaguides.net/2018/06/guide-to-create-custom-exceptions.html).

**14. What Is the Difference Between Final, Finally, and Finalize in Java?**

**1. final:** is used to apply restrictions on the class, method, and variable. The finalclass can't be inherited — nor can it be overridden or changed.

**2. finally:** this keyword is used with the try-catch block to provide statements that will always get executed even if some exception arises. Usually, finally is used to close resources.

**3. finalize:** is used to perform clean up processing just before the object is garbage collected.

**15. What Happens When an Exception Is Thrown by the Main Method?**

When an exception is thrown by [main() method](http://www.javaguides.net/2018/10/java-main-method-interview-questions-with-answers.html), Java Runtime terminates the program and prints the exception message and the stack trace in-system console.

**16. What Is the Try-With-Resources Statement?**

In Java, the try-with-resourcesstatement is a try statement that declares one or more resources. The resource is as an object that must be closed after finishing the program. The try-with-resources statement ensures that each resource is closed at the end of the statement execution.

For example:

1

public class BufferedReaderExample {

2

public static void main(String[] args) {

3

try (FileReader fr = new FileReader("C:/workspace/java-io-guide/sample.txt"); BufferedReader br = new BufferedReader(fr);) {

4

String sCurrentLine;

5

​

6

while ((sCurrentLine = br.readLine()) != null) {

7

System.out.println(sCurrentLine);

8

}

9

} catch (IOException e) {

10

e.printStackTrace();

11

}

12

}

13

}

Read more about the try-with-resource statement here: [The try-with-resources Statement with Examples](http://www.javaguides.net/2018/08/the-try-with-resources-statement-with-examples.html)

**17. What Is a Stacktrace and How Does it Relate to an Exception?**

A stack trace provides the names of the classes and methods that were called, from the start of the application to the point that an exception occurred.

It’s a very useful debugging tool since it enables us to determine exactly where the exception was thrown in the application and the original causes that led to it.

**18. What Are the Advantages of Java Exceptions?**

The following are advantages of using exceptions in your programs:

1. Separating Error-handling code from "regular" code

2: Propagating errors up the call stack

3: Grouping and differentiating error types

Read more detail about each advantage here at [Advantages of Java Exceptions with Examples](http://www.javaguides.net/2018/08/advantages-of-java-exceptions.html).

**19. Can You Throw any Exception Inside a Lambda Expression’s Body?**

When using a standard functional interface already provided by Java, you can only throw unchecked exceptions because standard functional interfaces do not have a “throws” clause in its method signatures:

1

List<Integer> integers = Arrays.asList(3, 9, 7, 0, 10, 20);

2

integers.forEach(i -> {

3

if (i == 0) {

4

throw new IllegalArgumentException("Zero not allowed");

5

}

6

System.out.println(Math.PI / i);

7

});

However, if you are using a custom functional interface, throwing checked exceptions is possible:

1

@FunctionalInterface

2

public static interface CheckedFunction<T> {

3

void apply(T t) throws Exception;

4

}

1

public void processTasks(

2

List<Task> taks, CheckedFunction<Task> checkedFunction) {

3

for (Task task : taks) {

4

try {

5

checkedFunction.apply(task);

6

} catch (Exception e) {

7

// ...

8

}

9

}

10

}

11

​

12

processTasks(taskList, t -> {

13

// ...

14

throw new Exception("Something happened");

15

});

**20. What Are the Rules We Need to Follow When Overriding a Method That Throws an Exception?**

Several rules dictate how exceptions must be declared in the context of inheritance.

When the parent class method doesn’t throw any exceptions, the child class method can’t throw any checked exceptions, but it may throw any unchecked. Here’s an example code to demonstrate this:

1

class Parent {

2

void doSomething() {

3

// ...

4

}

5

}

6

​

7

class Child extends Parent {

8

void doSomething() throws IllegalArgumentException {

9

// ...

10

}

11

}

The next example will fail to compile since the overriding method throws a checked exception not declared in the overridden method:

1

class Parent {

2

void doSomething() {

3

// ...

4

}

5

}

6

​

7

class Child extends Parent {

8

void doSomething() throws IOException {

9

// Compilation error

10

}

11

}

When the parent class method throws one or more checked exceptions, the child class method can throw any unchecked exception, including all, none, or a subset of the declared checked exceptions and an even a greater number of these as long as they have the same scope or narrower. Here’s an example code that successfully follows the previous rule:

1

class Parent {

2

void doSomething() throws IOException, ParseException {

3

// ...

4

}

5

​

6

void doSomethingElse() throws IOException {

7

// ...

8

}

9

}

10

​

11

class Child extends Parent {

12

void doSomething() throws IOException {

13

// ...

14

}

15

​

16

void doSomethingElse() throws FileNotFoundException, EOFException {

17

// ...

18

}

19

}

Note that both methods respect the rule. The first throws fewer exceptions than the overridden method, and the second, even though it throws more, they’re narrower in scope. However, if we try to throw a checked exception that the parent class method doesn’t declare or we throw one with a broader scope, we’ll get a compilation error:

1

class Parent {

2

void doSomething() throws FileNotFoundException {

3

// ...

4

}

5

}

6

​

7

class Child extends Parent {

8

void doSomething() throws IOException {

9

// Compilation error

10

}

11

}

When theparentclass method has a throws clause with an unchecked exception, the childclass method can throw none or any number of unchecked exceptions, even though they are not related. Here’s an example that honors that rule:

1

class Parent {

2

void doSomething() throws IllegalArgumentException {

3

// ...

4

}

5

}

6

​

7

class Child extends Parent {

8

void doSomething()

9

throws ArithmeticException, BufferOverflowException {

10

// ...

11

}

12

}

**21. Java Exception Handling Best Practices**

1. Clean up resources in a finally block or use a [try-with-resources statement](http://www.javaguides.net/2018/08/the-try-with-resources-statement-with-examples.html)
2. Throw a specific exception
3. Do not catch the Exception class rather catch specific subclasses
4. Never catch a Throwable class
5. Always correctly wrap the exceptions in custom exceptions so that the stack trace is not lost
6. Catch the most specific exception first
7. Don’t ignore exceptions rather log the exceptions
8. Never throw any exception from the  finally block
9. Don’t use the printStackTrace() statement or similar methods
10. Use the finally blocks instead of catch blocks if you are not going to handle the exception
11. Validate user input to catch adverse conditions very early in the request processing
12. Throw exceptions with descriptive messages

Learn each exception handling best practices with examples on[Java Exception Handling Best Practices](http://www.javaguides.net/2018/06/guide-to-java-exception-handling-best-practices.html)

**I kindly suggest that if you have other interview questions regarding Java exception handling that will help others prepare for their interview, leave a note in the comments below!**

**References**

[Java Exception Handling Tutorial](http://www.javaguides.net/p/java-exception-handling-tutorial.html)

[Java Exception Interview Questions and Answers - JournalDev](https://www.journaldev.com/2167/java-exception-interview-questions-and-answers)

[Java Exceptions Interview Questions (+ Answers) - Baeldung](https://www.baeldung.com/java-exceptions-interview-questions)

Topics:

JAVA, EXCEPTIONS, QUESTIONS ANSWERS, DEVELOPERS, CODE, CHECKED EXCEPTIONS, UNCHECKED

Published at DZone with permission of Ramesh Fadatare. [See the original article here.](http://www.javaguides.net/2018/11/java-exception-handling-interview-questions-and-answers.html)

Opinions expressed by DZone contributors are their own.

**Popular on DZone**

* [**10 Microservices Interview Questions You Must Know**](https://dzone.com/articles/microservices-interview-questions-answers?fromrel=true)
* [**High-Performance Batch Processing Using Apache Spark and Spring Batch**](https://dzone.com/articles/using-apache-spark-and-spring-batch-for-processing?fromrel=true)
* [**API Security Weekly: Issue #134**](https://dzone.com/articles/api-security-weekly-issue-134?fromrel=true)
* [**Event-Driven APIs and Schema Governance for Apache Kafka**](https://dzone.com/articles/event-driven-apis-and-schema-governance-for-apache?fromrel=true)

**ABOUT US**

* [**About DZone**](https://dzone.com/pages/about)
* [**Send feedback**](mailto:support@dzone.com)
* [**Careers**](https://devada.com/careers/)
* [**Sitemap**](https://dzone.com/sitemap)

**ADVERTISE**

* [**Developer Marketing Blog**](https://devada.com/blog/developer-marketing)
* [**Advertise with DZone**](https://dzone.com/pages/advertise)
* [**+1 (919) 238-7100**](tel:+19192387100)

**CONTRIBUTE ON DZONE**

* [**Article Submission Guidelines**](https://dzone.com/articles/dzones-article-submission-guidelines)
* [**MVB Program**](https://dzone.com/pages/mvb)
* [**Become a Contributor**](https://dzone.com/pages/contribute)
* [**Visit the Writers' Zone**](https://dzone.com/writers-zone)

**LEGAL**

* [**Terms of Service**](https://dzone.com/pages/tos)
* [**Privacy Policy**](https://dzone.com/pages/privacy)

**CONTACT US**

* **600 Park Offices Drive**
* **Suite 150**
* **Research Triangle Park, NC 27709**
* [**support@dzone.com**](mailto:support@dzone.com)
* [**+1 (919) 678-0300**](tel:+19196780300)

**Let's be friends:**

**DZone.com is powered by**

[Skip to content](https://javaconceptoftheday.com/java-exception-handling-interview-questions-and-answers/#content)

[**Java Concept Of The Day**](https://javaconceptoftheday.com/)

**Java Tutorial Site For Beginners**

* [Home](https://javaconceptoftheday.com/)
* [Arrays](https://javaconceptoftheday.com/arrays-in-java/)
* [Strings](https://javaconceptoftheday.com/introduction-strings/)
* [Exception Handling](https://javaconceptoftheday.com/exception-handling-java/)
* [Threads](https://javaconceptoftheday.com/threads-in-java/)
* [Generics](https://javaconceptoftheday.com/why-we-need-generics-in-java/)
* [Collections](https://javaconceptoftheday.com/collection-framework-class-hierarchy/)
* [JDBC](https://javaconceptoftheday.com/java-jdbc-tutorial/)
* [Interview Programs](https://javaconceptoftheday.com/most-asked-java-interview-programs-with-solutions/)

Top of Form

[Expand Search Form](https://javaconceptoftheday.com/java-exception-handling-interview-questions-and-answers/#search-menu)

Bottom of Form

[30 Java Exception Handling Interview Questions And Answers](https://javaconceptoftheday.com/java-exception-handling-interview-questions-and-answers/)

[pramodbablad](https://javaconceptoftheday.com/author/pramodbablad/) October 4, 2015 [7](https://javaconceptoftheday.com/java-exception-handling-interview-questions-and-answers/#comments)

Java exception handling is one of the favorite topic of the many interviewers to test candidate’s basic Java skills. In this post, I have shared 30 most asked Java exception handling interview questions and answers. I hope it will help you.

**30 Most Asked Java Exception Handling Interview Questions And Answers**

**1) What is an exception?**

Exception is an abnormal condition which occurs during the execution of a program and disrupts normal flow of the program. This exception must be handled properly. If it is not handled, program will be terminated abruptly.

**2) How the exceptions are handled in java? OR Explain exception handling mechanism in java?**

Exceptions in java are handled using try, catch and finally blocks.

try block : The code or set of statements which are to be monitored for exception are kept in this block.

catch block : This block catches the exceptions occurred in the try block.

finally block : This block is always executed whether exception is occurred in the try block or not and occurred exception is caught in the catch block or not.

**3) What is the difference between error and exception in java?**

Errors are mainly caused by the environment in which an application is running. For example, OutOfMemoryError happens when JVM runs out of memory. Where as exceptions are mainly caused by the application itself. For example, NullPointerException occurs when an application tries to access null object.

Click [here](https://javaconceptoftheday.com/difference-between-error-vs-exception-in-java/) to see more about Error Vs Exception in java.

**4) Can we keep other statements in between try, catch and finally blocks?**

No. We shouldn’t write any other statements in between try, catch and finally blocks. They form a one unit.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18 | try  {      // Statements to be monitored for exceptions  }    //You can't keep statements here    catch(Exception ex)  {      //Cathcing the exceptions here  }    //You can't keep statements here    finally  {      // This block is always executed  } |

**5) Can we write only try block without catch and finally blocks?**

No, It shows compilation error. The try block must be followed by either catch or finally block. You can remove either catch block or finally block but not both.

**6) There are three statements in a try block – statement1, statement2 and statement3. After that there is a catch block to catch the exceptions occurred in the try block. Assume that exception has occurred in statement2. Does statement3 get executed or not?**

No. Once a try block throws an exception, remaining statements will not be executed. control comes directly to catch block.

**7) What is unreachable catch block error?**

When you are keeping multiple catch blocks, the order of catch blocks must be from most specific to most general ones. i.e sub classes of Exception must come first and super classes later. If you keep super classes first and sub classes later, compiler will show unreachable catch block error.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22 | public class ExceptionHandling  {      public static void main(String[] args)      {          try          {              int i = Integer.parseInt("abc");   //This statement throws NumberFormatException          }            catch(Exception ex)          {              System.out.println("This block handles all exception types");          }            catch(NumberFormatException ex)          {              //Compile time error              //This block becomes unreachable as              //exception is already caught by above catch block          }      }  } |

**8) Explain the hierarchy of exceptions in java?**

Click [here](https://javaconceptoftheday.com/hierarchy-exceptions-java/) to see the hierarchy of exceptions in java.

**9) What are run time exceptions in java. Give example?**

The exceptions which occur at run time are called as run time exceptions. These exceptions are unknown to compiler. All sub classes of java.lang.RunTimeException and java.lang.Error are run time exceptions. These exceptions are unchecked type of exceptions. For example, NumberFormatException, NullPointerException, ClassCastException, ArrayIndexOutOfBoundException, StackOverflowError etc.

**10) What is OutOfMemoryError in java?**

OutOfMemoryError is the sub class of java.lang.Error which occurs when JVM runs out of memory.

**11) what are checked and unchecked exceptions in java?**

Checked exceptions are the exceptions which are known to compiler. These exceptions are checked at compile time only. Hence the name checked exceptions. These exceptions are also called compile time exceptions. Because, these exceptions will be known during compile time.

Unchecked exceptions are those exceptions which are not at all known to compiler. These exceptions occur only at run time. These exceptions are also called as run time exceptions. All sub classes of java.lang.RunTimeException and java.lang.Error are unchecked exceptions.

Click [here](https://javaconceptoftheday.com/checked-unchecked-exceptions-java/) to see more about checked and unchecked exceptions.

**12) What is the difference between ClassNotFoundException and NoClassDefFoundError in java?**

Click [here](https://javaconceptoftheday.com/classnotfoundexception-vs-noclassdeffounderror-in-java/) to see the differences between ClassNotFoundException and NoClassDefFoundError in java.

**13) Can we keep the statements after finally block If the control is returning from the finally block itself?**

No, it gives unreachable code error. Because, control is returning from the finally block itself. Compiler will not see the statements after it. That’s why it shows unreachable code error.

**14) Does finally block get executed If either try or catch blocks are returning the control?**

Yes, finally block will be always executed no matter whether try or catch blocks are returning the control or not.

**15) Can we throw an exception manually? If yes, how?**

Yes, we can throw an exception manually using throw keyword. Syntax for throwing an exception manually is

**throw InstanceOfThrowableType;**

Below example shows how to use throw keyword to throw an exception manually.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10 | try  {      NumberFormatException ex = new NumberFormatException();    //Creating an object to NumberFormatException explicitly        throw ex;        //throwing NumberFormatException object explicitly using throw keyword  }  catch(NumberFormatException ex)  {      System.out.println("explicitly thrown NumberFormatException object will be caught here");  } |

Click [here](https://javaconceptoftheday.com/throwing-rethrowing-exception-java/) to see more about throw keyword.

**16) What is Re-throwing an exception in java?**

Exceptions raised in the try block are handled in the catch block. If it is unable to handle that exception, it can re-throw that exception using throw keyword. It is called re-throwing an exception.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11 | try  {      String s = null;      System.out.println(s.length());   //This statement throws NullPointerException  }  catch(NullPointerException ex)  {      System.out.println("NullPointerException is caught here");        throw ex;     //Re-throwing NullPointerException  } |

**17) What is the use of throws keyword in java?**

Click [here](https://javaconceptoftheday.com/throws-keyword-java/) to see the uses of throws keyword in java.

**18) Why it is always recommended that clean up operations like closing the DB resources to keep inside a finally block?**

Because finally block is always executed whether exceptions are raised in the try block or not and raised exceptions are caught in the catch block or not. By keeping the clean up operations in finally block, you will ensure that those operations will be always executed irrespective of whether exception is occurred or not.

**19) What is the difference between final, finally and finalize in java?**

Click [here](https://javaconceptoftheday.com/difference-between-final-finally-and-finalize-in-java/) to see the differences between final, finally and finalize in java.

**20) How do you create customized exceptions in java?**

Click [here](https://javaconceptoftheday.com/user-defined-exceptions-java/) to see about customized exceptions in java.

**21) What is ClassCastException in java?**

ClassCastException is a RunTimeException which occurs when JVM unable to cast an object of one type to another type.

**22) What is the difference between throw, throws and throwable in java?**

Click [here](https://javaconceptoftheday.com/difference-between-throw-throws-and-throwable-in-java/) to see the differences between throw, throws and throwable in java.

**23) What is StackOverflowError in java?**

StackOverflowError is an error which is thrown by the JVM when stack overflows.

**24) Can we override a super class method which is throwing an unchecked exception with checked exception in the sub class?**

No. If a super class method is throwing an unchecked exception, then it can be overridden in the sub class with same exception or any other unchecked exceptions but can not be overridden with checked exceptions.

**25) What are chained exceptions in java?**

Click [here](https://javaconceptoftheday.com/chained-exceptions-java/) to see about chained exceptions in java.

**26) Which class is the super class for all types of errors and exceptions in java?**

java.lang.Throwable is the super class for all types of errors and exceptions in java.

**27) What are the legal combinations of try, catch and finally blocks?**

1)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8 | try  {      //try block  }  catch(Exception ex)  {      //catch block  } |

2)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8 | try  {      //try block  }  finally  {      //finally block  } |

3)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12 | try  {      //try block  }  catch(Exception ex)  {      //catch block  }  finally  {      //finally block  } |

**28) What is the use of printStackTrace() method?**

printStackTrace() method is used to print the detailed information about the exception occurred.

**29) Give some examples to checked exceptions?**

ClassNotFoundException, SQLException, IOException

**30) Give some examples to unchecked exceptions?**

NullPointerException, ArrayIndexOutOfBoundsException, NumberFormatException

**Also Read :**

* [Java Strings Interview Questions](https://javaconceptoftheday.com/java-string-interview-questions-and-answers/)
* [Java Array Interview Questions](https://javaconceptoftheday.com/java-array-interview-questions-and-answers/)
* [Java Inheritance Interview Questions](https://javaconceptoftheday.com/java-interview-questions-on-inheritance/)
* [Interview Questions On main()](https://javaconceptoftheday.com/java-interview-questions-on-main-method/)
* [Interview Questions On final keyword](https://javaconceptoftheday.com/java-interview-questions-on-final-keyword/)
* [85+ Java Interview Programs](https://javaconceptoftheday.com/java-interview-programs-with-solutions/)
* [15 Simple but confusing interview questions](https://javaconceptoftheday.com/15-simple-but-confusing-java-interview-questions/)
* [40+ Core Java Coding Q&A](https://javaconceptoftheday.com/core-java-coding-questions-which-will-test-your-java-basic-skills/)
* [25+ simple basic Java interview questions](https://javaconceptoftheday.com/top-25-simple-basic-java-interview-questions-for-freshers/)

Categories [Exception Handling](https://javaconceptoftheday.com/category/exception-handling/), [Java Interview Questions](https://javaconceptoftheday.com/category/java-interview-questions/)

Previous:[String To Integer And Integer To String Conversion In Java](https://javaconceptoftheday.com/string-to-integer-integer-to-string-conversion-in-java/)

Next:[50 Java Keywords With Examples](https://javaconceptoftheday.com/java-keywords-with-examples/)

**7 Comments**

1. **sangam**  
   July 7, 2017 (12:35 pm) [#](https://javaconceptoftheday.com/java-exception-handling-interview-questions-and-answers/#comment-2422)

What design pattern is used to implement exception handling features in most languages?  
plz reply..

[Reply](https://javaconceptoftheday.com/java-exception-handling-interview-questions-and-answers/#comment-2422)

* + **dipesh**  
    October 16, 2019 (11:57 am) [#](https://javaconceptoftheday.com/java-exception-handling-interview-questions-and-answers/#comment-4652)

it is chain of responsibility design pattern

[Reply](https://javaconceptoftheday.com/java-exception-handling-interview-questions-and-answers/#comment-4652)

1. **Hardik**  
   September 10, 2017 (7:36 pm) [#](https://javaconceptoftheday.com/java-exception-handling-interview-questions-and-answers/#comment-2695)

Chain Of Responsibility Pattern

[Reply](https://javaconceptoftheday.com/java-exception-handling-interview-questions-and-answers/#comment-2695)

1. **Vivek**  
   September 27, 2017 (1:13 pm) [#](https://javaconceptoftheday.com/java-exception-handling-interview-questions-and-answers/#comment-2763)

5) Can we write only try block without catch and finally blocks?  
Yes, From java 7 onwards, if a class implements AutoCloseable interface, then it could be used as an argument in the try section and then it is not compulsory to have catch or finally block.  
Please refer to the below example. Here Scanner class implements AutoCloseable interface.

public class Test{  
public static void main(String[] args) {  
//After the completion of try block, the scanner object would be auto closed.  
try(Scanner sc = new Scanner(System.in)){  
System.out.println(sc.hashCode());  
}  
}  
}

[Reply](https://javaconceptoftheday.com/java-exception-handling-interview-questions-and-answers/#comment-2763)

1. **Pragyesh Srivastav**  
   October 27, 2018 (4:21 pm) [#](https://javaconceptoftheday.com/java-exception-handling-interview-questions-and-answers/#comment-3927)

5) Yes , In Java 7 with the introduction of try with resources concept , we are able to write try block without catch and finally blocks.  
The only constraint is the resources which we are declaring as try block parameter must implements Autocloseable interface.  
A resource is said to be Autocloseable if and only if corresponding class implements java.lang.Autocloseable  
interface.

[Reply](https://javaconceptoftheday.com/java-exception-handling-interview-questions-and-answers/#comment-3927)

1. **Chetan**  
   July 14, 2019 (1:39 pm) [#](https://javaconceptoftheday.com/java-exception-handling-interview-questions-and-answers/#comment-4472)

consider following example snipped  
try{  
int a = 10/0; // statement 1  
}catch(ArithmeticException ae){  
// handle code  
}  
My question is – exception arise at statement 1 but how java internally identify that exception is ArithmeticException or else ?

[Reply](https://javaconceptoftheday.com/java-exception-handling-interview-questions-and-answers/#comment-4472)

* + **Marjan**  
    September 23, 2019 (11:27 pm) [#](https://javaconceptoftheday.com/java-exception-handling-interview-questions-and-answers/#comment-4615)

When an exception occurs inside a Java method, the method creates an Exception object and passes the Exception object to the JVM (in Java term, the method ” throw ” an Exception ). The Exception object contains the type of the exception, and the state of the program when the exception occurs.

[Reply](https://javaconceptoftheday.com/java-exception-handling-interview-questions-and-answers/#comment-4615)

**Leave a Reply**

Top of Form

Your email address will not be published. Required fields are marked \*

Comment

Name \*

Email \*

Website

 Notify me of follow-up comments by email.

 Notify me of new posts by email.



Bottom of Form

**Subscribe To Our Latest Updates**

Top of Form

Join 12,119 other subscribers

Email Address

Subscribe

Bottom of Form

**Java 8 Tutorials**

[Java 8 Lambda Expressions](https://javaconceptoftheday.com/java-8-lambda-expressions/)

[Java 8 Functional Interfaces](https://javaconceptoftheday.com/java-8-functional-interfaces/)

[Java 8 Method References](https://javaconceptoftheday.com/java-8-method-references/)

[Java 8 Interface Changes](https://javaconceptoftheday.com/java-8-interface-changes-default-methods-and-static-methods/)

[Java 8 Streams](https://javaconceptoftheday.com/java-8-streams-beginners-guide/)

[Java 8 Collectors](https://javaconceptoftheday.com/java-8-collectors-tutorial/)

[Interface Vs Abstract Class After Java 8](https://javaconceptoftheday.com/interface-vs-abstract-class-after-java-8/)

[Collections Vs Streams](https://javaconceptoftheday.com/collections-and-streams-in-java/)

[Solving Real Time Queries Using Java 8 - Employee Management System](https://javaconceptoftheday.com/solving-real-time-queries-using-java-8-features-employee-management-system/)

**Java Interview Questions**

[Exception Handling Interview Questions](https://javaconceptoftheday.com/java-exception-handling-interview-questions-and-answers/)

[Strings Interview Questions](https://javaconceptoftheday.com/java-string-interview-questions-and-answers/)

[Array Interview Questions](https://javaconceptoftheday.com/java-array-interview-questions-and-answers/)

[25 Fresher's Basic Java Interview Questions](https://javaconceptoftheday.com/top-25-simple-basic-java-interview-questions-for-freshers/)

[15 Simple But Confusing Java Interview Questions](https://javaconceptoftheday.com/15-simple-but-confusing-java-interview-questions/)

[Inheritance Interview Questions](https://javaconceptoftheday.com/java-interview-questions-on-inheritance/)

[Tricky Java Interview Questions](https://javaconceptoftheday.com/tricky-core-java-interview-coding-questions/)

**Recent Posts :**

* [Java Servlet Life Cycle](https://javaconceptoftheday.com/java-servlet-life-cycle/)
* [Magic Number Program In Java](https://javaconceptoftheday.com/magic-number-program-in-java/)
* [Servlet Vs GenericServlet Vs HttpServlet In Java Servlets](https://javaconceptoftheday.com/servlet-vs-genericservlet-vs-httpservlet-in-java-servlets/)
* [Java Servlet API](https://javaconceptoftheday.com/java-servlet-api/)
* [Java Servlets Architecture](https://javaconceptoftheday.com/java-servlets-architecture/)
* [Differences Between Iterator Vs Spliterator In Java 8](https://javaconceptoftheday.com/differences-between-iterator-vs-spliterator-in-java-8/)

**Popular Posts :**

* [85+ Popular Java Interview Programs With Solutions](https://javaconceptoftheday.com/java-interview-programs-with-solutions/)
* [30 Java Exception Handling Interview Questions And Answers](https://javaconceptoftheday.com/java-exception-handling-interview-questions-and-answers/)
* [40 Java Inheritance Practice Coding Questions](https://javaconceptoftheday.com/java-inheritance-practice-coding-questions/)
* [How To Count Occurrences Of Each Character In String In Java?](https://javaconceptoftheday.com/how-to-count-occurrences-of-each-character-in-string-in-java/)
* [Solving Real Time Queries Using Java 8 Features -Employee Management System](https://javaconceptoftheday.com/solving-real-time-queries-using-java-8-features-employee-management-system/)
* [Java String Interview Questions And Answers](https://javaconceptoftheday.com/java-string-interview-questions-and-answers/)

**Content Menu**

[**Getting Started**](https://javaconceptoftheday.com/java-exception-handling-interview-questions-and-answers/#GettingStarted)

[**Classes & Objects**](https://javaconceptoftheday.com/java-exception-handling-interview-questions-and-answers/#ClassesObjects)

[**Inheritance**](https://javaconceptoftheday.com/java-exception-handling-interview-questions-and-answers/#Inheritance)

[**Polymorphism**](https://javaconceptoftheday.com/java-exception-handling-interview-questions-and-answers/#Polymorphism)

[**Abstraction**](https://javaconceptoftheday.com/java-exception-handling-interview-questions-and-answers/#Abstraction)

[**Packages**](https://javaconceptoftheday.com/java-exception-handling-interview-questions-and-answers/#packages)

[**Type Casting**](https://javaconceptoftheday.com/java-exception-handling-interview-questions-and-answers/#TypeCasting)

[**Nested Classes**](https://javaconceptoftheday.com/java-exception-handling-interview-questions-and-answers/#NestedClasses)

[**Java Enums**](https://javaconceptoftheday.com/java-exception-handling-interview-questions-and-answers/#JavaEnums)

[**Wrapper Classes**](https://javaconceptoftheday.com/java-exception-handling-interview-questions-and-answers/#WrapperClasses)

[**Arrays**](https://javaconceptoftheday.com/java-exception-handling-interview-questions-and-answers/#Arrays)

[**Strings**](https://javaconceptoftheday.com/java-exception-handling-interview-questions-and-answers/#Strings)

[**Exception Handling**](https://javaconceptoftheday.com/java-exception-handling-interview-questions-and-answers/#ExceptionHandling)

[**Threads**](https://javaconceptoftheday.com/java-exception-handling-interview-questions-and-answers/#Threads)

[**File I/O**](https://javaconceptoftheday.com/java-exception-handling-interview-questions-and-answers/#FileIO)

[**Generics**](https://javaconceptoftheday.com/java-exception-handling-interview-questions-and-answers/#Generics)

[**Java Collections**](https://javaconceptoftheday.com/java-exception-handling-interview-questions-and-answers/#JavaCollections)

[**JDBC**](https://javaconceptoftheday.com/java-exception-handling-interview-questions-and-answers/#JDBC)

[**Interview Programs**](https://javaconceptoftheday.com/java-exception-handling-interview-questions-and-answers/#InterviewPrograms)

[**Interview Q&A**](https://javaconceptoftheday.com/java-exception-handling-interview-questions-and-answers/#InterviewQA)

[**x Vs y**](https://javaconceptoftheday.com/java-exception-handling-interview-questions-and-answers/#XvsY)

[**Java 8**](https://javaconceptoftheday.com/java-exception-handling-interview-questions-and-answers/#Java8)

[Java Concept Of The Day](https://javaconceptoftheday.com/) © 2021 [About Us](https://javaconceptoftheday.com/about-us/) [Privacy Policy](https://javaconceptoftheday.com/privacy-policy/) [Contact Us](https://javaconceptoftheday.com/contact-us/)

* [**Start Here**](https://www.baeldung.com/start-here)
* [**Courses ▼**](javascript:void(0))
* [**Guides ▼**](javascript:void(0))
* [**About ▼**](javascript:void(0))

Java Exceptions Interview Questions (+ Answers)

Last modified: April 24, 2020

by [baeldung](https://www.baeldung.com/author/baeldung/)

* [**Java**](https://www.baeldung.com/category/java/)**+**
* [**Exception**](https://www.baeldung.com/tag/exception/)
* [**Interview**](https://www.baeldung.com/tag/interview/)

**Get started with Spring 5 and Spring Boot 2, through the *Learn Spring* course:**

[**>> CHECK OUT THE COURSE**](https://www.baeldung.com/ls-course-start)

[This article is part of a series:](javascript:void(0);)

**1. Overview**

Exceptions are an essential topic that every Java developer should be familiar with. This article provides answers to some of the questions that might pop up during an interview.

**2. Questions**

**Q1. What Is an Exception?**

An exception is an abnormal event that occurs during the execution of a program and disrupts the normal flow of the program's instructions.

**Q2. What Is the Purpose of the Throw and Throws Keywords?**

The *throws* keyword is used to specify that a method may raise an exception during its execution. It enforces explicit exception handling when calling a method:

**public** **void** **simpleMethod**() **throws** Exception {

// ...

}

The *throw* keyword allows us to throw an exception object to interrupt the normal flow of the program. This is most commonly used when a program fails to satisfy a given condition:

**if** (task.isTooComplicated()) {

**throw** **new** TooComplicatedException("The task is too complicated");

}

**Q3. How Can You Handle an Exception?**

By using a *try-catch-finally* statement:

**try** {

// ...

} **catch** (ExceptionType1 ex) {

// ...

} **catch** (ExceptionType2 ex) {

// ...

} **finally** {

// ...

}

The block of code in which an exception may occur is enclosed in a *try* block. This block is also called “protected” or “guarded” code.

If an exception occurs, the *catch* block that matches the exception being thrown is executed, if not, all *catch* blocks are ignored.

The *finally* block is always executed after the *try* block exits, whether an exception was thrown or not inside it.

**Q4. How Can You Catch Multiple Exceptions?**

There are three ways of handling multiple exceptions in a block of code.

The first is to use a *catch* block that can handle all exception types being thrown:

**try** {

// ...

} **catch** (Exception ex) {

// ...

}

You should keep in mind that the recommended practice is to use exception handlers that are as accurate as possible.

Exception handlers that are too broad can make your code more error-prone, catch exceptions that weren't anticipated, and cause unexpected behavior in your program.

The second way is implementing multiple catch blocks:

**try** {

// ...

} **catch** (FileNotFoundException ex) {

// ...

} **catch** (EOFException ex) {

// ...

}

Note that, if the exceptions have an inheritance relationship; the child type must come first and the parent type later. If we fail to do this, it will result in a compilation error.

The third is to use a multi-catch block:

**try** {

// ...

} **catch** (FileNotFoundException | EOFException ex) {

// ...

}

This feature, first introduced in Java 7; reduces code duplication and makes it easier to maintain.

**Q5. What Is the Difference Between a Checked and an Unchecked Exception?**

A checked exception must be handled within a *try-catch* block or declared in a *throws* clause; whereas an unchecked exception is not required to be handled nor declared.

Checked and unchecked exceptions are also known as compile-time and runtime exceptions respectively.

All exceptions are checked exceptions, except those indicated by *Error*, *RuntimeException*, and their subclasses.

**Q6. What Is the Difference Between an Exception and Error?**

An exception is an event that represents a condition from which is possible to recover, whereas error represents an external situation usually impossible to recover from.

All errors thrown by the JVM are instances of *Error* or one of its subclasses, the more common ones include but are not limited to:

* *OutOfMemoryError* – thrown when the JVM cannot allocate more objects because it is out memory, and the garbage collector was unable to make more available
* *StackOverflowError* – occurs when the stack space for a thread has run out, typically because an application recurses too deeply
* *ExceptionInInitializerError* – signals that an unexpected exception occurred during the evaluation of a static initializer
* *NoClassDefFoundError* – is thrown when the classloader tries to load the definition of a class and couldn't find it, usually because the required *class* files were not found in the classpath
* *UnsupportedClassVersionError* – occurs when the JVM attempts to read a *class* file and determines that the version in the file is not supported, normally because the file was generated with a newer version of Java

Although an error can be handled with a *try* statement, this is not a recommended practice since there is no guarantee that the program will be able to do anything reliably after the error was thrown.

**Q7. What Exception Will Be Thrown Executing the Following Code Block?**

Integer[][] ints = { { 1, 2, 3 }, { **null** }, { 7, 8, 9 } };

System.out.println("value = " + ints[1][1].intValue());

It throws an *ArrayIndexOutOfBoundsException*since we're trying to access a position greater than the length of the array.

**Q8. What Is Exception Chaining?**

Occurs when an exception is thrown in response to another exception. This allows us to discover the complete history of our raised problem:

**try** {

task.readConfigFile();

} **catch** (FileNotFoundException ex) {

**throw** **new** TaskException("Could not perform task", ex);

}

**Q9. What Is a Stacktrace and How Does It Relate to an Exception?**

A stack trace provides the names of the classes and methods that were called, from the start of the application to the point an exception occurred.

It's a very useful debugging tool since it enables us to determine exactly where the exception was thrown in the application and the original causes that led to it.

**Q10. Why Would You Want to Subclass an Exception?**

If the exception type isn't represented by those that already exist in the Java platform, or if you need to provide more information to client code to treat it in a more precise manner, then you should create a custom exception.

Deciding whether a custom exception should be checked or unchecked depends entirely on the business case. However, as a rule of thumb; if the code using your exception can be expected to recover from it, then create a checked exception otherwise make it unchecked.

Also, you should inherit from the most specific *Exception* subclass that closely relates to the one you want to throw. If there is no such class, then choose *Exception*as the parent.

**Q11. What Are Some Advantages of Exceptions?**

Traditional error detection and handling techniques often lead to spaghetti code hard to maintain and difficult to read. However, exceptions enable us to separate the core logic of our application from the details of what to do when something unexpected happens.

Also, since the JVM searches backward through the call stack to find any methods interested in handling a particular exception; we gain the ability to propagate an error up in the call stack without writing additional code.

Also, because all exceptions thrown in a program are objects, they can be grouped or categorized based on its class hierarchy. This allows us to catch a group of exceptions in a single exception handler by specifying the exception's superclass in the *catch* block.

**Q12. Can You Throw Any Exception Inside a Lambda Expression's Body?**

When using a standard functional interface already provided by Java, you can only throw unchecked exceptions because standard functional interfaces do not have a “throws” clause in method signatures:

List<Integer> integers = Arrays.asList(3, 9, 7, 0, 10, 20);

integers.forEach(i -> {

**if** (i == 0) {

**throw** **new** IllegalArgumentException("Zero not allowed");

}

System.out.println(Math.PI / i);

});

However, if you are using a custom functional interface, throwing checked exceptions is possible:

@FunctionalInterface

**public** **static** **interface** **CheckedFunction**<**T**> {

**void** **apply**(T t) **throws** Exception;

}

**public** **void** **processTasks**(

List<Task> taks, CheckedFunction<Task> checkedFunction) {

**for** (Task task : taks) {

**try** {

checkedFunction.apply(task);

} **catch** (Exception e) {

// ...

}

}

}

processTasks(taskList, t -> {

// ...

**throw** **new** Exception("Something happened");

});

**Q13. What Are the Rules We Need to Follow When Overriding a Method That Throws an Exception?**

Several rules dictate how exceptions must be declared in the context of inheritance.

When the parent class method doesn't throw any exceptions, the child class method can't throw any checked exception, but it may throw any unchecked.

Here's an example code to demonstrate this:

**class** **Parent** {

**void** **doSomething**() {

// ...

}

}

**class** **Child** **extends** **Parent** {

**void** **doSomething**() **throws** IllegalArgumentException {

// ...

}

}

The next example will fail to compile since the overriding method throws a checked exception not declared in the overridden method:

**class** **Parent** {

**void** **doSomething**() {

// ...

}

}

**class** **Child** **extends** **Parent** {

**void** **doSomething**() **throws** IOException {

// Compilation error

}

}

When the parent class method throws one or more checked exceptions, the child class method can throw any unchecked exception; all, none or a subset of the declared checked exceptions, and even a greater number of these as long as they have the same scope or narrower.

Here's an example code that successfully follows the previous rule:

**class** **Parent** {

**void** **doSomething**() **throws** IOException, ParseException {

// ...

}

**void** **doSomethingElse**() **throws** IOException {

// ...

}

}

**class** **Child** **extends** **Parent** {

**void** **doSomething**() **throws** IOException {

// ...

}

**void** **doSomethingElse**() **throws** FileNotFoundException, EOFException {

// ...

}

}

Note that both methods respect the rule. The first throws fewer exceptions than the overridden method, and the second, even though it throws more; they're narrower in scope.

However, if we try to throw a checked exception that the parent class method doesn't declare or we throw one with a broader scope; we'll get a compilation error:

**class** **Parent** {

**void** **doSomething**() **throws** FileNotFoundException {

// ...

}

}

**class** **Child** **extends** **Parent** {

**void** **doSomething**() **throws** IOException {

// Compilation error

}

}

When the parent class method has a throws clause with an unchecked exception, the child class method can throw none or any number of unchecked exceptions, even though they are not related.

Here's an example that honors the rule:

**class** **Parent** {

**void** **doSomething**() **throws** IllegalArgumentException {

// ...

}

}

**class** **Child** **extends** **Parent** {

**void** **doSomething**()

**throws** ArithmeticException, BufferOverflowException {

// ...

}

}

**Q14. Will the Following Code Compile?**

**void** **doSomething**() {

// ...

**throw** **new** RuntimeException(**new** Exception("Chained Exception"));

}

Yes. When chaining exceptions, the compiler only cares about the first one in the chain and, because it detects an unchecked exception, we don't need to add a throws clause.

**Q15. Is There Any Way of Throwing a Checked Exception from a Method That Does Not Have a Throws Clause?**

Yes. We can take advantage of the type erasure performed by the compiler and make it think we are throwing an unchecked exception, when, in fact; we're throwing a checked exception:

**public** <T extends Throwable> T **sneakyThrow**(Throwable ex) **throws** T {

**throw** (T) ex;

}

**public** **void** **methodWithoutThrows**() {

**this**.<RuntimeException>sneakyThrow(**new** Exception("Checked Exception"));

}

**3. Conclusion**

In this article, we’ve explored some of the questions that are likely to appear in technical interviews for Java developers, regarding exceptions. This is not an exhaustive list, and it should be treated only as the start of further research.

We, at Baeldung, wish you success in any upcoming interviews.

**Next »**

[Java Annotations Interview Questions (+ Answers)](https://www.baeldung.com/java-annotations-interview-questions)

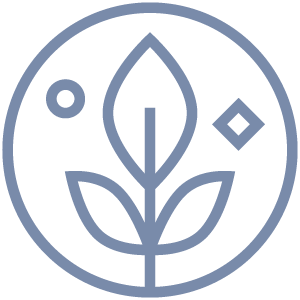
**« Previous**

[Java Flow Control Interview Questions (+ Answers)](https://www.baeldung.com/java-flow-control-interview-questions)

**Get started with Spring 5 and Spring Boot 2, through the *Learn Spring* course:**

[**>> CHECK OUT THE COURSE**](https://www.baeldung.com/ls-course-end)





Learning to "Build your API  
**with Spring**"?

Top of Form

>> Get the eBook

Bottom of Form

Comments are closed on this article!



Top of Form

Bottom of Form

**CATEGORIES**

* [SPRING](https://www.baeldung.com/category/spring/)
* [REST](https://www.baeldung.com/category/rest/)
* [JAVA](https://www.baeldung.com/category/java/)
* [SECURITY](https://www.baeldung.com/category/security-2/)
* [PERSISTENCE](https://www.baeldung.com/category/persistence/)
* [JACKSON](https://www.baeldung.com/category/json/jackson/)
* [HTTP CLIENT-SIDE](https://www.baeldung.com/category/http/)

**SERIES**

* [JAVA “BACK TO BASICS” TUTORIAL](https://www.baeldung.com/java-tutorial)
* [JACKSON JSON TUTORIAL](https://www.baeldung.com/jackson)
* [HTTPCLIENT 4 TUTORIAL](https://www.baeldung.com/httpclient-guide)
* [REST WITH SPRING TUTORIAL](https://www.baeldung.com/rest-with-spring-series)
* [SPRING PERSISTENCE TUTORIAL](https://www.baeldung.com/persistence-with-spring-series)
* [SECURITY WITH SPRING](https://www.baeldung.com/security-spring)
* [SPRING REACTIVE TUTORIALS](https://www.baeldung.com/spring-reactive-guide)

**ABOUT**

* [ABOUT BAELDUNG](https://www.baeldung.com/about)
* [THE COURSES](https://courses.baeldung.com/)
* [JOBS](https://www.baeldung.com/tag/active-job/)
* [THE FULL ARCHIVE](https://www.baeldung.com/full_archive)
* [WRITE FOR BAELDUNG](https://www.baeldung.com/contribution-guidelines)
* [EDITORS](https://www.baeldung.com/editors)
* [OUR PARTNERS](https://www.baeldung.com/partners)
* [ADVERTISE ON BAELDUNG](https://www.baeldung.com/advertise)
* [TERMS OF SERVICE](https://www.baeldung.com/terms-of-service)

* [PRIVACY POLICY](https://www.baeldung.com/privacy-policy)

* [COMPANY INFO](https://www.baeldung.com/baeldung-company-info)

* [CONTACT](https://www.baeldung.com/contact)

[upGrad blog](https://www.upgrad.com/blog/)

* [**PROGRAMS**](https://www.upgrad.com/blog/exception-handling-interview-questions-answers/)
* [**START LEARNING**](https://www.upgrad.com/?utm_source=BLOG&utm_medium=TOP_MENUBAR&utm_campaign=ALL_ALL_ALL_BLOG_TOP_MENUBAR)
* [**GLOBAL MBA**](https://www.upgrad.com/blog/)
* [**MBA PROGRAM**](https://www.upgrad.com/mba-courses/?utm_source=BLOG&utm_medium=MAIN_MENUBAR&utm_campaign=ALL_ALL_ALL_BLOG_MAIN_MENUBAR)
* [**DOWNLOAD APP**](https://upgrad.onelink.me/9IEK/7d1d4d0b)

Top 30 Exception Handling Interview Questions and Answers [For Freshers & Experienced]

**[](https://www.upgrad.com/blog/author/rohan/) by**[**Rohan Vats**](https://www.upgrad.com/blog/author/rohan/)

SEP 29, 2020

[Home](https://www.upgrad.com/blog) > [Full Stack Development](https://www.upgrad.com/blog/full-stack-development/) > Top 30 Exception Handling Interview Questions and Answers [For Freshers & Experienced]

Exception handling is a concept that is implemented in algorithms to handle possible runtime errors, which may disrupt the normal flow of a program. Some of the errors which can be handled using this concept are:

* ClassNotFoundException
* IOException
* SQLException
* RemoteException
* RuntimeException:
  + ArithmeticException
  + NullPointerException
  + NumberFormatException
  + IndexOutOfBoundsException
    - ArrayIndexOutOfBoundsException
    - StringIndexOutOfBoundsException

The merit of this implementation is to prevent a crash of the program if there is an exception while executing the program. Without Exception Handling, the program will throw an error when it encounters an exception, and the rest of the program will not be executed. However, implementing this concept will give a workaround in which the rest of the program is executed if they are independent with respect to the exception incurred.

**Read:**[Must Read 30 Selenium Interview Questions & Answers: Ultimate Guide](https://www.upgrad.com/blog/selenium-interview-questions-answers/)

**Exception Handling Interview Questions and Answers**

**1. What do you mean by an exception?**

It is an abnormal condition that is sometimes encountered when a program is executed. It disrupts the normal flow of the program. It is necessary to handle this exception; otherwise,it can cause the program to be terminated abruptly.

**2. Explain how exceptions can be handled in Java. What is the exception handling mechanism behind the process?**

[](https://www.upgrad.com/full-stack-developer-course-pgd-iiitb/?utm_source=BLOG&utm_medium=BODY&utm_campaign=TV_FSD_PGD_BLOG_BODY_POPULAR)

There are three parts to the exception handling mechanism. These are called:

* Try block: The section of the code which is first attempted to be executed and monitored for any exception that might occur.
* Catch block: If any exception is thrown by the ‘try’ block, it is caught by this code section.
* Finally block: Code under this section is always executed irrespective of exceptions caught in ‘try’ block, if any. Even if there is no exception, the code under this block will be executed.

**3. Is it possible to keep other statements in between ‘try’, ‘catch’, and ‘finally’ blocks?**

It is not recommended to include any statements between the sections of ‘try’, ‘catch’, and ‘finally’ blocks, since they form one whole unit of the exception handling mechanism.

try

{

    //Code which is monitored for exceptions.

}

//You can’t keep statements here

catch(Exception ex)

{

    //Catch the exceptions thrown by try block, if any.

}

//You can’t keep statements here

finally

{

    //This block is always executed irrespective of exceptions.

}

**4. Will it be possible to only include a ‘try’ block without the ‘catch’ and ‘finally’ blocks?**

This would give a compilation error. It is necessary for the ‘try’ block to be followed with either a ‘catch’ block or a ‘finally’ block, if not both. Either one of ‘catch’ or ‘finally’ blocks is needed so that the flow of exception handling is undisrupted.

**5. Will it be possible to keep the statements after the ‘finally’ block if the control is returning from the finally block itself?**

This will result in an unreachable catch block error. This is because the control will be returning from the ‘finally’ block itself. The compiler will fail to execute the code after the line with the exception. That is why the execution will show an unreachable code error.

[Dreaming to Study Abroad? Here is the Right program for you](https://www.upgrad.com/machine-learning-ai-pgd-iiitb/?utm_source=BLOG&utm_medium=TEXTCTA&utm_campaign=DV_ML_PGD_BLOG_TEXTCTA_89307)

**6. Explain an unreachable catch block error.**

In the case of multiple catch blocks, the order in which catch blocks are placed is from the most specific to the most general ones. That is, the sub classes of an exception should come first, and then the super classes will follow. In case that the super classes are kept first, followed by the sub classes after it, the compiler will show an unreachable catch block error.

public class ExceptionHandling

{

    public static void main(String[] args)

    {

        try

        {

            int i = Integer.parseInt(“test”);

//This statement will throw a NumberFormatException //because the given input is string, while the //specified format is integer.

        }

        catch(Exception ex)

        {

System.out.println(“This block handles all exception types”);

//All kinds of exceptions can be handled in this //block since it is a super class of exceptions.

        }

        catch(NumberFormatException ex)

        {

            //This will give compile time error

            //This block will become unreachable as the

//exception would be already caught by the above //catch block

        }

    }

}

**7. Consider three statements in a ‘try’ block: statement1, statement2, and statement3. It is followed by a ‘catch’ block to catch the exceptions that occurred during the execution of the ‘try’ block. Assume that the exception is thrown at statement2. Do you think the statement3 will be executed?**

Statement3 will not be executed. If an exception is thrown by the ‘try’ block at any point, the remaining code after the exception will not be executed. Instead, the flow control will directly come to the ‘catch’ block.

**8. Differentiate error and exception in Java.**

The key difference between error and exception is that while the error is caused by the environment in which the JVM(Java Virtual Machine) is running, exceptions are caused by the program itself. For example, OutOfMemory is an error that occurs when the JVM exhausts its memory.

But, NullPointerException is an exception that is encountered when the program tries to access a null object. Recovering from an error is not possible. Hence, the only solution to an error is to terminate the execution. However, it is possible to workaround exceptions using try and catch blocks or by throwing exceptions back to the caller function.

**Must Read:**[Java Interview Questions & Answers](https://www.upgrad.com/blog/java-interview-questions-answers/)

**9. What are the types of exceptions? Explain them.**

**There are two types of exceptions:**

**Checked Exceptions**

The type of exceptions that are known and recognized by the compiler. These exceptions can be checked in compile time only. Therefore, they are also called compile time exceptions. These can be handled by either using try and catch blocks or by using a throw clause. If these exceptions are not handled appropriately, they will produce compile time errors. Examples include the subclasses of java.lang.Exception except for the RunTimeException.

**Unchecked Exceptions**

The type of exceptions that are not recognized by the compiler. They occur at run time only. Hence, they are also called run time exceptions. They are not checked at compile time. Hence, even after a successful compilation, they can cause the program to terminate prematurely if not handled appropriately. Examples include the subclasses of java.lang.RunTimeException and java.lang.Error.

**10. What is the hierarchy of exceptions in Java?**

The java.lang.Throwable is a super class of all errors and exceptions in Java. This class extends the java.lang.Object class. The argument of catch block should be its type or its sub class type only. The Throwable class includes two sub classes:

1. **java.lang.Error :** This is a super class for all error types in Java. Common errors included under this are –
   1. java.lang.VirtualMachineError: Under this –
      1. StackOverFlowError
      2. OutOfMemoryError
   2. java.lang.AssertionError
   3. java.lang.LinkageError: Under this –
      1. NoClassDefFoundError
      2. IncompatibleClassChangeError
2. **java.lang.Exception:** This is a super class of all exception types in Java. Common exceptions under this are –
   1. RunTimeException
      1. ArithmeticException
      2. NumberFormatException
      3. NullPointerException
      4. ArrayIndexOutOfBoundsException
      5. ClassCastException
   2. java.lang.InterruptedException
   3. java.lang.IOException
   4. java.lang.SQLException
   5. java.lang.ParseException

**11. What are runtime exceptions in Java? Give a few examples.**

The exceptions that occur at run time are called run time exceptions. The compiler cannot recognise these exceptions, like unchecked exceptions. It includes all sub classes of java.lang.RunTimeException and java.lang.Error. Examples include, NumberFormatException, NullPointerException, ClassCastException, ArrayIndexOutOfBoundException, StackOverflowError etc.

**12. Define OutOfMemoryError in Java.**

It is the sub class of java.lang.Error which is encountered when the JVM runs out of memory.

**13. Differentiate between NoClassDefFoundError and ClassNotFoundException in Java.**

Both NoClassDefFoundError and ClassNotFoundException occur when a particular class is not found in run time. However, they occur under different scenarios. NoClassDefFoundError is when an error occurs because a particular class was present at compile time but it was missing at run time. ClassNotFoundException occurs when an exception is encountered for an application trying to load a class at run time which is not updated in the classpath.

**14. Does the ‘finally’ block get executed if either of ‘try’ or ‘catch’ blocks return the control?**

The ‘finally’ block is always executed irrespective of whether try or catch blocks are returning the control or not.

**15. Is it possible to throw an exception manually? If yes, explain how.**

It is possible to throw an exception manually. It is done using the ‘throw’ keyword. The syntax for throwing an exception manually is

*throw InstanceOfThrowableType;*

Here is an example of using the ‘throw’ keyword to throw an exception manually.

try

{

    NumberFormatException ex = new NumberFormatException(); //Here we create an object for NumberFormatException explicitly

    throw ex; //throwing NumberFormatException object explicitly using throw keyword

}

catch(NumberFormatException ex)

{

    System.out.println(“In this block, the explicitly thrown NumberFormatException object can be caught.”);

}

**Read:**[Top 35 Spring Interview Questions & Answers: Ultimate Guide](https://www.upgrad.com/blog/spring-interview-questions-answers/)

**16. What do you mean by rethrowing an exception in Java?**

The exceptions which are raised in the ‘try’ block are handled in the ‘catch’ block. If the ‘catch’ block is unable to handle that exception, it is possible that it can rethrow the same exception using the ‘throw’ keyword. This mechanism is called rethrowing an exception. The implementation is as follows:

try

{

    String s = null;

    System.out.println(s.length()); //This statement throws a NullPointerException

}

catch(NullPointerException ex)

{

    System.out.println(“Here the NullPointerException is caught”);

    throw ex; //Rethrowing the NullPointerException

}

**17. Why do you use the ‘throws’ keyword in Java?**

If it is possible for a method to throw an exception if it could not be handled, it should specify that exception using the ‘throws’ keyword. It will be helpful to the caller functions of that method in handling that exception. The syntax for using the ‘throws’ keyword is,

return\_type method\_name(parameter\_list) throws exception\_list

{

     //code

}

Here, exception\_list is the list of exceptions which may be thrown by the method. These exceptions should be separated by commas. An example of the code :

public class ExceptionHandling

{

    public static void main(String[] args)

    {

        try

        {

            methodWithThrows();

        }

        catch(NullPointerException ex)

        {

            System.out.println(&quot;NullPointerException thrown by methodWithThrows() method will be caught here&quot;);

        }

    }

    static void methodWithThrows() throws NullPointerException

    {

        String s = null;

        System.out.println(s.length()); //This statement throws NullPointerException

    }

}

**18. It is often recommended to keep clean up operations like closing the DB resources inside the ‘finally’ block. Why is it necessary?**

The ‘finally’ block is always executed irrespective of the fact if exceptions are raised in the ‘try’ block or if the raised exceptions are caught in the ‘catch’ block or not. Keeping the clean up operations in ‘finally’ block ensures the operation of these operations in any case, and will not be affected by exceptions, which may or may not rise.

**19. How would you differentiate between final, finally and finalize in Java?**

First, ‘final’ is a keyword that can be used to make a variable or a method or a class as unchangeable. To put it simply, if a variable is declared as final, once it is initialized, its value can not be altered. If a method is declared as final, it cannot be overridden or modified in the sub class. If a class is declared as final, it cannot be extended into further classes.

Second, ‘finally’ is a block which is used in exception handling along with the ‘try’ and ‘catch’ blocks. This block is always executed irrespective of a raised exception or if the raised exception is handled. Usually, this block is used to perform clean up operations to close the resources like database connection, I/O resources, etc.

Third, the finalize() method is a protected method. It belongs to java.lang.Object class. Every class created in Java inherits this method. The garbage collector thread calls this method before an object is removed from the memory. Before an object is removed from the memory, this method is used to perform some of the clean-up operations.

protected void finalize() throws Throwable

{

    //Clean up operations

}

**20. What are customized exceptions in java?**

Exception classes can be thrown in Java as per the requirements of the program flow. These exceptions are called user-defined exceptions. They are also called customized exceptions. These exceptions must extend any one of the classes in the exceptions’ hierarchy.

**21. How would you explain a ClassCastException in Java?**

When the JVM is unable to cast an object of one type to another type, this exception is raised. It is a RunTimeException.

**22. Differentiate between throw, throws and throwable in Java.**

First, the keyword ‘throw’ is used to throw an exception manually in Java. Using this keyword, it is possible to throw an exception from any method or block. However, it is essential that the exception must be of type java.lang.Throwable class or it belongs to one of the sub classes of java.lang.Throwable class.

Second, the keyword ‘throws’ is used in the method signature in Java. If the method is capable of throwing exceptions, it is indicated by this method. The mentioned exceptions are handled by their respective caller functions. It is done either by using try and catch blocks or by using the throws keyword.

Third, the super class for all types of errors and exceptions in Java is called Throwable. It is a member of the java.lang package. The JVM or the throw statement raises only instances of this class or its subclasses. The catch block should contain only one argument and it should be of this type or its subclasses. In case customized exceptions are created, they should extend this class too.

**23. Explain the StackOverflowError in Java.**

This is an error that is thrown by the JVM when the stack overflows in runtime.

**24. Is it possible to override a super class method that throws an unchecked exception with checked exceptions in the sub class?**

It is not possible because if a super class method throws an unchecked exception, it will be overridden in the sub class with the same exception or any other unchecked exceptions. But, it can not be overridden with checked exceptions.

**25. Define chained exceptions in Java.**

In a program, one exception can throw many exceptions by inducing a domino effect. This causes a chain of exceptions. It is beneficial to know the location of the actual cause of the exception. This is possible with the chained exceptions feature in Java. This has been introduced since JDK 1.4. For implementation of chained exceptions in Java, two new constructors and two new methods are included in the Throwable class. These are,

**Constructors Of Throwable class:**

* 1. Throwable(Throwable cause): The cause is the exception that raises the current exception.
  2. Throwable(String msg, Throwable cause): The msg string is the exception message. The exception that raises the current exception is the cause here.

**Methods Of Throwable class:**

* 1. getCause() method : The real cause of a raised exception is returned by this method.
  2. initCause(Throwable cause) method : The cause of the calling exception is set by this method.

**26. Which class is defined as a super class for all types of errors and exceptions in Java?**

The super class for all types of errors and exceptions is java.lang.Throwable in Java.

**27. What can classify as a correct combination of try, catch and finally blocks?**

A combination of try and catch block.

try

{

    //try block

}

catch(Exception ex)

{

    //catch block

}

A combination of try and finally block.

try

{

    //try block

}

finally

{

    //finally block

}

A combination of all three: try, block, finally blocks.

try

{

    //try block

}

catch(Exception ex)

{

    //catch block

}

finally

{

    //finally block

}

**28. Why do you use printStackTrace() method?**

This method is used to print detailed information about the exception that occurred.

**29. What are some examples of checked exceptions?**

[](https://www.upgrad.com/full-stack-developer-course-pgd-iiitb/?utm_source=BLOG&utm_medium=BODY&utm_campaign=TV_FSD_PGD_BLOG_BODY_POPULAR)

Some examples of checked exceptions include ClassNotFoundException, SQLException, and IOException.

**30. What are some examples of unchecked exceptions?**

Some examples of unchecked exceptions include NullPointerException, ArrayIndexOutOfBoundsException and NumberFormatException.

**Also Read:**[Must Read 47 OOPS Interview Questions & Answers For Freshers & Experienced](https://www.upgrad.com/blog/oops-interview-questions-answers-for-freshers-experienced/)

**Wrapping up**

If you’re interested to learn more about big data, check out upGrad & IIIT-B’s [PG Diploma in Full-stack Software Development](https://www.upgrad.com/full-stack-developer-course-pgd-iiitb/?utm_source=BLOG&utm_medium=BODY&utm_campaign=TV_FSD_PGD_BLOG_BODY_89307) which is designed for working professionals and offers 500+ hours of rigorous training, 9+ projects, and assignments, IIIT-B Alumni status, practical hands-on capstone projects & job assistance with top firms.

[**EXCEPTION HANDLING**](https://www.upgrad.com/blog/tag/exception-handling/) [**EXCEPTION HANDLING INTERVIEW QUESTIONS**](https://www.upgrad.com/blog/tag/exception-handling-interview-questions/) [**INTERVIEW QUESTIONS AND ANSWERS**](https://www.upgrad.com/blog/tag/interview-questions-and-answers/)

Prepare for a Career of the Future

**UPGRAD AND IIIT-BANGALORE'S PG DIPLOMA IN FULL STACK SOFTWARE DEVELOPMENT**

[**ENROLL TODAY**](https://www.upgrad.com/full-stack-developer-course-pgd-iiitb/?utm_source=BLOG&utm_medium=FOOTER&utm_campaign=TV_FSD_PGD_BLOG_FOOTER_89307)

**Leave a comment**

Top of Form

Your email address will not be published.

Comment

Name

Email

Website



Bottom of Form

**Post navigation**

[**PREV**](https://www.upgrad.com/blog/google-cloud-vs-aws/)

[**NEXT**](https://www.upgrad.com/blog/ssis-interview-questions-and-answers/)

Our Trending Software Engineering Courses

* [PG Certification in Full Stack Development with Job Guarantee\* - Duration 5 Months](https://www.upgrad.com/tech-academy-fsd/?utm_source=BLOG&utm_medium=SIDEBAR&utm_campaign=TV_FSD_CERT_BLOG_SIDEBAR_TRENDING_1)
* [PG Diploma in Software Development Specialisation in Full Stack Development from IIIT-B - Duration 12 Months](https://www.upgrad.com/full-stack-developer-course-pgd-iiitb/?utm_source=BLOG&utm_medium=SIDEBAR&utm_campaign=TV_FSD_PGD_BLOG_SIDEBAR_TRENDING_2)
* [PG Diploma in Software Development Specialisation in Cyber Security IIIT-B - Duration 12 Months](https://www.upgrad.com/cyber-security-pgd-iiitb/?utm_source=BLOG&utm_medium=SIDEBAR&utm_campaign=TV_CC_PGD_BLOG_SIDEBAR_TRENDING_3)

Editor’s Picks

* [Best Online MBA Courses in India for 2021: Which One Should You Choose?](https://www.upgrad.com/blog/best-online-mba-courses-india/)
* [42 Exciting Python Project Ideas & Topics for Beginners [2021]](https://www.upgrad.com/blog/python-projects-ideas-topics-beginners/)
* [Top 9 Highest Paid Jobs in India for Freshers 2021 [A Complete Guide]](https://www.upgrad.com/blog/top-highest-paid-jobs-in-india-for-freshers/)

Accelerate Your Career with upGrad

Top of Form









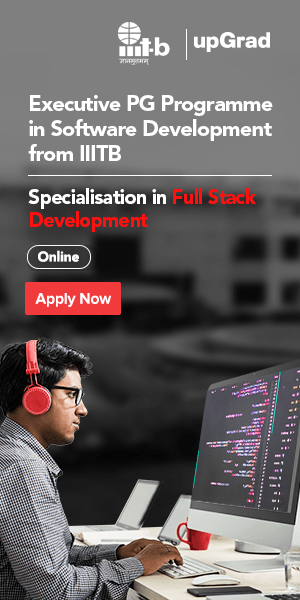




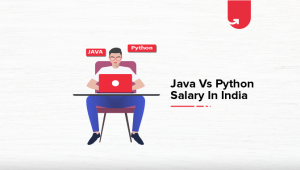
*By clicking 'Submit' you Agree to*[*UpGrad's Terms & Conditions*](https://www.upgrad.com/terms?iref=Blog_Leadform_TermsNConditions)*.*

Bottom of Form

Our Popular Software Engineering Courses

[](https://www.upgrad.com/full-stack-developer-course-pgd-iiitb/?utm_source=BLOG&utm_medium=FOOTER&utm_campaign=TV_FSD_PGD_BLOG_SIDEBAR_POPULAR)

**Related Articles**

[[](https://www.upgrad.com/blog/java-vs-python-salary/)](https://www.upgrad.com/blog/java-vs-python-salary/)

[Java vs Python Salary : Various Job Roles in India](https://www.upgrad.com/blog/java-vs-python-salary/)

[Akshay Supleby Akshay SupleMay 28, 2021](https://www.upgrad.com/blog/java-vs-python-salary/)

[[](https://www.upgrad.com/blog/parameterized-constructor-in-c-plus-plus/)](https://www.upgrad.com/blog/parameterized-constructor-in-c-plus-plus/)

[Parameterized Constructor In C++: Working & Examples](https://www.upgrad.com/blog/parameterized-constructor-in-c-plus-plus/)

[Rohan Vatsby Rohan VatsMay 27, 2021](https://www.upgrad.com/blog/parameterized-constructor-in-c-plus-plus/)

[[](https://www.upgrad.com/blog/software-testing-projects-topics/)](https://www.upgrad.com/blog/software-testing-projects-topics/)

[9 Exciting Software Testing Projects & Topics For Beginners [2021]](https://www.upgrad.com/blog/software-testing-projects-topics/)

[Rohan Vatsby Rohan VatsMay 27, 2021](https://www.upgrad.com/blog/software-testing-projects-topics/)

Building Careers of Tomorrow

**DATA SCIENCE**

* [Data Science | Executive PG](https://www.upgrad.com/data-science-pgd-iiitb/?utm_source=BLOG&utm_medium=MAIN_FOOTER&utm_campaign=DV_DA_PGD_BLOG_MAIN_FOOTER)
* [Data Science | Advanced Certificate](https://www.upgrad.com/data-science-pgc-iiitb/?utm_source=BLOG&utm_medium=MAIN_FOOTER&utm_campaign=DV_DA_PGC_BLOG_MAIN_FOOTER)
* [Data Science | Master](https://www.upgrad.com/data-science-masters-degree-iiitb/?utm_source=BLOG&utm_medium=MAIN_FOOTER&utm_campaign=DV_DA_MS_BLOG_MAIN_FOOTER)
* [Machine Learning & AI | Executive PG](https://www.upgrad.com/machine-learning-ai-pgd-iiitb/?utm_source=BLOG&utm_medium=MAIN_FOOTER&utm_campaign=DV_ML_PGD_BLOG_MAIN_FOOTER)
* [Machine Learning & AI | Masters](https://www.upgrad.com/masters-in-ml-ai-ljmu-iiitb/?utm_source=BLOG&utm_medium=MAIN_FOOTER&utm_campaign=DV_ML_MS_BLOG_MAIN_FOOTER)
* [Machine Learning & NLP | Advanced Certificate](https://www.upgrad.com/machine-learning-nlp-pgc-iiitb/?utm_source=BLOG&utm_medium=MAIN_FOOTER&utm_campaign=DV_ML_PGC_BLOG_MAIN_FOOTER)
* [Machine Learning and Cloud | Advanced Certification](https://www.upgrad.com/advanced-certification-ml-cloud-iitm/?utm_source=BLOG&utm_medium=MAIN_FOOTER&utm_campaign=DV_ML_CERT_BLOG_MAIN_FOOTER)

**TECHNOLOGY**

* [Full Stack Development | Executive PG](https://www.upgrad.com/full-stack-developer-course-pgd-iiitb/?utm_source=BLOG&utm_medium=MAIN_FOOTER&utm_campaign=TV_FSD_PGD_BLOG_MAIN_FOOTER)
* [Computer Science | Masters](https://www.upgrad.com/software-engineering-mcs-ljmu/?utm_source=BLOG&utm_medium=MAIN_FOOTER&utm_campaign=TV_FSD_MS_BLOG_MAIN_FOOTER)
* [Full Stack Development | Placement Track](https://www.upgrad.com/pgc-software-development-bridgelabz/?utm_source=BLOG&utm_medium=MAIN_FOOTER&utm_campaign=TV_FSD_PGC_BLOG_MAIN_FOOTER)
* [Full Stack Development | PG Certification](https://www.upgrad.com/tech-academy-fsd-techm/?utm_source=BLOG&utm_medium=MAIN_FOOTER&utm_campaign=TA_SE_PGC_BLOG_MAIN_FOOTER)
* [Blockchain Technology | Executive PG](https://www.upgrad.com/blockchain-certification-pgd-iiitb/?utm_source=BLOG&utm_medium=MAIN_FOOTER&utm_campaign=TV_BC_PGD_BLOG_MAIN_FOOTER)
* [Blockchain Technology | Executive Program](https://www.upgrad.com/blockchain-managers/?utm_source=BLOG&utm_medium=MAIN_FOOTER&utm_campaign=TV_BC_EV_BLOG_MAIN_FOOTER)
* [Blockchain Technology | Advanced Certificate](https://www.upgrad.com/blockchain-certification-pgc-iiitb/?utm_source=BLOG&utm_medium=MAIN_FOOTER&utm_campaign=TV_BC_PGC_BLOG_MAIN_FOOTER)
* [Big Data | Executive PG](https://www.upgrad.com/big-data-pgd-iiitb/?utm_source=BLOG&utm_medium=MAIN_FOOTER&utm_campaign=TV_BE_PGD_BLOG_MAIN_FOOTER)

**MANAGEMENT**

* [MBA (Executive) | NMIMS](https://www.upgrad.com/mba-business-analytics-nmims/?utm_source=BLOG&utm_medium=MAIN_FOOTER&utm_campaign=MBA_MBA_NMIMS_BLOG_MAIN_FOOTER)
* [MBA (Global) | DBS Australia](https://www.upgrad.com/mba-deakin-university/?utm_source=BLOG&utm_medium=MAIN_FOOTER&utm_campaign=MBA_DK_MBAG_BLOG_MAIN_FOOTER)
* [MBA (Global) | LBS UK](https://www.upgrad.com/mba-liverpool-business-school/?utm_source=BLOG&utm_medium=MAIN_FOOTER&utm_campaign=MBA_LBS_MBAG_BLOG_MAIN_FOOTER)
* [Digital Marketing | Advanced Certificate](https://www.upgrad.com/digital-marketing-and-communication-pgc-mica/?utm_source=BLOG&utm_medium=MAIN_FOOTER&utm_campaign=MV_DM_PGC_BLOG_MAIN_FOOTER)
* [Management | PG Program](https://www.upgrad.com/management-pgp-imt/?utm_source=BLOG&utm_medium=MAIN_FOOTER&utm_campaign=MV_IMT_PGP_BLOG_MAIN_FOOTER)
* [Product Management | Certification](https://www.upgrad.com/product-management/?utm_source=BLOG&utm_medium=MAIN_FOOTER&utm_campaign=MV_PM_CERT_BLOG_MAIN_FOOTER)
* [Sales & Digital Marketing | PG Program](https://www.upgrad.com/sales-and-marketing-pgpm-imt/?utm_source=BLOG&utm_medium=MAIN_FOOTER&utm_campaign=MV_SDM_PGP_BLOG_MAIN_FOOTER)
* [Life Insurance | PG Program](https://www.upgrad.com/insurance-hdfc-pgp/?utm_source=BLOG&utm_medium=MAIN_FOOTER&utm_campaign=JB_PGPI_PGP_BLOG_MAIN_FOOTER)
* [Business Analytics | Certification](https://www.upgrad.com/business-analytics-certification/?utm_source=BLOG&utm_medium=MAIN_FOOTER&utm_campaign=MV_BA_CERT_BLOG_MAIN_FOOTER)
* [Life Insurance (PNB) | PG Program](https://www.upgrad.com/insurance-pnb-pgpli/?utm_source=BLOG&utm_medium=MAIN_FOOTER&utm_campaign=JG_PNB_PGP_BLOG_MAIN_FOOTER)
* [Management | PG Program](https://www.upgrad.com/management-academy-pgp-imt-joblink/?utm_source=BLOG&utm_medium=MAIN_FOOTER&utm_campaign=MBA_IMT_ALL_BLOG_MAIN_FOOTER)
* [MBA (Global) | Deakin Business School](https://www.upgrad.com/mba-imt-deakin-university/?utm_source=BLOG&utm_medium=MAIN_FOOTER&utm_campaign=MBA_DK_ALL_BLOG_MAIN_FOOTER)
* [MBA (DFB) | JGU](https://www.upgrad.com/mba-digital-finance-and-banking-jgu/?utm_source=BLOG&utm_medium=MAIN_FOOTER&utm_campaign=MV_JGU_MBAF_BLOG_MAIN_FOOTER)

**ENTERPRISE PROGRAMS**

* [Data For Enteprise](https://www.upgrad.com/enterprise-data-sciences/)
* [Technology For Enterprise](https://www.upgrad.com/enterprise-product-and-technology/)
* [Management For Enterprise](https://www.upgrad.com/enterprise-management/)
* [UPGRAD](https://www.upgrad.com/?utm_source=BLOG&utm_medium=MAIN_FOOTER&utm_campaign=BLOG_MAIN_FOOTER)
* [Contact Us](https://www.upgrad.com/contact/)
* [Terms & Conditions](https://www.upgrad.com/terms/)
* [Privacy Policy](https://www.upgrad.com/privacy/)

© 2015–2021 upGrad Education Private Limited.

[**Javarevisited**](https://javarevisited.blogspot.com/)

Blog about Java, Programming, Spring, Hibernate, Interview Questions, Books and Online Course Recommendations from Udemy, Pluralsight, Coursera, etc

* [Home](https://javarevisited.blogspot.com/)
* [core java](https://javarevisited.blogspot.com/search/label/core%20java?max-results=3)
* [spring](https://javarevisited.blogspot.com/search/label/spring?max-results=3)
* [hibernate](https://javarevisited.blogspot.com/search/label/hibernate?max-results=3)
* [collections](https://javarevisited.blogspot.com/search/label/java%20collection%20tutorial?&max-results=3)
* [multithreading](https://javarevisited.blogspot.com/search/label/Java%20multithreading%20Tutorials?max-results=3)
* [design patterns](https://javarevisited.blogspot.com/search/label/design%20patterns?max-results=3)
* [interview questions](https://javarevisited.blogspot.com/search/label/interview%20questions?max-results=3)
* [coding](https://javarevisited.blogspot.com/search/label/Coding%20Interview%20Question?max-results=3)
* [data structure](https://javarevisited.blogspot.com/search/label/data%20structure%20and%20algorithm?max-results=3)
* [OOP](https://javarevisited.blogspot.com/search/label/object%20oriented%20programming?max-results=3)
* [java 8](https://javarevisited.blogspot.com/search/label/Java%208?max-results=3)
* [books](https://javarevisited.blogspot.com/search/label/books?max-results=3)
* [About Me](https://javarevisited.blogspot.com/p/about-me.html)
* [Java Certifications](https://javarevisited.blogspot.com/search/label/Java%20Certification%20OCPJP%20SCJP?&max-results=3)
* [JDBC](https://javarevisited.blogspot.com/search/label/JDBC?&max-results=3)
* [jsp-servlet](https://javarevisited.blogspot.com/search/label/jsp-servlet?&max-results=3)
* [JSON](https://javarevisited.blogspot.com/search/label/Java%20JSON%20tutorial?&max-results=3)
* [SQL](https://javarevisited.blogspot.com/search/label/SQL?max-results=3)
* [Linux](https://javarevisited.blogspot.com/search/label/linux?max-results=3)
* [Courses](https://javarevisited.blogspot.com/search/label/courses?max-results=3)
* [online resources](https://javarevisited.blogspot.com/search/label/online%20resources?&max-results=3)
* [jvm-internals](https://javarevisited.blogspot.com/search/label/JVM%20Internals?max-results=3)
* [REST](https://javarevisited.blogspot.com/search/label/REST?max-results=3)
* [Eclipse](https://javarevisited.blogspot.com/search/label/Eclipse?max-results=3)
* [jQuery](https://javarevisited.blogspot.com/search/label/JQuery?max-results=3)
* [Java IO](https://javarevisited.blogspot.com/search/label/java%20IO%20tutorial?&max-results=3)
* [Java XML](https://javarevisited.blogspot.com/search/label/Java%20xml%20tutorial?&max-results=3)

**[10 Java Exception and Error Interview Questions Answers](https://javarevisited.blogspot.com/2013/06/10-java-exception-and-error-interview-questions-answers-programming.html" \o "10 Java Exception and Error Interview Questions Answers)**

You will always see some interview questions from Exception and Error handling in core Java Interviews. Exception handling is an important aspect of Java application development and its key to writing robust, stable Java programs, which makes it natural favorites on interviews. Questions from Error and Exception in Java mostly based on the concept of Exception and Error in Java, How to handle Exception , [best practices to follow during Exception handling](http://javarevisited.blogspot.com/2013/03/0-exception-handling-best-practices-in-Java-Programming.html) etc. Though multithreading, garbage collection, JVM concepts and questions from object oriented design rules these interviews, you should always expect and prepare some questions on effective error handling.

Some Interviewer also  test debugging skill of programmers, as resolving Exceptions quickly is another trait of solid Java programming knowledge.  
  
If programmer is familiar with infamous and dodgy [ClassNotFoundException](http://javarevisited.blogspot.com/2011/08/classnotfoundexception-in-java-example.html) or [OutOfMemoryError](http://javarevisited.blogspot.com/2012/01/tomcat-javalangoutofmemoryerror-permgen.html), there is a good chance that he has some good practical experience under his belt.  
  
In this article we will see some Java Error and Exception interview questions asked to fresher, experienced and senior Java developers in Java J2EE interviews.

**Java Exception and Error Interview Questions**

[Java Exception and Error Interview Question Answers](https://4.bp.blogspot.com/-K6q0DQ1v-tw/TWu8owBtc2I/AAAAAAAAADA/oBoHDBiJ8ag/s40/17.jpg)Here is my list of frequently asked questions from Java Error and Exception topics in various programming interviews to Java and J2EE developers.  I have also shared my answers for these questions for quick revision, and provided source for more in depth understanding.  
  
I have tried to include questions of various difficulty level, including simplest of simple for freshers and some tricky questions for senior Java developers.  
  
If you think, there is a good question, which is not included in this list, please feel free to share it via comment. You can also share error handling questions asked to you on interviews or any question, for which you don’t know the answer.

**1) What is Exception in Java?**

This is always been first interview question on Exception and mostly asked on fresher level interviews. I haven't seen anybody asking about what is Exception in senior and experienced level interviews, but this is quite popular at entry level. In simple word Exception is Java’s way to convey both system and programming errors.  
  
In Java Exception feature is implemented by using class like Throwable, Exception, RuntimeException and keywords like throw, throws, try, catch and finally. All Exception are derived form Throwable class.  
  
Throwable further divides errors in too category one is java.lang.Exception and other is java.lang.Error.  java.lang.Error deals with system errors like java.lang.StackOverFlowError or [Java.lang.OutOfMemoryError](http://javarevisited.blogspot.com/2011/09/javalangoutofmemoryerror-permgen-space.html) while Exception is mostly used to deal with programming mistakes, non availability of requested resource etc.

**2) What is the difference between Checked and Unchecked Exception in Java?**

This is another popular Java Exception interview question appears in almost all level of Java interviews. Main difference between Checked and Unchecked Exception lies in there handling. Checked Exception requires to be handled at compile time using try, catch and finally keywords or else compiler will flag error. This is not a requirement for Unchecked Exceptions. Also all exceptions derived from java.lang.Exception classes are checked exception, exception those which extends RuntimeException, these are known as unchecked exception in Java. You can also check next article for [more differences between Checked and Unchecked Exception](http://javarevisited.blogspot.com/2011/12/checked-vs-unchecked-exception-in-java.html).

**3) What is the similarity between NullPointerException and ArrayIndexOutOfBoundException in Java?**

This is Java Exception interview question was not very popular, but appears in various fresher level interviews, to see whether candidate is familiar with concept of checked and unchecked exception or not. By the way answer of this interview question is both of them are example of unchecked exception and derived form RuntimeException. This question also opens door for difference of array in Java and C programming language, as arrays in C are unbounded and never throw ArrayIndexOutOfBoundException.

**4) What best practices you follow while doing Exception handling in Java?**

This Exception interview question in Java is very popular while hiring senior java developer of Technical Lead. Since exception handling is crucial part of project design and good knowledge of this is desirable. There are lot of best practices, which can help to make your code robust and flexible at same time, here are few of them:

1) Returning boolean instead of returning null to avoid NullPointerException at callers end. Since NPE is most infamous of all Java exceptions, there are lot of techniques and [coding best practices to minimize NullPointerException](http://javarevisited.blogspot.com/2013/05/ava-tips-and-best-practices-to-avoid-nullpointerexception-program-application.html). You can check that link for some specific examples.

2) Non empty catch blocks. Empty catch blocks  are considered as one of the bad practices in Exception handling because they just ate Exception without any clue, at bare minimum print stack trace but you should do alternative operation which make sense or defined by requirements.

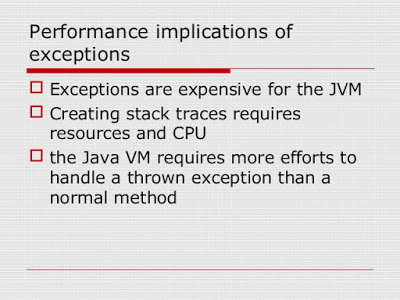
3) Prefer Unchecked exception over checked until you have a very good reason of not to do so. it improves readability of

code by removing boiler plate exception handling code

.

4) Never let your database Exception flowing till client error. since most of application deal with database and [SQLException](http://javarevisited.blogspot.com/2012/01/javasqlsqlexception-invalid-column.html) is a checked Exception in Java you should consider handling any database related errors in DAO layer of your application and only returning alternative value or something meaningful RuntimeException which client can understand and take action.

5) calling close() methods for connections, statements, and streams on finally block in Java.

[](https://2.bp.blogspot.com/-Xya2SK0Iva0/VXhOeWE7svI/AAAAAAAAC-8/FzhFhLejgQY/s1600/Java+Exception+Performance+impact.jpg)

I have already shared lot of these in my post [Top 10 Java exception handling best practices](http://javarevisited.blogspot.com/2013/03/0-exception-handling-best-practices-in-Java-Programming.html), you can also refer that for more knowledge on this topic.

**5) Why do you think Checked Exception exists in Java since we can also convey error using RuntimeException?**

This is a controversial question and you need to be careful while answering this interview question. Though they will definitely like to hear your opinion, what they are most interested in convincing reason. One of the reasons I see is that its a design decision, which is influenced by experience in programming language prior to Java like C++.  
  
Most of the checked exceptions are in java.io package, which makes sense because if you request any system resource and it's not available then a robust program must be able to handle that situation gracefully. By declaring IOException as checked Exception, Java ensures that you provide that gracefully exception handling.  
  
Another possible reason could be to ensuring that system resources like file descriptors, which are limited in numbers, should be released as soon as you are done with that using catch or finally block. Effective Java book from Joshua Bloch has a couple of items on this topic, which is again worth reading.

**6) What is the difference between throw and throws keyword in Java?**

One more Java Exception interview questions from beginners kitty. throw and throws keyword may look quite similar, especially if you are new to Java programming and haven't seen much of it. Though they are similar in terms that both are used in Exception handling, they are different on how and where they are used in code. throws keyword is used in method signature to declare which checked exception method can throw, you can also declare unchecked exception, but that is not mandatory by the compiler.  
  
This signifies a lot of things like method is not going to handle Exception instead it's throwing it, if method throws checked Exception then the caller should provide compile-time exception handling etc.  
  
On the other hand, throw keyword is actually used to throw any Exception. Syntactically you can throw any Throwable (i.e. Throwable or any class derived from Throwable) , throw keyword transfers control of execution to the caller so it can be used in place of return keyword. Most common example of using throw in place of return is throwing UnSupportedOperationException from an empty method as shown below :

private static void show() {

throw new UnsupportedOperationException("Not yet implemented");

}

See [this article](http://javarevisited.blogspot.com/2012/02/difference-between-throw-and-throws-in.html) for more differences between these two keywords in Java.

**7) What is Exception chaining in Java?**

Exception chaining is a popular exception handling concept in Java, where another exception is thrown in response of an exception and creating a chain of Exceptions. This technique mostly used to wrap a checked exception into an unchecked or RuntimeException. By the way if you are throwing new exception due to another exception then always include original exception so that handler code can access root cause by using methods like getCause() and initCause().

**8) Have you written your own custom Exception in Java? How do you do that?**

Ofcourse most of us has written custom or business Exceptions like AccountNotFoundExcepiton. Main purpose of asking this Java Exception interview question is to find out how you use this feature. This can be used for sophisticated and precise exception handling with tweak involved in whether you would choose a checked or unchecked exception.  
  
By creating a specific exception for specific case, you also gives lot of options to caller to deal with them elegantly. I always prefer to have a precise exception than a general exception. Though creating lots of specific exceptions quickly increasing number of classes in your project, maintaining a practical balance between specific and general exceptions are key to success.

**9) What changes have been introduced in JDK7 related to Exception handling in Java?**

A relatively new and recent Exception interview question in Java. JDK7 has introduced two major feature which is related to Error and Exception handling,  one is the ability to handle [multiple exceptions in one catch block](http://javarevisited.blogspot.com/2011/07/jdk7-multi-cache-block-example-tutorial.html), popularly known as multi-cache block and other is [ARM blocks in Java 7](http://javarevisited.blogspot.sg/2011/09/arm-automatic-resource-management-in.html) for automatic resource management, also known as to try with the resource.  
  
Both of these features can certainly help to reduce boilerplate code required for handling checked exceptions in Java and significantly improves the readability of code. Knowledge of this feature, not only helps to write better error and exception code in Java but also helps to do well during interviews. I also recommend reading Java 7 Recipes book to get more insight on useful features introduced in Java 7, including these two.

**10) Have you faced OutOfMemoryError in Java? How did you solve that?**

This Java Error interview question is mostly asked on senior-level Java interviews and here the interviewer is interested in your approach to tackling dangerous OutOfMemoryError. Admit it we always face this error no matter which kind of project you are working so if you say no it doesn't go very well with the interviewer. I suggest even if you are not familiar or not face it in reality but have 3 to 4 years of experience in Java, be prepare for it.  
  
At the same time, this is also a chance to impress the interviewer by showing your advanced technical knowledge related to finding memory leaks, profiling, and debugging.  
  
I have noticed that these skills almost always creates a positive impression. You can also see my post on [how to fix java.lang.OutOfMemoryError](http://javarevisited.blogspot.com/2011/09/javalangoutofmemoryerror-permgen-space.html) for more detail on this topic.

**11) Does code form finally execute if the method returns before finally block or JVM exits?**

This Java exception interview question can also be asked in code format, where given a code with System.exit() in the try block and something in finally block. It’s worth knowing that, finally block in Java executes even when the return keyword is used in try block. Only time they don’t execute is when you call JVM to exit by executing System.exit(0)from try block in Java.

**12) What is the difference in final, finalize, and finally keyword in Java?**

Another classic interview question in core Java, this was asked to one of my friend on his telephonic interview for core Java developer with Morgan Stanley. final and finally are keyword, while finalize is method. final keyword is very useful for creating ad [Immutable class in Java](http://javarevisited.blogspot.com/2013/03/how-to-create-immutable-class-object-java-example-tutorial.html) By making a class final, we prevent it from being extended, similarly by making a method final, we prevent it from being overridden,. On the other hand, finalize() method is called  by garbage collector, before that object is collected, but this is not guaranteed by Java specification. finally keyword is the only one which is related to error and exception handling and you should always have finally block in production code for closing connection and resources. See [here](http://javarevisited.blogspot.com/2012/11/difference-between-final-finally-and-finalize-java.html) for more detailed answer of this question.

**13) What is wrong with following code :**

 public static void start() throws IOException, RuntimeException{

**throw** **new** RuntimeException("Not able to Start");

 }

 public static void main(String args[]) {

**try** {

          start();

    } **catch** (Exception ex) {

            ex**.**printStackTrace();

    } **catch** (RuntimeException re) {

            re**.**printStackTrace();

    }

 }

This code will throw compiler error on line where RuntimeException  variable “re” is written on catch block. since Exception is super class of RuntimeException, all RuntimeException thrown by start() method will be captured by first catch block and code will never reach second catch block and that's the reason compiler will flag error as  *“exception java.lang.RuntimeException has already been caught"*.

**14) What is wrong with following code in Java:**

public class *SuperClass* {

    public void **start**() throws IOException{

**throw** **new** IOException("Not able to open file");

    }

}

public class *SubClass* extends *SuperClass*{

    public void **start**() throws Exception{

**throw** **new** Exception("Not able to start");

    }

}

In this code compiler will complain on sub class where start() method gets overridden. As per [rules of method overriding in Java](http://java67.blogspot.com/2012/08/what-is-method-overriding-in-java-example-tutorial.html), an overridden method can not throw Checked Exception which is higher in hierarchy than original method. Since here start() is throwing IOException in super class, start() in sub class can only throw either IOException or any sub class of IOException but not super class of IOException e.g. Exception.

**15) What is wrong with following Java Exception code:**

public static void **start**(){

   System.**out**.println("Java Exception interivew question Answers for Programmers");

}

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

**try**{

      start();

   }**catch**(**IOException** ioe){

      ioe.printStackTrace();

   }

}

In this Java Exception example code, the compiler will complain about the line where we are handling IOException, since IOException is a checked Exception and start() method doesn't throw IOException, so the compiler will flag error as "exception java.io.IOException is never thrown in body of corresponding try statement", but if you change IOException to Exception compiler error will disappear because Exception can be used to catch all RuntimeException which doesn't require a declaration in a throws clause.  
  
 I like this little tricky Java Exception interview question because it's not easy to figure out the result by chaining IOException to Exception. You can also check [Java Puzzlers by Joshua Bloch and Neil Gafter](http://www.amazon.com/dp/032133678X/?tag=javamysqlanta-20)for some tricky questions based on Java Errors and Exceptions.

These are some of Java Error and Exception interview questions, I have mostly seen in both fresher and experienced level of Java interviews. There are a lot more questions on Exception which I haven't included and if you think you have a good question missed out than let me know and I will make effort to include it on this list of java exceptions questions and answers.  
  
One last question of Java Exception I am leaving for you guys is **"Why Java Exception considered to be a better alternative of returning error codes"**, let me know what is your thought on this list of Java Exception interview questions and answers.

**Further Learning**  
[Complete Java Masterclass](https://click.linksynergy.com/fs-bin/click?id=JVFxdTr9V80&subid=0&offerid=323058.1&type=10&tmpid=14538&RD_PARM1=https%3A%2F%2Fwww.udemy.com%2Fjava-the-complete-java-developer-course%2F)  
[Java Fundamentals: The Java Language](https://pluralsight.pxf.io/c/1193463/424552/7490?u=https%3A%2F%2Fwww.pluralsight.com%2Fcourses%2Fjava-fundamentals-language)  
[Java In-Depth: Become a Complete Java Engineer!](https://click.linksynergy.com/fs-bin/click?id=JVFxdTr9V80&subid=0&offerid=323058.1&type=10&tmpid=14538&RD_PARM1=https%3A%2F%2Fwww.udemy.com%2Fjava-in-depth-become-a-complete-java-engineer%2F)

Top of Form

**DOWNLOAD YOUR FREE JAVA INTERVIEW GUIDE**

Join my list for exclusive discounts, coupons, and early access to my books and courses.

​

And, download a free chapter from my best selling Grokking the Java interview book when you sign up



Download

We respect your privacy. Unsubscribe at any time.

[Built with ConvertKit](https://convertkit.com/?utm_source=dynamic&utm_medium=referral&utm_campaign=poweredby&utm_content=form)

Bottom of Form

By javin paul [[https://img1.blogblog.com/img/icon18_email.gif](https://draft.blogger.com/email-post.g?blogID=8712770457197348465&postID=8351021684635911098)](https://draft.blogger.com/email-post.g?blogID=8712770457197348465&postID=8351021684635911098)

[Email This](https://draft.blogger.com/share-post.g?blogID=8712770457197348465&postID=8351021684635911098&target=email) [BlogThis!](https://draft.blogger.com/share-post.g?blogID=8712770457197348465&postID=8351021684635911098&target=blog) [Share to Twitter](https://draft.blogger.com/share-post.g?blogID=8712770457197348465&postID=8351021684635911098&target=twitter) [Share to Facebook](https://draft.blogger.com/share-post.g?blogID=8712770457197348465&postID=8351021684635911098&target=facebook)

Labels: [core java](https://javarevisited.blogspot.com/search/label/core%20java?max-results=3), [core java interview question](https://javarevisited.blogspot.com/search/label/core%20java%20interview%20question?max-results=3), [error and exception](https://javarevisited.blogspot.com/search/label/error%20and%20exception?max-results=3)

**12 comments :**

**[Javier](https://draft.blogger.com/profile/11682700480438441325)said...**

Hello,  
  
The "what's wrong" questions had reminded me a funny error that you could make if you want to "raise" an exception but you forget the keyword "throw".  
  
You can find a footnote at http://www.javaadvent.com/2012/12/of-hacking-enums-and-modifying-final.html  
  
At the beginning of the article there is a code that has a problem. The author mentioned it, and he provides an explanation at the end.  
  
I think reading it it's worth it, so I recommend you, as well as  
  
I think it's worth reading it, so I recommend you to do it, as well as all the articles at http://www.javaadvent.com/

[June 27, 2013 at 11:58 PM](https://javarevisited.blogspot.com/2013/06/10-java-exception-and-error-interview-questions-answers-programming.html?showComment=1372402732953#c2570957371301566779)

**[Developer Dude](https://draft.blogger.com/profile/06513887281284502342)said...**

"4) What best practices you follow while doing Exception handling in Java ?"  
  
You missed: do not return from a finally clause.  
  
https://weblogs.java.net/blog/staufferjames/archive/2007/06/\_dont\_return\_in.html

[July 10, 2013 at 8:30 AM](https://javarevisited.blogspot.com/2013/06/10-java-exception-and-error-interview-questions-answers-programming.html?showComment=1373470240639#c6672205442646134232)

**[Unknown](https://draft.blogger.com/profile/00858214960977873443)said...**

how to slove this error in java programming??  
  
java.lang.NoClassDefFoundError: pro2/Sen2  
Caused by: java.lang.ClassNotFoundException: pro2.Sen2  
at java.net.URLClassLoader$1.run(URLClassLoader.java:202)  
at java.security.AccessController.doPrivileged(Native Method)  
at java.net.URLClassLoader.findClass(URLClassLoader.java:190)  
at java.lang.ClassLoader.loadClass(ClassLoader.java:306)  
at sun.misc.Launcher$AppClassLoader.loadClass(Launcher.java:301)  
at java.lang.ClassLoader.loadClass(ClassLoader.java:247)  
Exception in thread "main"

[January 17, 2014 at 1:27 AM](https://javarevisited.blogspot.com/2013/06/10-java-exception-and-error-interview-questions-answers-programming.html?showComment=1389950831671#c9215479821324468394)

**[Unknown](https://draft.blogger.com/profile/04031518329126158540)said...**

Hi Sir Iam new to java and  
my qsn is:At what time Exception occurs(compile time or runtime)?

[March 26, 2014 at 10:55 AM](https://javarevisited.blogspot.com/2013/06/10-java-exception-and-error-interview-questions-answers-programming.html?showComment=1395856551652#c8878618193741284011)

**[www.letsride.in](https://draft.blogger.com/profile/15204790426694961202)said...**

Also all exceptions derived from java.lang.Exception classes are checked exception, exception those which extends RuntimeException, these are known as unchecked exception in Java.  
  
There is a type error is this... please update  
It should be " except those which extends RuntimeException "

[April 3, 2014 at 7:39 AM](https://javarevisited.blogspot.com/2013/06/10-java-exception-and-error-interview-questions-answers-programming.html?showComment=1396535988478#c2324999609943365424)

**Anonymous said...**

HI,  
  
I am sharing recently ask interview questions as below:  
1.Is there another way to handle exception without using try,catch and finally?  
2.Can we create custom run time exception?

[September 1, 2014 at 7:03 PM](https://javarevisited.blogspot.com/2013/06/10-java-exception-and-error-interview-questions-answers-programming.html?showComment=1409623434886#c8372028429782964110)

**[Unknown](https://draft.blogger.com/profile/06188693685179050553)said...**

@Anonymous...  
1. using throws declaration  
2. yes we can by extending RuntimeException class.

[May 5, 2015 at 11:38 PM](https://javarevisited.blogspot.com/2013/06/10-java-exception-and-error-interview-questions-answers-programming.html?showComment=1430894332462#c8137048155199336128)

**[Unknown](https://draft.blogger.com/profile/15682636966885662395)said...**

@Anonymous  
You can use UncaughtExceptionHandler interface  
  
http://stackoverflow.com/questions/6243887/java-exceptions-handling-exceptions-without-try-catch

[September 26, 2015 at 7:37 PM](https://javarevisited.blogspot.com/2013/06/10-java-exception-and-error-interview-questions-answers-programming.html?showComment=1443321422860#c3359388632753307721)

**[Priya Gupta](https://draft.blogger.com/profile/05464343814343099354)said...**

Hi,  
  
Recently I attended one interview and the question has been asked if we can catch 'RuntimeException'? I said yes we can catch but RuntimeException's should not occur at the first place as this is programing errors but the interviewer doesn't seem to be satisfied. Please correct me if i am wrong.

[November 30, 2015 at 2:46 AM](https://javarevisited.blogspot.com/2013/06/10-java-exception-and-error-interview-questions-answers-programming.html?showComment=1448880396185#c3567460586488807561)

**[Unknown](https://draft.blogger.com/profile/04841836707257970810)said...**

@Priya: Your answer to the interviewer is right. Generally, it is not recommended to catch runtime exceptions as trying to handle them would cause more harm as it is usually very difficult to recover from them as they are about some programmming flaws. However, in some very specific situations it would be required to handle them.

[April 26, 2016 at 9:42 AM](https://javarevisited.blogspot.com/2013/06/10-java-exception-and-error-interview-questions-answers-programming.html?showComment=1461688930062#c2481378740685388031)

**[Unknown](https://draft.blogger.com/profile/04242085146565991061)said...**

There is no need to handle runtimeexception because these exceprion are handle by jvm automatically

[June 13, 2017 at 9:24 PM](https://javarevisited.blogspot.com/2013/06/10-java-exception-and-error-interview-questions-answers-programming.html?showComment=1497414263745#c7570504397842056228)

**[Unknown](https://draft.blogger.com/profile/01298609197226121088)said...**

please give short answers only.if anyone asked question

[August 6, 2017 at 9:55 AM](https://javarevisited.blogspot.com/2013/06/10-java-exception-and-error-interview-questions-answers-programming.html?showComment=1502038510879#c7430175285187503037)

**Post a Comment**

[Newer Post](https://javarevisited.blogspot.com/2015/06/2-ways-to-check-if-string-is-palindrome-java.html)[Older Post](https://javarevisited.blogspot.com/2015/06/org.postgresql.Driver-javalangclassnotfoundexception.html)[Home](https://javarevisited.blogspot.com/)

Subscribe to: [Post Comments ( Atom )](https://javarevisited.blogspot.com/feeds/8351021684635911098/comments/default)

**Disclosure - Javarevisited is learner-supported. When you buy through links on our site, we may earn an affiliate commission.**

**Join my list for updates**

Top of Form

Follow

Bottom of Form

**Search This Blog**

Top of Form

|  |  |
| --- | --- |
|  |  |

Bottom of Form

**Subscribe for Blog Posts**

Top of Form

|  |  |
| --- | --- |
|  |  |

Bottom of Form

**Interview Questions**

* [core java interview question](https://javarevisited.blogspot.com/search/label/core%20java%20interview%20question?max-results=3) (174)
* [data structure and algorithm](https://javarevisited.blogspot.com/search/label/data%20structure%20and%20algorithm?max-results=3) (75)
* [Coding Interview Question](https://javarevisited.blogspot.com/search/label/Coding%20Interview%20Question?max-results=3) (74)
* [interview questions](https://javarevisited.blogspot.com/search/label/interview%20questions?max-results=3) (63)
* [SQL Interview Questions](https://javarevisited.blogspot.com/search/label/SQL%20Interview%20Questions?max-results=3) (34)
* [object oriented programming](https://javarevisited.blogspot.com/search/label/object%20oriented%20programming?max-results=3) (33)
* [design patterns](https://javarevisited.blogspot.com/search/label/design%20patterns?max-results=3) (31)
* [thread interview questions](https://javarevisited.blogspot.com/search/label/thread%20interview%20questions?max-results=3) (30)
* [collections interview questions](https://javarevisited.blogspot.com/search/label/collections%20interview%20questions?max-results=3) (25)
* [spring interview questions](https://javarevisited.blogspot.com/search/label/spring%20interview%20questions?max-results=3) (23)
* [database interview questions](https://javarevisited.blogspot.com/search/label/database%20interview%20questions?max-results=3) (16)
* [servlet interview questions](https://javarevisited.blogspot.com/search/label/servlet%20interview%20questions?max-results=3) (15)
* [Programming interview question](https://javarevisited.blogspot.com/search/label/Programming%20interview%20question?max-results=3) (6)
* [hibernate interview questions](https://javarevisited.blogspot.com/search/label/hibernate%20interview%20questions?max-results=3) (6)

**Best of Javarevisited**

* [How Spring MVC works internally?](http://javarevisited.blogspot.sg/2017/06/how-spring-mvc-framework-works-web-flow.html)
* [How to design a vending machine in Java?](http://javarevisited.blogspot.sg/2016/06/design-vending-machine-in-java.html#axzz4sZVwtCgs)
* [How HashMap works in Java?](http://javarevisited.blogspot.sg/2011/02/how-hashmap-works-in-java.html#axzz4sZOoYUxv)
* [Why String is Immutable in Java?](http://javarevisited.blogspot.sg/2010/10/why-string-is-immutable-or-final-in-java.html#axzz4sZOoYUxv)
* [10 Articles Every Programmer Must Read](http://javarevisited.blogspot.sg/2014/05/10-articles-every-programmer-must-read.html)
* [How to convert lambda expression to method reference in Java 8?](http://javarevisited.blogspot.sg/2017/08/how-to-convert-lambda-expression-to-method-reference-in-java8-example.html)
* [10 Tips to improve Programming Skill](http://javarevisited.blogspot.sg/2014/01/10-tips-to-improve-programming-skill-become-better-programmer.html#axzz4sZOoYUxv)
* [10 OOP design principles programmer should know](http://javarevisited.blogspot.sg/2012/03/10-object-oriented-design-principles.html#axzz4sZOoYUxv)
* [How Synchronization works in Java?](http://javarevisited.blogspot.sg/2011/04/synchronization-in-java-synchronized.html#axzz4sZOoYUxv)
* [10 tips to work fast in Linux](http://javarevisited.blogspot.sg/2011/03/unix-command-tutorial-working-fast-in.html)
* [5 Books to improve Coding Skills](http://www.java67.com/2016/02/5-books-to-improve-coding-skills-of.html)

**Java Tutorials**

* [date and time tutorial](https://javarevisited.blogspot.com/search/label/date%20and%20time%20tutorial?&max-results=3) (23)
* [FIX protocol tutorial](https://javarevisited.blogspot.com/search/label/FIX%20protocol%20tutorial?&max-results=3) (15)
* [Java Certification OCPJP SCJP](https://javarevisited.blogspot.com/search/label/Java%20Certification%20OCPJP%20SCJP?&max-results=3) (31)
* [java collection tutorial](https://javarevisited.blogspot.com/search/label/java%20collection%20tutorial?&max-results=3) (77)
* [java IO tutorial](https://javarevisited.blogspot.com/search/label/java%20IO%20tutorial?&max-results=3) (29)
* [Java JSON tutorial](https://javarevisited.blogspot.com/search/label/Java%20JSON%20tutorial?&max-results=3) (12)
* [Java multithreading Tutorials](https://javarevisited.blogspot.com/search/label/Java%20multithreading%20Tutorials?&max-results=3) (59)
* [Java Programming Tutorials](https://javarevisited.blogspot.com/search/label/Java%20Programming%20Tutorials?&max-results=3) (19)
* [Java xml tutorial](https://javarevisited.blogspot.com/search/label/Java%20xml%20tutorial?&max-results=3) (16)
* [JDBC](https://javarevisited.blogspot.com/search/label/JDBC?&max-results=3) (32)
* [jsp-servlet](https://javarevisited.blogspot.com/search/label/jsp-servlet?&max-results=3) (37)
* [online resources](https://javarevisited.blogspot.com/search/label/online%20resources?&max-results=3) (216)

**Followers**

**Translate This Blog**



  Powered by [**[Google Translate](https://translate.google.com/)Translate**](https://translate.google.com/)

**Categories**

* [courses](https://javarevisited.blogspot.com/search/label/courses) (239)
* [SQL](https://javarevisited.blogspot.com/search/label/SQL) (65)
* [database](https://javarevisited.blogspot.com/search/label/database) (46)
* [linux](https://javarevisited.blogspot.com/search/label/linux) (42)
* [Java Certification OCPJP SCJP](https://javarevisited.blogspot.com/search/label/Java%20Certification%20OCPJP%20SCJP) (31)
* [Eclipse](https://javarevisited.blogspot.com/search/label/Eclipse) (30)
* [JVM Internals](https://javarevisited.blogspot.com/search/label/JVM%20Internals) (24)
* [JQuery](https://javarevisited.blogspot.com/search/label/JQuery) (21)
* [REST](https://javarevisited.blogspot.com/search/label/REST) (18)
* [Maven](https://javarevisited.blogspot.com/search/label/Maven) (15)
* [Testing](https://javarevisited.blogspot.com/search/label/Testing) (15)
* [general](https://javarevisited.blogspot.com/search/label/general) (13)

**Blog Archive**

* [►](javascript:void(0))[2021](https://javarevisited.blogspot.com/2021/)(254)
* [►](javascript:void(0))[2020](https://javarevisited.blogspot.com/2020/)(299)
* [►](javascript:void(0))[2019](https://javarevisited.blogspot.com/2019/)(54)
* [►](javascript:void(0))[2018](https://javarevisited.blogspot.com/2018/)(21)
* [▼](javascript:void(0))[2017](https://javarevisited.blogspot.com/2017/)(720)
  + [►](javascript:void(0))[December](https://javarevisited.blogspot.com/2017/12/)(83)
  + [►](javascript:void(0))[November](https://javarevisited.blogspot.com/2017/11/)(29)
  + [►](javascript:void(0))[October](https://javarevisited.blogspot.com/2017/10/)(43)
  + [►](javascript:void(0))[September](https://javarevisited.blogspot.com/2017/09/)(39)
  + [►](javascript:void(0))[August](https://javarevisited.blogspot.com/2017/08/)(50)
  + [►](javascript:void(0))[July](https://javarevisited.blogspot.com/2017/07/)(49)
  + [▼](javascript:void(0))[June](https://javarevisited.blogspot.com/2017/06/)(48)
    - [JSP - How to check if ArrayList is Empty using JST...](https://javarevisited.blogspot.com/2017/06/jsp-how-to-check-if-arraylist-is-empty-jstl-example.html)
    - [How to create Custom Exception in Java - Tutorial ...](https://javarevisited.blogspot.com/2014/06/how-to-create-custom-exception-in-java.html)
    - [Builder Design pattern in Java - Example Tutorial](https://javarevisited.blogspot.com/2012/06/builder-design-pattern-in-java-example.html)
    - [Is Joshua Bloch's Effective Java Still Valid in th...](https://javarevisited.blogspot.com/2017/06/is-joshua-blochs-effective-java-still-valid-in-the-era-of-java-8.html)
    - [10 xargs command example in Linux - Unix tutorial](https://javarevisited.blogspot.com/2012/06/10-xargs-command-example-in-linux-unix.html)
    - [2 Ways to check If String is Palindrome in Java? R...](https://javarevisited.blogspot.com/2015/06/2-ways-to-check-if-string-is-palindrome-java.html)
    - [10 Java Exception and Error Interview Questions An...](https://javarevisited.blogspot.com/2013/06/10-java-exception-and-error-interview-questions-answers-programming.html)
    - [java.lang.ClassNotFoundException: org.postgresql.D...](https://javarevisited.blogspot.com/2015/06/org.postgresql.Driver-javalangclassnotfoundexception.html)
    - [How to use CopyOnWriteArraySet in Java with Example](https://javarevisited.blogspot.com/2014/06/how-to-use-copyonwritearrayset-in-java-example-tutorial.html)
    - [How to Escape JSON String in Java- Eclipse IDE Tips](https://javarevisited.blogspot.com/2017/06/how-to-escape-json-string-in-java-eclipse-IDE.html)
    - [List of special bash parameter used in Unix or Li...](https://javarevisited.blogspot.com/2011/06/special-bash-parameters-in-script-linux.html)
    - [What is -XX:+UseCompressedOops in 64 bit JVM](https://javarevisited.blogspot.com/2012/06/what-is-xxusecompressedoops-in-64-bit.html)
    - [HashSet in Java – 10 Examples Programs Tutorial](https://javarevisited.blogspot.com/2012/06/hashset-in-java-10-examples-programs.html)
    - [How to create HTTP Server in Java - ServerSocket E...](https://javarevisited.blogspot.com/2015/06/how-to-create-http-server-in-java-serversocket-example.html)
    - [3 Exampls to Convert an Array to ArrayList in Java](https://javarevisited.blogspot.com/2011/06/converting-array-to-arraylist-in-java.html)
    - [JUnit4 Annotations : Test Examples and Tutorial](https://javarevisited.blogspot.com/2012/06/junit4-annotations-test-examples-and.html)
    - [2 Ways to Parse CSV Files in Java - BufferedReader...](https://javarevisited.blogspot.com/2015/06/2-ways-to-parse-csv-files-in-java-example.html)
    - [3 Ways to Find Duplicate Elements in an Array - Java](https://javarevisited.blogspot.com/2015/06/3-ways-to-find-duplicate-elements-in-array-java.html)
    - [10 example of using Vim or VI editor in UNIX and L...](https://javarevisited.blogspot.com/2011/06/vi-editor-in-unix-example-tutorial-and.html)
    - [Java Lock and Condition Example using Producer Con...](https://javarevisited.blogspot.com/2015/06/java-lock-and-condition-example-producer-consumer.html)
    - [3 ways to solve java.lang.NoClassDefFoundError in ...](https://javarevisited.blogspot.com/2011/06/noclassdeffounderror-exception-in.html)
    - [How to Add Two Integer Numbers without using Plus ...](https://javarevisited.blogspot.com/2013/06/how-to-add-two-integer-numbers-without-plus-arithmetic-operator-java-example.html)
    - [How to Close Java Program or Swing Application wit...](https://javarevisited.blogspot.com/2012/06/how-to-close-java-program-or-swing.html)
    - [How to use Comparator and Comparable in Java? With...](https://javarevisited.blogspot.com/2011/06/comparator-and-comparable-in-java.html)
    - [How to use Callable and Future in Java? Example](https://javarevisited.blogspot.com/2015/06/how-to-use-callable-and-future-in-java.html)
    - [How to Count number of Set bits or 1's of Integer ...](https://javarevisited.blogspot.com/2014/06/how-to-count-number-of-set-bits-or-1s.html)
    - [How to Connect to MySQL database in Java with Example](https://javarevisited.blogspot.com/2015/06/how-to-connect-to-mysql-database-in-java-jdbc-example.html)
    - [How to converts Java Object to XML - JAXB Example](https://javarevisited.blogspot.com/2015/06/how-to-converts-java-object-to-xml-jaxb.html)
    - [Is Java Compiled or Interpreted Programming language?](https://javarevisited.blogspot.com/2014/06/is-java-interpreted-or-compiled-programming-language.html)
    - [How to Generate MD5 checksum for Files in Java](https://javarevisited.blogspot.com/2013/06/how-to-generate-md5-checksum-for-files.html)
    - [How to compare two lists of values in Microsoft Ex...](https://javarevisited.blogspot.com/2017/06/how-to-compare-two-lists-of-values-in.html)
    - [How to Read Write Excel file in Java - POI Example](https://javarevisited.blogspot.com/2015/06/how-to-read-write-excel-file-java-poi-example.html)
    - [Top 30 Programming questions asked in Interview - ...](https://javarevisited.blogspot.com/2011/06/top-programming-interview-questions.html)
    - [10 Interview Questions on Java Generics for Progra...](https://javarevisited.blogspot.com/2012/06/10-interview-questions-on-java-generics.html)
    - [SynchronousQueue Example in Java - Produer Consume...](https://javarevisited.blogspot.com/2014/06/synchronousqueue-example-in-java.html)
    - [java.lang.NullPointerException - Common Cause of N...](https://javarevisited.blogspot.com/2012/06/common-cause-of-javalangnullpointerexce.html)
    - [Top 30 Array Interview Questions and Answers for P...](https://javarevisited.blogspot.com/2015/06/top-20-array-interview-questions-and-answers.html)
    - [10 Examples of Date Command in Linux and UNIX](https://javarevisited.blogspot.com/2012/06/10-examples-of-date-command-in-unix.html)
    - [Exception in thread "main" java.lang.ExceptionInIn...](https://javarevisited.blogspot.com/2014/06/exception-in-thread-main-java-lang-exceptionininitializerError-fix.html)
    - [How to get current URL, parameters and Hash tag us...](https://javarevisited.blogspot.com/2013/06/JQuery-JavaScript-tutorial-how-to-get-current-url-parameters-hashtag.html)
    - [How to get JAR files of Jackson Libary for JSON Pr...](https://javarevisited.blogspot.com/2017/06/how-to-get-jar-files-of-jackson-libary-for-JSON.html)
    - [5 Reasons to Use Composition over Inheritance in J...](https://javarevisited.blogspot.com/2013/06/why-favor-composition-over-inheritance-java-oops-design.html)
    - [What is the cost of Spring Framework Certification...](https://javarevisited.blogspot.com/2017/06/what-is-cost-of-spring-framework-certification-training-India-USA.html)
    - [How Volatile in Java works? Example of volatile ke...](https://javarevisited.blogspot.com/2011/06/volatile-keyword-java-example-tutorial.html)
    - [2 Examples to read Zip Files in Java, ZipFile vs Z...](https://javarevisited.blogspot.com/2014/06/2-examples-to-read-zip-files-in-java-zipFile-vs-zipInputStream.html)
    - [How to find CPU and Memory used by Java process in...](https://javarevisited.blogspot.com/2013/06/find-cpu-and-memory-used-by-java-solaris-prstat-command-example.html)
    - [20 Design Patterns and Software Design Interview Q...](https://javarevisited.blogspot.com/2012/06/20-design-pattern-and-software-design.html)
    - [Difference between Inheritance and Composition in ...](https://javarevisited.blogspot.com/2015/06/difference-between-inheritance-and-Composition-in-Java-OOP.html)
  + [►](javascript:void(0))[May](https://javarevisited.blogspot.com/2017/05/)(43)
  + [►](javascript:void(0))[April](https://javarevisited.blogspot.com/2017/04/)(73)
  + [►](javascript:void(0))[March](https://javarevisited.blogspot.com/2017/03/)(97)
  + [►](javascript:void(0))[February](https://javarevisited.blogspot.com/2017/02/)(75)
  + [►](javascript:void(0))[January](https://javarevisited.blogspot.com/2017/01/)(91)
* [►](javascript:void(0))[2016](https://javarevisited.blogspot.com/2016/)(85)

**References**

1. [Oracle's Java Tech Network](http://www.oracle.com/technetwork/java/index.html)
2. [jQuery Documentation](https://jquery.com/)
3. [Microsoft SQL Server Documentation](https://docs.microsoft.com/en-us/sql/sql-server/sql-server-technical-documentation)
4. [Java SE 8 API Documentation](https://docs.oracle.com/javase/8/docs/api/)
5. [Spring Documentation](https://spring.io/docs)
6. [Oracle's JAva Certification](https://education.oracle.com/)
7. [Spring Security 5 Documentation](https://docs.spring.io/spring-security/site/docs/5.0.3.RELEASE/reference/htmlsingle/)

**Pages**

* [Privacy Policy](https://javarevisited.blogspot.com/p/privacy-policy.html)
* [Terms and Conditions](https://javarevisited.blogspot.com/p/last-updated-may-20-2020-please-read.html)

Copyright by Javin Paul 2010-2021. Powered by [Blogger](https://draft.blogger.com/).

**Search This Blog**

Top of Form

|  |  |
| --- | --- |
|  |  |

Bottom of Form

Read more: <https://javarevisited.blogspot.com/2013/06/10-java-exception-and-error-interview-questions-answers-programming.html#ixzz6wMrWNRQ6>

[**JAVA HUNGRY**](https://javahungry.blogspot.com/)

*Java developers tutorials and coding.*

Top of Form

Bottom of Form

* [**HOME**](https://javahungry.blogspot.com/)
* [**STRING**](https://javahungry.blogspot.com/p/string.html)
* [**COLLECTIONS**](https://javahungry.blogspot.com/2017/12/java-collections-framework-tutorial.html)
* [**INTERVIEW**](https://javahungry.blogspot.com/p/interview.html)
* [**EXCEPTION**](https://javahungry.blogspot.com/p/exception.html)
* [**DATA-STRUCTURES**](https://javahungry.blogspot.com/p/data-structures-in-java.html)
* [OOPS](https://javahungry.blogspot.com/p/oops.html)

* [THREADS](https://javahungry.blogspot.com/p/threads.html)

* [BOOKS/COURSES](https://javahungry.blogspot.com/p/books-and-courses.html)

* [JAVA CODING PROGRAM](https://javahungry.blogspot.com/2018/09/70-top-java-interview-programs-with-coding-solutions.html)

* [MORE JAVA TOPICS](https://javahungry.blogspot.com/p/more-java-topics.html)

Top 50 Exception Handling Interview Questions And Answers In Java

In this post, I will be sharing the top 50 exception handling interview questions and answers in java for freshers (0-1 years) and experienced java developers.  
  
I have divided the post into two sections:  
  
1. Face to Face Round Exception Handling Interview Questions and Answers  
  
2. Written/Coding Round Interview Questions  
  
***Face to Face Round Exception Handling Interview Questions and Answers:***  
  
**Q1. What is an Exception in java?**  
  
An Exception is a failure condition that occurs during the execution of a program and disrupts the normal flow of the program. It has to be handled properly, failing which program will be terminated abruptly.  
  
**Q2. How the exceptions are handled in java?**  
  
Exceptions handling can be done using try, catch and finally blocks.  
  
**try :** The code or set of statements that may raise exception should be try block.  
**catch :** This block catches the exceptions thrown in the try block.  
**finally :** This block of code is always executed whether an exception has occurred in the try block or not except in one scenario explained in below question.  
  
**Q3 Is finally block always get executed in the java program?**  
 **This question is very important**. finally block is always executed but there is one scenario when finally block does not execute.  
By using **System.exit(0)** in the try or catch block, results in **finally block does not execute**. The reason is System.exit(0) line terminates the running java virtual machine. Termination leads to no more execution of the program.  
  
**This is the only scenario when finally block fails to execute.**

**public** **class** **JavaHungry**{

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

{

**try**

{

System.out.println("Inside try block ");

/\* After executing below line

jvm terminates the program \*/

System.exit(**0**);

}

**catch** (Exception e)

{

System.out.println("Inside catch block");

}

**finally**

{

System.out.println("Inside finally block");

}

}

}

**Output :**  
Inside try block  
  
**Q4. What are the differences between Error and Exception in java?**  
  
Main differences between Error and Exception are :  
  
a. Errors are caused by the JVM environment in which the application is running. Example: OutOfMemoryError while Exceptions are caused by the application itself. Example: NullPointerException.  
b. Errors can only occur at runtime while Exceptions can occur at compile time or runtime.  
  
You can find more differences between Error and Exception [here](https://javahungry.blogspot.com/2019/10/difference-between-error-and-exception.html).  
  
**Q5. What statements can exist in between try, catch and finally blocks?**  
  
No, try, catch and finally forms a single unit and no other statements should exist in between try, catch and finally blocks.  
  
**Q6. Are we allowed to use only try block without a catch and finally blocks?**  
  
***Prior to Java 7:***  
No, it is not allowed. If used it shows compilation error. The try block must be followed by a catch block or finally block.  
  
***After Java 7 (Correct Answer):***  
Yes, it is possible to have a try block without a catch and finally blocks. The introduction of try-with-resources concept makes it possible.  
The only constraint is resources which we are passing as a parameter in try block must implement [AutoCloseable](https://docs.oracle.com/javase/8/docs/api/java/lang/AutoCloseable.html) interface.

**import** **java.util.\***;

**public** **class** **JavaHungry**{

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

{

/\* After completion of try block,

the Scanner object would be auto closed

as Scanner class implements AutoCloseable

interface. \*/

**try**(Scanner sc = **new** Scanner(System.in))

{

System.out.println(" try without catch/finally block ");

}

}

}

**Output :**  
try without catch/finally block  
  
**Q7. What are Checked and Unchecked exceptions in java?**  
  
Exceptions which are known to the compiler are called Checked exceptions. Checked exceptions are checked at compile-time only.  
  
Unchecked exceptions occur only at run time. Unchecked exceptions are also called as run time exceptions. All subclasses of java.lang.RuntimeException and java.lang.Error is of Unchecked type.  
  
**Q8. What is the difference between Checked and Unchecked exceptions in java?**  
  
This is one of the most popular interview questions for java developers. Make sure this question is in your to-do list before appearing for the interview.  
Main differences between Checked and Unchecked exceptions are :  
  
a. Checked exceptions are checked at compile time while Unchecked exceptions are checked at run time.  
b. Checked exceptions must be handled by try/catch block or throws keyword while Unchecked exceptions are not necessary to handle.  
  
find a detailed explanation [here](https://javahungry.blogspot.com/2019/08/difference-checked-and-unchecked-exception.html).  
  
**Q9. What is the difference between final, finally and finalize in java?**  
  
**final keyword:**  
By declaring a variable as final, the value of final variable cannot be changed.  
By declaring a method as final, method cannot be overridden.  
By declaring a class as final, class cannot be extended.  
  
**finally:**  
Used after try or try-catch block, will get executed after the try and catch blocks without considering whether an exception is thrown or not.  
  
**finalize:**  
Finalize method is the method that Garbage Collector always calls just before the deletion/destroying the object which is no longer in use in the code.  
  
**Q10 What is try-with-resources concept in java? How it differs from an ordinary try statement?**  
  
According to [Java docs](https://docs.oracle.com/javase/tutorial/essential/exceptions/tryResourceClose.html), try-with-resources statement is a try statement that declares one or more resources. It ensures that each resource is closed at the end of the statement.  
  
try-with-resources statement can have catch or finally block similar to ordinary try statement. In a try-with-resources statement, JVM makes sure catch or finally block is run after the resources declared have been closed.  
  
**Q11. What is RuntimeException in java. Give example?**  
  
The exceptions which occur at runtime are called as RuntimeException. These exceptions are unknown to the compiler. All subclasses of java.lang.RuntimeException are RuntimeExceptions.  
  
For example:  
NumberFormatException, NullPointerException, ClassCastException, ArrayIndexOutOfBoundException  etc.  
  
**Q12. What is the difference between ClassNotFoundException and NoClassDefFoundError in java?**  
  
This question is important because very few Java developers are aware of the difference between ClassNotFoundException and NoClassDefFoundError.  
  
**ClassNotFoundException:**  
An exception that occurs when you try to load a class at run time using Class.forName() or loadClass() methods and mentioned classes are not found in the classpath is called ClassNotFoundException.  
  
**NoClassDefFoundError:**  
An exception that occurs when a particular class is present at compile-time but was missing at run time is called NoClassDefFoundError.  
  
find a detailed explanation [here](https://javahungry.blogspot.com/2016/04/4-difference-between.html).  
  
**Q13. Can we throw an exception manually/explicitly?**  
  
Yes, using ***throw*** keyword we can throw an exception manually.  
  
**Syntax:**

**throw** InstanceOfThrowableType;

**For example:**

**public** **class** **JavaHungry**{

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

{

**try**

{

// Creating an object of ArithmeticException

ArithmeticException ae = **new** ArithmeticException();

//Manually throwing ArithmeticException

**throw** ae;

}

**catch** (ArithmeticException e)

{

System.out.println("Caught the manually thrown Exception");

}

}

}

**Output:**  
Caught the manually thrown Exception  
  
  
**Q14. Does catch block rethrow an exception in java?**  
  
Yes, catch block can rethrow an exception using throw keyword. It is called re-throwing an exception.  
  
For example :

**public** **class** **JavaHungry**{

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

{

**try**

{

// Creating an object of ArithmeticException

ArithmeticException ae = **new** ArithmeticException();

//Manually throwing ArithmeticException

**throw** ae;

}

**catch** (ArithmeticException e)

{

System.out.println("Rethrowing the caught exception below ");

//Rethrowing ArithmeticException

**throw** e;

}

}

}

**Output :**  
Rethrowing the caught exception below  
  
Exception in thread "main" java.lang.ArithmeticException  
    at JavaHungry.main(JavaHungry.java:7)  
  
**Q15. What is the use of throws keyword in java?**  
  
throws keyword is used to declare an exception. You can find a detailed explanation [here](https://javahungry.blogspot.com/2019/09/difference-between-throw-and-throws.html).  
  
**Q16. Why it is always recommended that clean up activities like closing the DB connections and I/O resources to keep inside a finally block?**  
  
finally block will always be executed except one scenario as discussed above in Q3. By ensuring the cleanup operations in finally block, you will assure that those operations will be always executed irrespective of whether an exception has occurred or not.  
  
**Q17. What is OutOfMemoryError in Exception Handling?**  
  
OutOfMemoryError is the subclass of java.lang.Error. It occurs when JVM runs out of memory.  
  
**Q18. What is ClassCastException in Exception Handling?**  
  
RunTimeException which occurs when JVM not able to cast an object of one type to another type is called ClassCastException.  
  
**Q19. What is the difference between throws and throw in java?**  
  
This is one of the most frequently asked interview questions for java developers.  
Main differences between throws and throw are :  
  
a. throws keyword is used when writing methods, to declare that the method in question throws the specified (checked) exception.  
throw is used when an instruction is to explicitly throw the exception.  
b.  throws is used with a method signature while the throw is used inside a method.  
  
You can find a detailed explanation of the difference between throw and throws in java [here](https://javahungry.blogspot.com/2019/09/difference-between-throw-and-throws.html).  
  
**Q20. What is StackOverflowError in Exception Handling?**  
  
StackOverflowError is thrown by the JVM when stack overflows in a program.  
  
**Q21. Which class is the root class for all types of errors and exceptions in Exception Hierarchy?**  
  
java.lang.Throwable is the superclass for all types of errors and exceptions in java.  
  
**Q22. When do we use printStackTrace() method in java?**  
  
printStackTrace() function is used to print the detailed information about the exception thrown by the try/catch block.  
  
**Q23. Give some examples of Checked exceptions?**  
  
SQLException, ClassNotFoundException, IOException  
  
**Q24. Give some examples of Unchecked exceptions?**  
  
ArrayIndexOutOfBoundsException, NullPointerException, NumberFormatException  
  
**Q25. List the Methods in the Throwable class?**  
  
Below are the important methods of Throwable class:  
  
•    getMessage()  
•    Throwable getCause()  
•    toString()  
•    printStackTrace()  
•    StackTraceElement [] getStackTrace()  
  
**Q26. What is a SQLException in Exception Handling?**  
  
An exception that provides information related to database access error or other errors is called SQL Exception.  
  
**Q27. What is NumberFormatException in java?**  
  
NumberFormatException is thrown when you try to convert a String into a number.  
  
**Q28. What is ArrayIndexOutOfBoundsException in java?**  
  
ArrayIndexOutOfBoundsException arises while trying to access an index of the array that does not exist or out of the bound of this array.  
  
**Q29. What will happen if an exception is thrown by the main method?**  
  
When an exception is thrown by the main method then JVM terminates the program. As a result, you will find the exception message and stack trace in the system console.  
  
**Q30. Is it legal to have an empty catch block?**  
  
Yes, we can have an empty catch block in java but it is not the best practice. If an exception is caught by the empty catch block, then we do not have any information about the exception occurred. You should provide at least the logging statement to log the exception details.  
  
**Q31. What are the keywords in Java for Exception Handling?**  
  
throw, throws, try, catch and finally  
  
**Q32. List some important methods of Java Exception class?**  
  
a. printStackTrace()  
b. toString()  
c. getMessage()  
  
**Q33. What are the advantages of using Exceptions in your programs?**  
  
According to [Java doc](https://docs.oracle.com/javase/tutorial/essential/exceptions/advantages.html),  
  
a. Separating "regular" code from error handling code.  
b. The ability to propagate errors reporting up the call stack of methods.  
c. Differentiating and grouping error types.     
  
***Written/Coding round Interview Questions:***  
  
Please go through the below link before attempting questions 46-50.  
  
**Q34-45**  **[Programs on Exception Handling in Java with Answers.](https://javahungry.blogspot.com/2019/09/programs-on-exception-handling-in-java.html" \t "_blank)**  
  
**Q46. What is unreachable catch block error?**  
   
When there is more than one catch block, then, the order of catch blocks must be from most specific to most general ones. In other words, subclasses of Exception must come first and superclasses later. If you try to keep superclasses first and subclasses later, the compiler will stop you and show unreachable catch block error.  
  
**Q47. Can we provide the statements after finally block if the control is returning from the finally block itself?**  
  
No, because control is returning from the finally block itself so it shows unreachable code error.  
  
**Q48. Does finally block get executed if either try or catch blocks are returning the control?**  
  
Yes, finally block will be always executed no matter whether try or catch blocks are returning the control or not. There is one scenario where finally block does not execute, for more information check out Q3.  
  
**Q49.  Suppose there is a catch block corresponding to a try block with three statements – statement1, statement2, and statement3. Assume that exception is thrown in statement2. Does statement3 get executed?**  
  
Statement3 will not get executed because once a try block throws an exception, remaining statements will not be executed.  
  
**Q50. What will happen if we override a superclass method which is throwing an unchecked exception with a checked exception in the subclass?**  
  
If the overridden method throwing an unchecked exception, then the subclass overriding method must have the same exception or any other unchecked exceptions. If you are trying to throw checked exception with subclass overriding method then it will give a compilation error.  
  
**Rule of thumb :**An overriding method (subclass method) can not throw a broader exception than an overridden method (superclass method).  
  
That's all for today, please mention in comments if you have any questions related to exception handling in java interview questions and answers for freshers and experienced.

**About The Author**

Subham Mittal has worked in Oracle for 3 years .  
For more java articles ,Click here to [Subscribe JavaHungry](https://javahungry.blogspot.com/p/subscribe-for-our-newsletter.html)

You might also like:

[[](https://javahungry.blogspot.com/2019/09/difference-between-throw-and-throws.html)](https://javahungry.blogspot.com/2019/09/difference-between-throw-and-throws.html)

[5 Difference Between throw and throws in Java with Examples](https://javahungry.blogspot.com/2019/09/difference-between-throw-and-throws.html)

[[](https://javahungry.blogspot.com/2019/09/programs-on-exception-handling-in-java.html)](https://javahungry.blogspot.com/2019/09/programs-on-exception-handling-in-java.html)

[Programs on Exception Handling in Java for Interview](https://javahungry.blogspot.com/2019/09/programs-on-exception-handling-in-java.html)

[[https://1.bp.blogspot.com/-0dtxEj7ZfOw/XWJPAQZbZMI/AAAAAAAABD0/HA56KWO3BDcb-Wp4R9E4lTkw1nZVnG_LQCLcBGAs/w120-h1-p/checked%2Band%2Bunchecked%2Bexception%2Bin%2Bjava.png](https://javahungry.blogspot.com/2019/08/difference-checked-and-unchecked-exception.html)](https://javahungry.blogspot.com/2019/08/difference-checked-and-unchecked-exception.html)

[Difference Between Checked and Unchecked Exception with Examples](https://javahungry.blogspot.com/2019/08/difference-checked-and-unchecked-exception.html)

[[https://2.bp.blogspot.com/-6RZcdJk412I/Vb2JWSumF1I/AAAAAAAAApU/wfMTdNwApXo/w120-h1-p/NoClassDefFoundError.jpg](https://javahungry.blogspot.com/2016/04/4-difference-between.html)](https://javahungry.blogspot.com/2016/04/4-difference-between.html)

[4 Difference between NoClassDefFoundError and ClassNotFoundException](https://javahungry.blogspot.com/2016/04/4-difference-between.html)

[[https://3.bp.blogspot.com/-VGEVDcc04ZU/UcFqIfZccXI/AAAAAAAAAKs/UQX8k5Qlr_g/w120-h1-p/Interrupted+Exception.jpg](https://javahungry.blogspot.com/2013/06/how-to-create-manually-thrown-interrupted-exception-in-threads-example-java-program-code.html)](https://javahungry.blogspot.com/2013/06/how-to-create-manually-thrown-interrupted-exception-in-threads-example-java-program-code.html)

[How to manually generate Interrupted Exception in Threads : Java Program Code](https://javahungry.blogspot.com/2013/06/how-to-create-manually-thrown-interrupted-exception-in-threads-example-java-program-code.html)

[[](https://javahungry.blogspot.com/2021/05/nested-try-catch.html)](https://javahungry.blogspot.com/2021/05/nested-try-catch.html)

[Nested try catch block in Java](https://javahungry.blogspot.com/2021/05/nested-try-catch.html)

[[](https://javahungry.blogspot.com/2021/05/java-finally-block.html)](https://javahungry.blogspot.com/2021/05/java-finally-block.html)

[finally block in Java [Guide]](https://javahungry.blogspot.com/2021/05/java-finally-block.html)

[[](https://javahungry.blogspot.com/2021/05/java-throws.html)](https://javahungry.blogspot.com/2021/05/java-throws.html)

[throws keyword in java with examples](https://javahungry.blogspot.com/2021/05/java-throws.html)

|  |  |  |
| --- | --- | --- |
|  |  |  |

[Login](https://javahungry.blogspot.com/2019/11/exception-handling-interview-questions-and-answers.html)

https://gravatar.com/avatar/undefined?d=https://embed.widgetpack.com/widget/img/avatar.png&r=PG&s=36



Comments

[Sort by Best](https://javahungry.blogspot.com/2019/11/exception-handling-interview-questions-and-answers.html)

Be the first to comment!

[ADD WIDGETPACK TO YOUR SITE](https://widgetpack.com/comment-system)

[POWERED BY WIDGET PACK™](https://widgetpack.com/)

WHATS HOT

* [Difference between Arraylist and Vector : Core Java Interview Collection Question](https://javahungry.blogspot.com/2013/12/difference-between-arraylist-and-vector-in-java-collection-interview-question.html)

* [Best Books for Learning Java](https://javahungry.blogspot.com/2014/02/best-books-for-learning-java-must-read.html)

* [Amazon Interview Question : First Non repeated character in String](https://javahungry.blogspot.com/2013/12/first-non-repeated-character-in-string-java-program-code-example.html)

* [Count total number of times each alphabet appears in the string java program code with example](https://javahungry.blogspot.com/2013/10/count-total-number-of-times-each-alphabet-appears-in-the-string.html)

* [Java 8 new features : Lambda expressions , optional class , Defender methods with examples](https://javahungry.blogspot.com/2013/10/oracle-java-8-new-features-lambda-expressions-optional-defender-methods-example.html)

[Subscribe for Our Newsletter](https://javahungry.blogspot.com/p/subscribe-for-our-newsletter.html)

POPULAR POSTS

* [Java Interview Questions](https://javahungry.blogspot.com/2013/06/top-25-most-frequently-asked-core-java.html)

* [Top 50 Java Collections Interview Questions and Answers](https://javahungry.blogspot.com/2015/05/50-java-collections-interview-questions-and-answers.html)

* [Web Services Interview Questions](https://javahungry.blogspot.com/2015/07/web-services-interview-questions-and-answers.html)

* [Java Multithreading Interview Questions and Answers](https://javahungry.blogspot.com/2017/10/java-multithreading-interview-questions-and-answers.html)

* [Java Interview Programs](https://javahungry.blogspot.com/2018/09/70-top-java-interview-programs-with-coding-solutions.html)

[Return to Top ↑](https://javahungry.blogspot.com/2019/11/exception-handling-interview-questions-and-answers.html)

[Affiliate-Disclaimer](https://javahungry.blogspot.com/p/affilliate-disclaimer.html)[Privacy](https://javahungry.blogspot.com/p/this-privacy-policy-governs-manner-in.html)[Java Hungry](https://javahungry.blogspot.com/) Copyright © 2021 · All Rights Reserved. Designed by [studiopress](https://www.studiopress.com/)



* [Home](https://java2blog.com/)
* [Core Java](https://java2blog.com/core-java-tutorial-for-beginners-experienced/)
* [Tutorials](https://java2blog.com/java-exception-handling-interview-questions/)
* [Categories](https://java2blog.com/java-exception-handling-interview-questions/)
* [Tech Jobs](https://lcsjobs.io/)

search

[Tech Jobs](https://lcsjobs.io/)

* [keyboard\_arrow\_leftPrevious](https://java2blog.com/java-multithreading-interview-questions-and-answers/)

[Nextkeyboard\_arrow\_right](https://java2blog.com/java-serialization-interview-questions-and-answers/)

search

**Related Posts**

* [Java try with resources](https://java2blog.com/java-try-with-resources/)
* [Java interview questions](https://java2blog.com/java-interview-questions/)
* [SOLID Principles in Java](https://java2blog.com/solid-principles-java/)
* [Java interview questions for 2 years experience](https://java2blog.com/java-interview-questions-for-2-years-experience/)
* [Java interview questions for 5 years experience](https://java2blog.com/java-interview-questions-for-5-years-experience/)
* [Java technical interview questions](https://java2blog.com/java-technical-interview-questions/)
* [Top 15 Spring Boot Interview questions](https://java2blog.com/spring-boot-interview-questions/)
* [Spring interview questions and answers for experienced](https://java2blog.com/spring-interview-questions-and-answers/)
* [Restful web services interview questions](https://java2blog.com/restful-web-services-interview-questions/)
* [Top 20 Web services interview questions](https://java2blog.com/web-services-interview-questions/)

[**Home**](https://java2blog.com/) > [**Interview**](https://java2blog.com/category/interview/) > [**Interview Questions**](https://java2blog.com/category/interview/interview-questions/) > Java Exception Handling Interview Questions And Answers

**Java Exception Handling Interview Questions And Answers**

Exceptional handling is one of the most important topics in core java.  
Here is list of questions that may be asked on Exceptional handling.

**Question 1: What is Exception ?**

**Answer:**

* Java doc says “ An exception is an event, which occurs during the execution of a program, that disrupts the normal flow of the program’s instructions.”
* The term “exception” means “exceptional condition” and is an occurrence that changes the normal program flow.
* A bunch of things can lead to exceptions, including hardware failures, resource exhaustion, and good old bugs.
* When an exception event occurs in Java , an exception is said to be “thrown”.
* The code that is responsible for doing something about the exception is called an “exception handler” and it “catches” the thrown exception.
* Every Exception will be thrown at runtime.

**Question 2: How can you handle exception in java?**

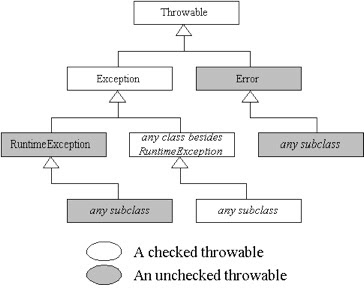
**Answer:**

**try-catch-finally**blocks are used to handle exception in java.

**try :**This block has code which can throw exception.

**catch :**This block is used to handle appropriate exception.

**finally :**This block is used to write any clean up code irrespective of whether any exception occurred or not.

**Question 3: Explain the Exception hierarchy.**  
[](https://4.bp.blogspot.com/-Ku-ly97GgvM/WAdHZW5RqII/AAAAAAAAEzM/DIuJ1OPSXNECi1t8doKhKqukAnQeGLlAQCPcB/s1600/ExceptionHierarchy.jpg)

**Question 4: Difference between checked exception, unchecked exceptionand and errors**

|  |  |  |  |
| --- | --- | --- | --- |
| **Parameter** | **Checked Exception** | **Unchecked Exception** | **Error** |
| How to recognise | sub class of Exception class | sub class of RuntimeException class | sub class of Error class |
| Good to  catch | Yes | Yes | No |
| Is Program required to handle or declared | Yes | No | No |
| Thrown by | Programatically | JVM | JVM |
| Recoverable | Yes | Yes | No |
| Example | IOException FileNotFoundException etc | NullPointerException ClassCastException etc | StackOverFlowError OutOfMemoryError etc |

**Question 5: Can we have try without catch block in java ?**

**Answer:**  
Yes, we can have try without catch block by using finally block. You can use try with finally. As you know finally block always executes even if you have exception or return statement in try block except in case of System.exit()..

**Question 6: What is RunTime exception in java?**

**Answer:**

RunTime exception is the exception which is thrown at run time. These exceptions occur due to programmatic errors and need to be corrected. Compiler is not aware of any such exception.

**Example:**

If you are working in java for quite some time, you might have got NullPointerException.

Let’s create a simple method which calculates if String’s length is odd or even.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9 | public static String checkOddEvenLenghtString(String str) {  if (str.length() % 2 == 0) {  return "Even";  } else {  return "Odd";  }  } |

If you call above method with checkOddEvenLenghtString(null), then it will throw NullPointerException as we are calling length method on null object.

**Question 7: What is checked exception or compile time exception?**

**Answer:**  
Checked exceptions are those exceptions which are checked at compile. If you do not handle them , you will get compilation error.It forces you to handle this exception in try-catch block.

**Question 8: Can you put other statements between try,catch and finally block?**

**Answer:**  
No, You can not put any statements between try, catch and finally block.

**Question 9: How do you create custom exception in java?**

**Answer:**

You just need to extend Exception class to create custom exception.If yo want to create Unchecked Exception, you need to extend RuntimeException.

Let’s understand this with example.You have list of employee and if anyemployee’s age in list of employee is less than 18, then you need to throw invalidAgeException(Our custom exception).  
**Example:**  
Create InvalidAgeException.java as below

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11 | package org.arpit.java2blog;    public class InvalidAgeException extends Exception{    InvalidAgeException(String message)  {    super(message);  }  } |

Create POJO class called Country.java

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29 | package org.arpit.java2blog;    public class Employee {    private String name;  private int age;    Country(String name, int age ){  this.name = name;  this.age=age;  }    public String toString() {     return name;    }  public String getName() {    return name;  }  public void setName(String name) {    this.name = name;  }     public int getAge() {    return age;  }  public void setName(int age) {    this.age = age;  }  } |

Let’s create EmployeeCheckMain.java. This class will have main method.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43 | package com.arpit.java2blog;    import java.util.ArrayList;  import java.util.List;    public class EmployeeCheckMain {    public static void main(String args[]) {    List<Employee> employees = new ArrayList<>();    Employee e1 = new Employee("Adam",20);    Employee e2 = new Employee("Brain",19);    Employee e3 = new Employee("Stuart",22);    Employee e4 = new Employee("Paul",21);    employees.add(e1);    employees.add(e2);    employees.add(e3);    employees.add(e4);    boolean safe;    try {     safe = checkListOfEmployees(employees);     if (safe)      System.out.println("We don't have any employee with less than 18");     Employee e5 = new Employee("Dev",17);     employees.add(e5);     checkListOfEmployees(employees);    } catch (InvalidAgeException e) {     e.printStackTrace();    }    }    public static boolean checkListOfEmployees(List<Employee> employees) throws InvalidAgeException {    for (int i = 0; i < employees.size(); i++) {     Employee employee = employees.get(i);     if (employee.getAge() < 18) {      throw new InvalidAgeException("Employee'sAge is less than 18");     }    }    return true;  }  } |

When you run above program, you will get following output:

|  |  |
| --- | --- |
| 1  2  3  4  5  6 | We don't have any employee with less than 18  com.arpit.java2blog.InvalidAgeException      at com.arpit.java2blog.EmployeeCheckMain.checkListOfEmployees(EmployeeCheckMain.java:36)      at com.arpit.java2blog.EmployeeCheckMain.main(EmployeeCheckMain.java:25) |

As you can see, if we have any employee whose age is less 18,we are throwing InvalidAgeException.

**Question 10: Is there any order in which catch block should be written?**

**Answer:**  
Yes, most specific exception should be written first and then generic one.  
**For example:**  
below code will give you compilation error.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17 | public static int exceptionTest()    {       try{     File f=new File("/usr/abc.txt");     }     catch(Exception e)     {       }     catch(IOException e) // Compilation error     {       }    } |

It will give you compilation error with below message. “Unreachable catch block for IOException. It is already handled by the catch block for Exception”.

**Question 11:  Difference between throw and throws keyword?**

**Answer:**  
Please follow [Difference between throw and throws keyword](https://www.java2blog.com/?p=13) to see the difference.

**Question 12 : 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  21  22  23  24 | public class ExceptionTest {    public static void main(String[] args) {     System.out.println(exceptionTest());    }    public static int exceptionTest()    {     int i=6;     try{      return i;     }     catch(Exception e)     {      i=10;       }     finally     {      System.out.println("In finally block");     }     return i;    }  } |

Output:

|  |  |
| --- | --- |
| 1  2  3  4 | In finally block  6 |

**Explanation:**  
If you notice we have return statement in try block, so before returning from exceptionTest() method, finally block will be executed. When you have return statement in try block, JVM will take note of value of i and this value will be returned by exceptionTest method.

**Question 13 : 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  21  22  23  24  25 | public class ExceptionTest {    public static void main(String[] args) {     System.out.println(exceptionTest());    }    public static int exceptionTest()    {     int i=6;     try{      throw new NullPointerException();     }     catch(Exception e)     {      i=10;      return i;     }     finally     {      i=20;      System.out.println("In finally block");     }      }  } |

Output:

|  |  |
| --- | --- |
| 1  2  3  4 | In finally block  10 |

**Explanation:**  
Flow of the program will be as below.

* Value of variable i will be set to 6.
* NullPointerException will be thrown from try block.
* Flow will go to catch block and value of i will be set to 10. JVM will make note of value of i and this will be returned by exceptionTest method.
* Before returning from exceptionTest method, finally block will be executed and “In finally block” will be printed on console.
* In the end, return value of exceptionTest method will be 10.

**Question 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  21  22  23  24  25  26 | public class ExceptionTest {    public static void main(String[] args) {     System.out.println(exceptionTest());    }    public static int exceptionTest()    {     int i=6;     try{      throw new NullPointerException();     }     catch(NullPointerException e)     {      i=10;      throw e;     }     finally     {      i=20;      System.out.println("In finally block");      return i;     }      }  } |

Output:

|  |  |
| --- | --- |
| 1  2  3  4 | In finally block  20 |

**Explanation:**  
Flow of the program will be as below.

* Value of variable i will be set to 6.
* NullPointerException will be thrown from try block.
* Flow will go to catch block and value of i will be set to 10. We are throwing NullPointerException from catch block.
* finally will get excuted and value of i will be set to 20.”In finally block” will be printed on console.
* In the end, return value of exceptionTest method will be 20.
* If you notice here, return statement from finally block actually suppressed the NullPointerException.

**Question 15: 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  21  22 | public class ExceptionTest {    public static void main(String[] args) {     System.out.println(exceptionTest());    }    public static int exceptionTest()    {     int i=6;     try{      i=50;      return i;     }     finally     {      i=20;      System.out.println("In finally block");      return i;     }      }  } |

Output:

|  |  |
| --- | --- |
| 1  2  3  4 | In finally block  20 |

**Explanation:**  
Flow of the program will be as below.

* Value of variable i will be set to 6.
* Value of variable i will be set to 10 and JVM will make note of return value of i as 10.
* finally will get excuted and value of i will be set to 20.”In finally block” will be printed on console.
* In the end, return value of exceptionTest method will be 20. It will override value returned by try block.

You may also like:

* [Core java interview questions](https://www.java2blog.com/?p=175)
* [Java Collections interview questions](https://www.java2blog.com/?p=250)
* [Java String interview questions](https://www.java2blog.com/?p=173)
* [OOPs interview questions in java](https://www.java2blog.com/?p=6)
* [Java Multithreading interview questions](https://www.java2blog.com/?p=9)
* [Java Serialization interview questions in java](https://www.java2blog.com/?p=14)
* [Method overloading and overriding interview questions](https://www.java2blog.com/?p=8)
* [web services interview questions](https://www.java2blog.com/?p=183)
* [restful web services interview questions](https://www.java2blog.com/?p=184)
* [Data structure and algorithm Interview Questions](https://www.java2blog.com/?p=101)
* [Spring interview questions](https://www.java2blog.com/?p=172)
* [Hibernate interview questions](https://www.java2blog.com/?p=180)

**Share this**

* [**Facebook**](https://www.facebook.com/sharer.php?caption=Java%20Exception%20Handling%20Interview%20Questions%20And%20Answers&u=https://java2blog.com/java-exception-handling-interview-questions/)
* [**Twitter**](https://twitter.com/intent/tweet?text=Java%20Exception%20Handling%20Interview%20Questions%20And%20Answers&url=https://java2blog.com/java-exception-handling-interview-questions/)
* [**LinkedIn**](https://www.linkedin.com/shareArticle?mini=true&url=https://java2blog.com/java-exception-handling-interview-questions/)
* [**Pinterest**](https://in.pinterest.com/pin/create/button/?url=https://java2blog.com/java-exception-handling-interview-questions/&media=https://java2blog.com/wp-content/uploads/2021/01/Exception-Handling-Interview-Questions.jpg&description=Java%20Exception%20Handling%20Interview%20Questions%20And%20Answers)
* [**Whatsapp**](https://api.whatsapp.com/send?text=https://java2blog.com/java-exception-handling-interview-questions/)

**[Prev](https://java2blog.com/java-multithreading-interview-questions-and-answers/)**

[Java Multithreading and Concurrency Interview Questions](https://java2blog.com/java-multithreading-interview-questions-and-answers/)

**[Next](https://java2blog.com/java-serialization-interview-questions-and-answers/)**

[Java Serialization interview questions and answers](https://java2blog.com/java-serialization-interview-questions-and-answers/)

**Author**



[Arpit Mandliya](https://java2blog.com/author/arpitmandliya/)

Follow Author

**Related Posts**

* 

[Hibernate](https://java2blog.com/category/hibernate/)[Interview Questions](https://java2blog.com/category/interview/interview-questions/)[Interview](https://java2blog.com/tag/interview/)

16 April

[Top 40+ Hibernate interview questions and answers](https://java2blog.com/hibernate-interview-questions-and-answers/)

Hibernate is most popular Object relational Mapping (ORM) framework and if you have used hibernate and has good experience on it, then you may face good hibernate interview questions. I am sharing some important hibernate interview questions here. 1. What is ORM? ORM stands for Object Relational mapping. It is programming paradigm which is used […]

[Read More](https://java2blog.com/hibernate-interview-questions-and-answers/)

* 

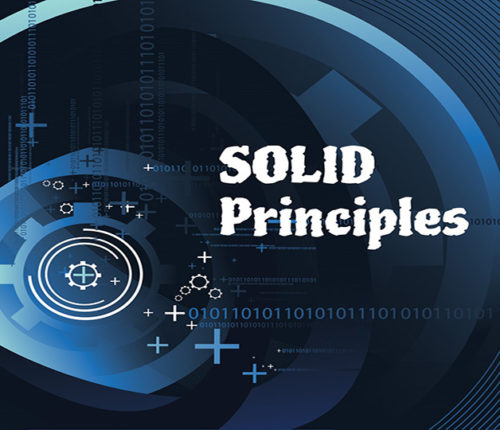
[Interview Questions](https://java2blog.com/category/interview/interview-questions/)[Core Java interview](https://java2blog.com/tag/core-java-interview/)[Interview](https://java2blog.com/tag/interview/)

22 May

[Java interview questions](https://java2blog.com/java-interview-questions/)

I have written a lot of java interview questions and this is index posts for all java interview questions. I will keep adding more interview questions to this post. Core Java Core java interview questions This list includes top 50 core java interview questions. Whether you are fresher or experienced programmer, this interview questions will […]

[Read More](https://java2blog.com/java-interview-questions/)

* 

[Interview Questions](https://java2blog.com/category/interview/interview-questions/)[SOLID](https://java2blog.com/category/core-java/solid/)[Core Java interview](https://java2blog.com/tag/core-java-interview/)[Interview](https://java2blog.com/tag/interview/)

09 November

[SOLID Principles in Java](https://java2blog.com/solid-principles-java/)

In this post, we will see 5 SOLID Principles in Java. Robert C. Martin gave five objected oriented design principles, and the acronym S.O.L.I.D is used for it. Each letter of this acronym talks about principles in Java. When you use all the principles of S.O.L.I.D in a combined manner, it becomes easier for you […]

[Read More](https://java2blog.com/solid-principles-java/)

* 

[Interview Questions](https://java2blog.com/category/interview/interview-questions/)[Core Java interview](https://java2blog.com/tag/core-java-interview/)[Interview](https://java2blog.com/tag/interview/)

08 November

[Java interview questions for 2 years experience](https://java2blog.com/java-interview-questions-for-2-years-experience/)

In this post, we are going to see Java interview questions for 2 to 3 years experienced. These are the most asked interview questions for freshers(0-3 years experienced). This question list will help you to crack java interview. I have already shared detailed answer over here before, you might find it helpful as well. I […]

[Read More](https://java2blog.com/java-interview-questions-for-2-years-experience/)

* 

[Interview Questions](https://java2blog.com/category/interview/interview-questions/)[Core Java interview](https://java2blog.com/tag/core-java-interview/)[Interview](https://java2blog.com/tag/interview/)

08 November

[Java interview questions for 5 years experience](https://java2blog.com/java-interview-questions-for-5-years-experience/)

In this post, we will see Java interview questions for 5 to 6 years experience. When you have 5 years of experience as java developer, you need to have a good understanding of collections, multithreading concepts. If you are looking for the below queries then this post will help you as well. Java interview questions […]

[Read More](https://java2blog.com/java-interview-questions-for-5-years-experience/)

* 

[Interview Questions](https://java2blog.com/category/interview/interview-questions/)[Core Java interview](https://java2blog.com/tag/core-java-interview/)[Interview](https://java2blog.com/tag/interview/)

11 September

[Java technical interview questions](https://java2blog.com/java-technical-interview-questions/)

In this post, we will see top Java technical interview asked in the Core Java interview.I have covered a lot of interview question on Java programming. Here is list of other java interview question you might find useful. Core java interview questions Java Collections interview questions Java String interview questions OOPs interview questions in java […]

[Read More](https://java2blog.com/java-technical-interview-questions/)

**Leave a Reply**

Top of Form

Your email address will not be published. Required fields are marked \*





 Save my name, email, and website in this browser for the next time I comment.



Bottom of Form

**Subscribe to our newletter**

Get quality tutorials to your inbox. Subscribe now.

Top of Form


Bottom of Form

Categories

* [Core java](https://java2blog.com/category/core-java/)
* [Interview](https://java2blog.com/category/interview/)
* [Algorithm](https://java2blog.com/category/algorithm/)
* [Python](https://java2blog.com/category/python/)
* [Spring Frameworks](https://java2blog.com/category/spring-frameworks/)

Popular Posts

* [Core Java tutorial](https://java2blog.com/core-java-tutorial-for-beginners-experienced/)
* [Spring Tutorial](https://java2blog.com/introduction-to-spring-framework/)
* [Spring Boot tutorial](https://java2blog.com/spring-boot-tutorial/)
* [Java 8 Stream](https://java2blog.com/java-8-stream/)
* [Garbage Collection in java](https://java2blog.com/garbage-collection-java/)
* [Top 100+ Java coding interview questions](https://java2blog.com/java-coding-interview-questions/)

**Let’s be Friends**

[Facebook](https://www.facebook.com/java2blog" \t "_blank)

[Twitter](https://www.twitter.com/java2blog" \t "_blank)

[LinkedIn](https://www.linkedin.com/in/arpit-mandliya-b325694b" \t "_blank)

© 2020 Java2Blog

[Skip to content](https://www.scientecheasy.com/2020/09/exception-handling-interview-questions.html/#content)

[](https://www.scientecheasy.com/)

* [**Home**](https://www.scientecheasy.com/)
* [**Core Java**](https://www.scientecheasy.com/2018/07/basic-core-java-tutorial-point.html/)
* [**String**](https://www.scientecheasy.com/2019/04/java-string-tutorial.html/)
* [**Collections**](https://www.scientecheasy.com/2018/10/collections-in-java-with-examples.html/)
* [**Selenium**](https://www.scientecheasy.com/2019/09/selenium-webdriver-tutorial.html/)
* Interview
* [**Thread**](https://www.scientecheasy.com/2020/08/java-thread-tutorial.html/)
* [**Exception Handling**](https://www.scientecheasy.com/2020/08/exception-handling-in-java.html/)
* [**Blog**](https://www.scientecheasy.com/blog/)

**65 Java Exception Handling Interview Questions Answers**

Here, we have listed the most important exception handling interview questions in Java with the best possible answers. These interview questions are generally asked in Java interviews by the interviewer.

If you nicely prepare answers to these exception handling interview questions in Java then definitely, you can easily answer questions asked by the interviewer.

**Java Exception Handling Interview Questions and Answers**

**1. What is Exception in Java?**

 An exception is an error that affects the normal execution of program. If an exception is not handled, the program abruptly terminates.

**2. What is the super or base class of all exceptions in Java?**

 Exception is the superclass of all exceptions in Java.

**3. What is the superclass for error and exception classes in Java?**

 Throwable is the superclass for error and exception classes in Java.

**4. What are the types of exceptions in Java API?**

 There are two types of exceptions available in Java API. They are:

a. **Predefined Exceptions (Built-in-Exceptions):** Predefined exceptions are those exceptions that are already defined by Java system. These exceptions are also called built-in-exceptions.

The built-in-exception is further divided into two categories: checked and unchecked exceptions.

b. **Custom Exceptions:** Custom exceptions are those exceptions that are created by users or programmers according to their own needs. The custom exceptions are also called user-defined exceptions that are created by extending the exception class.

**5. Why an exception occurs in the program?**

 There can be many reasons that might generate an exception in a Java program.

* Opening a non-existing file in your program.
* Reading a file from a disk but the file does exist there.
* Writing data to a disk but the disk is full or unformatted.
* When the program asks for user input and the user enters invalid data.
* When a user attempts to divide an integer value by zero, an exception occurs.
* When a data stream is in an invalid format, etc.

**6. What is Exception handling in Java?**

 The mechanism of handling unexpected errors in a java program is called exception handling. It is a powerful mechanism to handle runtime errors, ClassNotFoundException, FileNotFoundException, IOException, etc. so that the normal execution flow of the program can be maintained.

**7. What is exception handler in Java?**

 The code that catches the exception thrown by JVM is called exception handler in Java. In other words, an exception handler is a block of code that performs an action when an exception occurs in a program.

**8. What is the advantage of using exception handling in Java?**

 There are several advantages of using exception handling in java. They are:

* The main advantage of exception handling technique is to maintain the normal flow of the program.
* It provides flexibility in handling situations of errors.
* It allows us to define a user-friendly message to handle the exception.
* It helps to separate “Error-Handling code” from “Regular code.”

**9. Which of the following statements will throw an exception?**  
a. System.out.println(1/0);  
b. System.out.println(2.0/0);

 a will throw an exception named ArithmeticException. The second statement will give output infinity.

**10. Which exception may be thrown if the given code is executed?**

a.



1

**public** **class** Test

2

{

3

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

4

{

5

int[] list = **new** int[4];

6

System.out.println(list[4]);

7

}

8

}

 ArrayIndexOutOfBoundsException will be thrown if the above code is executed.

b.



1

**public** **class** Test

2

{

3

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

4

{

5

int a = 20;

6

int b = 30;

7

int c = 10;

8

int x = (a \* b)/(a - b + c);

9

System.out.println("Result: " +x);

10

}

11

}

12

​

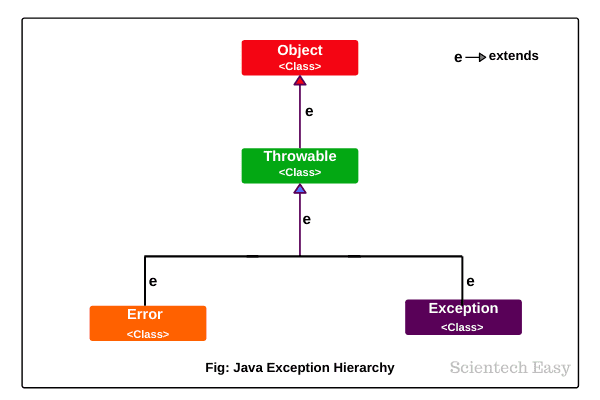
13

​

 ArithmeticException will be thrown if code is executed.

**11. Explain the Java exception hierarchy.**

 The hierarchy of exception in Java is shown in the below diagram.

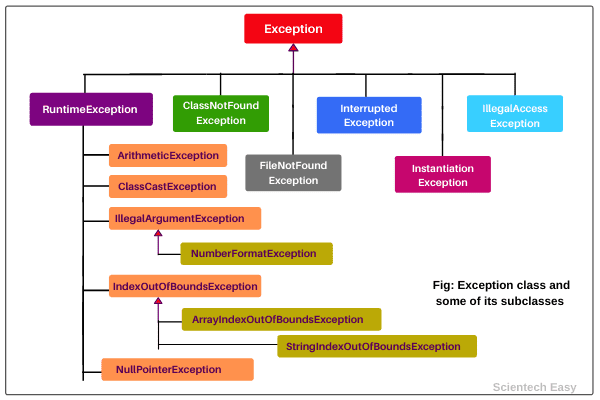
[](https://www.scientecheasy.com/2020/09/exception-handli%E2%80%A6erview-questions.html/)

Throwable class is the parent class of all exception types. It is an immediate subclass of the Object class. Below Throwable class, there are two subclasses (two child objects) Error and Exception.

**Error:** Error class is the subclass of Throwable class and a superclass of all the runtime error classes. It terminates the program if there is a problem related to a system or resources (JVM).

**Exception:** An exception is an abnormal condition that is caused by runtime error in the program. It is the superclass of all exceptions in Java. It is further divided into checked and unchecked (runtime) exceptions.

The exception class hierarchy has been shown in the below figure.

[](https://www.scientecheasy.com/2020/09/exception-handli%E2%80%A6erview-questions.html/)

Checked exceptions are those exceptions that are checked by Java compiler at compilation. A list of some important checked exceptions are given below:

* ClassNotFoundException
* InterruptedException
* InstantiationException
* IOException
* SQLException
* IllegalAccessException
* FileNotFoundException, etc

Runtime exceptions are those exceptions that are checked by JVM at runtime. Some important examples of runtime exceptions are given below:

* ArithmeticException
* ClassCastException
* NullPointerException
* ArrayIndexOutOfBoundsException
* NegativeArraySizeException
* ArrayStoreException
* IllegalThreadStateException
* SecurityException, etc.

**12. What does JVM do when an exception occurs in a program?**

 When JVM faces an exception in a program, it creates an exception object and throws it to inform us that an error has occurred. If the exception object is not caught and handled properly, JVM will display an error message and will terminate the rest of the program abnormally.

**13. How do we catch an exception?**

 We can catch an exception in either of two ways. They are:

* By try-catch block
* By using throws clause

**14. What is throwing an exception in Java?**

 When an exception occurs inside a method in java program, the method in which exception has occurred creates an exception object (i.e, an object of exception class) internally with the help of JVM and hands it over to the java runtime system (JVM). This process is called throwing an exception in java.

**15. What is catching an exception in Java?**

 The process of finding a handler by JVM to handle thrown exception is called catching an exception.

**16. What will happen to exception object after exception handling is done?**

 Once exception handling is done, the exception object will be garbage collected.

**17. What is the difference between checked and unchecked exceptions in Java?**

 Refer to answer to question 11.

**18. How will you handle the checked exception?**  
Or What are the different ways to handle checked exceptions?

 A checked exception can be handled either by using try and catch block or by using throws clause in the method declaration. If not handles properly, it will give a compile-time error.

**19. Which exception class can you use in the catch block to handle both checked and unchecked exceptions?**

 Exception class

**20. Can we throw checked exceptions from the static block?**

 We cannot throw because there is no specific place to catch it and it is called only once.

Pages: 1 [2](https://www.scientecheasy.com/2020/09/exception-handling-interview-questions.html/2/) [3](https://www.scientecheasy.com/2020/09/exception-handling-interview-questions.html/3/) [4](https://www.scientecheasy.com/2020/09/exception-handling-interview-questions.html/4/)

Categories[Core Java](https://www.scientecheasy.com/category/core-java/), [Interview](https://www.scientecheasy.com/category/interview/)Tags[Java Exception Interview Questions](https://www.scientecheasy.com/tag/java-exception-interview-questions/)

**Popular Java Tutorials**

Java Exception Handling Java Threads Java Collections Java List Interface & Implementation Classes Java Set Interface Java HashSet, LinkedHashSet, TreeSet Java Map Interface Map Implementation Classes Java Queue Java Deque Java Legacy Classes Java String

* [Home](https://www.scientecheasy.com/)
* [About Us](https://www.scientecheasy.com/2018/05/about-us.html/)
* [Contact](https://www.scientecheasy.com/contact/)
* [Privacy Policy](https://www.scientecheasy.com/category/privacy-policy/)

© Copyright 2020 Scientech Easy. All rights reserved.

[](https://www.hackertrail.com/talent/)

[](https://www.hackertrail.com/talent/)

* [**Interview Prep**](https://www.hackertrail.com/talent/backend/java/20-java-exception-handling-interview-questions-and-answers/)
* [**Practice Questions**](https://app.hackertrail.com/talent/)

Top of Form

SEARCH FOR:

When autocomplete results are available use up and down arrows to review and enter to go to the desired page. Touch device users, explore by touch or with swipe gestures.

Search

Bottom of Form

**20 Frequently Asked Exception Handling in Java Interview Questions and Answers (Updated 2021)**

* NOVEMBER 2, 2020
* 11 MINUTE READ

**7+**

Any Java interview is incomplete without Java exception handling interview questions.  Your Java programs can cope up with the unforeseen circumstances, called exceptions, quite efficiently through Java Exception handling which is a robust and user-friendly mechanism to handle errors in a Java program.

In this article, we will be covering the 20 most frequently asked exception handling in Java interview questions and answers for experienced and beginners to help them ace the interview.

The topics we will be covering are:

* Introduction to Exception handling in Java
* The Java Exception Hierarchy
* Types of Exceptions in Java
* Java Exception handling keywords
* Catching Exception in Java

**Introduction to Exception handling in Java**

**What is Java Exception Handling?**

An[exception in Java](https://docs.oracle.com/javase/tutorial/essential/exceptions/definition.html) is an occurrence of an unusual event at runtime, which interrupts the normal functioning of the program. If the program is not guided what to do in the case of an exception, it will terminate abnormally at the point of exception, without further executing the program. To avoid such program failures, there is a mechanism called Exception Handling in Java that prevents programs from failing under unanticipated situations. A few examples of exceptions occurring at run-time are:

* A program trying to access a non-existent file
* Invalid data entered by a user
* A program executing an invalid SQL query.

**Why do you need Java Exception Handling?**

If an exception occurs and there is no try and catch block to monitor and handle it, the program will terminate without executing the subsequent lines. Exception handling in Java is therefore required to ensure that the application does not end abruptly, execute catch and finally blocks in it, and tells it what to do in case of an exception.

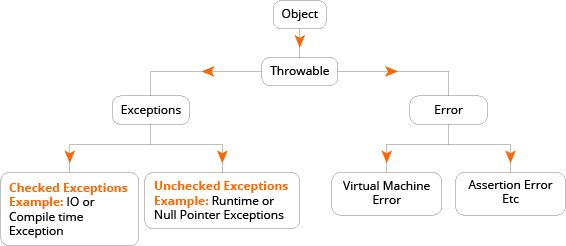
**What is the mechanism for handling exceptions in Java?**

Exception handling in Java is achieved through try, catch, and finally blocks.

* **try block**: The lines of code that are probable of causing an error are enclosed inside a try block so that they can be monitored for exceptions. The try block must be followed by either catch block or finally block or both, otherwise the program will throw a compilation error.
* **catch block**: If a try block throws an exception, the control directly goes to the catch block defined for that particular exception.  If no exception occurs, the program skips the catch block and executes normally.
* **finally block**: This block is optional, but if present, it always gets executed irrespective of the occurrence of an exception in the try block. The finally block can only be used in conjunction with the try-catch block.

**The Java Exception Hierarchy**

**What is the Java Exception Hierarchy**￼￼**?**

EXCEPTION HANDLING IN JAVA

When an exception occurs during the execution of an application, the JVM creates an Exception object in memory and interrupts the normal flow of the program. The throwable class creates a stack trace and also contains the relevant error message which can be obtained through getMessage() method. JVM then finds and executes a piece of code known as Exception Handler which processes the object of Exception Class.

**What is the difference between Errors and Exceptions?**

The key differences between errors and exceptions in Java are:

|  |  |
| --- | --- |
| **Errors** | **Exceptions** |
| Errors are generally caused by the application environment. Examples are OutOfMemoryError and StackOverflowError | Exceptions are caused by the code of the application itself. Examples are FileNotException and IndexOutOfBoundsException |
| It is usually impossible to recover from an error. | The application can recover from an exception by the use of **try-catch** blocks. |
| Errors are classified as unchecked as they are unknown to the compiler and occur at run-time. | Exceptions can either be “checked” or “unchecked,” i.e. they may or may not be caught during compilation. |

**Exception Handling Interview Questions: Types of Exceptions in Java**

**What are the types of Exceptions in Java?**

There are two types of exceptions based on how they are handled by JVM:

* **Checked Exceptions**

**Checked Exceptions** are known during the compilation process. The compiler checks whether the exception is handled or not and throws an error in case it is not handled programmatically. Examples of such exceptions are *SQLException, IOException, InvocationTargetException, and ClassNotFoundException.*

* **Unchecked Exceptions**

**Unchecked Exceptions** or run-time exceptions occur during program execution. These exceptions are not caught during the compilation process. All the exceptions under *Error*and *RuntimeException* classes are unchecked. Few examples of unchecked exceptions are ArithmeticException, NullPointerException, ArrayIndexOutOfBoundsException etc.

Based on how the exceptions are defined, here are other exception types

* **Built-In Exceptions**

The exceptions that already exist in the Java libraries are known as Built-in exceptions. All the checked and unchecked exceptions fall under this category.

* **User-defined Exceptions**

If a user comes across an exception scenario that is not covered by the built-in libraries, Java provides a provision to create a custom exception. A custom or user-defined exception can be constructed by creating a subclass of the ‘Exception’ class.

**How do you implement user-defined exception handling in Java?**

Here are the steps to execute **user-defined exception handling in Java**

1. Create your user-defined exception class by inheriting the built-in Exception class.
2. Next, create a constructor for your custom exception class. You can do this by either writing a default constructor within the CustomException or, you may create a parameterized constructor with a string argument.[Text Wrapping Break] This parameterized constructor can be used to store the details of the exception.
3. The ‘throw’ keyword is used to raise the user-defined exception. We create an object of the user-defined exception class and the throw clause to initiate that object.

Below is a user-defined**exception handling program in Java**:

// This program throws an exception whenever the balance

// amount is below Rs 1000

// Example taken from ￼GeekforGeeks

**class** MyException **extends** Exception

{

//store account information

**private** **static** **int** accno[] = {1001, 1002, 1003, 1004};

**private** **static** **String** name[] =

{"Customer1", "Customer2", "Customer3", "Customer4", "Customer5"};

**private** **static** **double** bal[] =

{10000.00, 12000.00, 5600.0, 999.00, 1100.55};

// default constructor

MyException() { }

// parameterized constructor

MyException(**String** str) { **super**(str); }

// write main()

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

{

**try** {

// display the heading for the table

System.out.println("ACCNO" + "\t" + "CUSTOMER" +

"\t" + "BALANCE");

// display the actual account information

**for** (**int** i = 0; i < 5 ; i++)

{

System.out.println(accno[i] + "\t" + name[i] +

"\t" + bal[i]);

// display own exception if balance < 1000

**if** (bal[i] < 1000)

{

MyException me =

new MyException("Balance is less than 1000");

**throw** me;

}

}

} //end of try

**catch** (MyException e) {

e.printStackTrace();

}

}

}

Runtime Error

MyException: Balance is less than 1000

at MyException.main(fileProperty.Java:36)

Output:

ACCNO CUSTOMER BALANCE

1001 Customer1 10000.0

1002 Customer2 12000.0

1003 Customer3 5600.0

1004 Customer4 999.0

**What are the dissimilarities between the two types of Java exceptions?**

Here are the differences between the exception types in Java

|  |  |
| --- | --- |
| **Checked Exceptions** | **Unchecked Exceptions** |
| A checked exception occurs and can be caught at compile time. | Unchecked exceptions happen during runtime. |
| The compiler checks for checked exceptions. | The compiler does not check for unchecked exceptions. |
| It is possible to handle checked exceptions thereby avoiding the exception as well as a compilation error. | Unchecked exceptions can’t be caught or handled during compilation because they get generated by the mistakes in the program. |
| They are the sub-class of the *Exception* class. | They are generally sub-class of the *Error* class. |
| The JVM expects the exception to be handled in the program itself. | Unchecked exceptions are not expected to be handled by the JVM. |

**Exception Handling Interview Questions: Types of Exception Handling Keywords**

**What are the different keywords in Exception handling in Java?**

Five keywords manage Java exception handling. These are: try, catch, throw, throws, and finally. We have already seen try, catch, and finally. Let us differentiate and understand the throw and throws keyword.

**What is the difference between throws and throw keywords in Java?**

The throws keyword is used to declare an exception in the signature of a method. This keyword can throw multiple exceptions, checked and unchecked both, separated by commas.

The method declaration will look like this:

public static void throwsKeywordExample() throws SocketException, ConnectionException

The throw keyword on the other hand is used to throw exceptions explicitly inside a method.  Unlike throws, there is only one exception per the ‘throw’ keyword.

**void** throwKeywordExample()

{

**throw** new ArithmeticException("Invalid value");

}

**How to differentiate between the finally, final, and finalize keywords?**

final, finally and finalize are three different keywords in Java all with specific usage. But they are often confused due to similarity names. Let us understand each of them:

final: Final is a keyword that is used to apply limitations on a class, function, or variable. You cannot inherit a final class, you can’t override a final method and you can’t change the value of a variable declared as final.

finally: finally keyword is used along with the try-catch block. This block consists of statements that need to be executed regardless of whether an exception occurs or not, for example, the closing of a database connection, etc.

finalize: The finalize() method of the Object class is used for clean up processing right before the object is garbage collected. The garbage collector calls this method when it finds that there are no more references to a particular object.

**Exception Handling Interview Questions: Catching Exceptions in Java**

**What is try-with-resources statement in Java?**

 A try statement that declares one or more resources is called a try-with-resources statement. Examples of resources are input and output stream, a database connection, etc. All such resources need to be closed at the end of the program execution. This statement ensures that the code closes the resources that were opened by the try keyword.

Here is an example taken from the oracle Java docs.

The below example reads the first line of a file via an instance of BufferedReader.  BufferedReader is the resource here that needs to be closed after its usage in the program.

// Try-with-resources example

**static** **String** readFirstLineFromFile(**String** path) **throws** IOException {

**try** (BufferedReader br =

new BufferedReader(new FileReader(path))) {

**return** br.readLine();

}

}

The try-with-resources is declared by declaring the resource object in parentheses immediately after the try keyword. As a BufferedReader instance is declared inside the try-with-resource statement, it will be closed even if the try statement terminates abruptly.

**What is stack trace in Java and why is it important in exception handling?**

A stack trace is a list of frames that contains information about methods that were called during your application’s execution. A stack trace gives the names of the methods and classes that were called throughout the lifecycle of the application until the occurrence of an exception.

In Java exception handling, stack trace can be used as a powerful debugging tool as it helps in determining the exact point where the exception occurred in the application and the reason why it occurred.

**What is a nested try-catch block?**

In exception handling, a nested try-catch block is one where we use a try-catch block inside another.

*Here is an example of nested try-catch block:*

/\*\*Example taken from Edureka.com\*\*/

**class** Exception{

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

**try**{

**try**{

System.out.println("going to divide");

**int** b=59/0;

}**catch**(ArithmeticException e){System.out.println(e);}

**try**{

**int** a[]=new **int**[5];

a[5]=4;

}

**catch**(ArrayIndexOutOfBoundsException e) {System.out.println(e);}

System.out.println("other statement);

}catch(Exception e)

{System.out.println("Exception handled");}

System.out.println("casual flow");

}

}

The above program will try to execute the inner try statement first. If the inner try statement doesn’t find a matching catch statement, the control will be transferred to the subsequent try block. The program will keep looking for catch statements until it succeeds or until all the nested try statements are executed. If no matching catch statement is found, the JRE will handle the exception.

**What is a multi-catch block in Java?**

There could be a scenario where a single piece of code can cause multiple exceptions. You can handle this situation by creating several catch clauses, each catching a different exception. When the program throws an exception, it inspects the catch statements in the order in which they appear. The first catch whose type matches the type of exception is executed. Below is an exception handling program in Java to understand the multi-catch block.

// Multi-catch block

// Code taken from Edureka.com

**public** **class** SampleMultipleCatchBlock{

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

**try**{

**int** a[]=new **int**[5];

a[5]=30/0;

}

**catch**(ArithmeticException e)

{System.out.println("task1 is completed");}

**catch**(ArrayIndexOutOfBoundsException e)

{System.out.println("task 2 completed");}

**catch**(Exception e)

{System.out.println("task 3 completed");}

System.out.println("remaining code");

}

}

**What are chained exceptions in Java?**

By using chained exceptions, you can associate one exception with another exception, i.e, the second exception is generally the cause for the first exception.

For instance, consider a method that throws an ArithmeticException because it attempted to divide by zero. However, the real cause for that exception was an I/O error that improperly set the divisor. Though the method ought to throw an ArithmeticException, as it is the error that occurred, the calling code should also know that the actual cause was an I/O error.

**How do you catch Multiple Exceptions in a single block of code?**

There are three ways by which you can catch multiple exceptions in a single code block:

1. By using a catch block that handles all the exception types being thrown by a method:

try {

// ...

} **catch** (Exception ex) {

// ...

}

A thing to remember is that the above code is not a recommended method though. It uses the generic Exception class as it handles all the exceptions. However, it is best to use exception handlers that are precise to the exceptions being thrown.

1. The second way is to implement multiple catch blocks:

try {

// ...

} **catch** (FileNotFoundException ex) {

// ...

} **catch** (EOFException ex) {

// ...

}

We need to keep in mind that if the exceptions share an inheritance relationship, the child type has to come before the parent type otherwise it will result in a compilation error.

1. The third is to use a multi-catch block which we have already covered in the previous section.

**What will be the result of the main method throwing an exception?**

If an exception is thrown by the main() method, the Java Runtime Environment will terminate the application and print the stack trace in-system console along with the exception message.

**Exception Handling Best Practices**

**What are the best practices in Java exception handling?**

Below are the top 10 best practices to follow for implementing Exception Handling in Java

1. You ought to use a finally block to clean-up resources. Alternatively, you can use a try-with-resources statement.
2. You should throw specific exceptions as far as possible.
3. You must not catch the Exception class, instead you should catch specific subclasses.
4. The Throwable class is not to be caught.
5. Make it a habit to correctly wrap the built-in exceptions in custom exceptions to ensure a readable stack trace.
6. Catch the most specific exception first
7. No exception should be thrown from the finally block
8. Avoid calling printStackTrace() or similar methods within your code.
9. Your code should validate the user input to minimize user errors as well as catch an adverse condition fairly early.
10. Include descriptive messages while throwing exceptions.

**How to use exception handling with method overriding?**

* When exception handling is used with method overriding, it causes ambiguity. The compiler gets confused as to which method definition it should follow, the one in the superclass or the subclass. Certain rules can help prevent this ambiguity while using exceptions with inheritance.
* If the parent class method does not throw exceptions, the overriding class method should throw only unchecked exceptions. A checked exception will result in a compilation error.
* In the case of the parent class method throws checked exceptions, the child class method may throw any unchecked exception as well as any or all of the checked exceptions thrown by the superclass.
* The child class can also declare a few checked exceptions that the parent class does not throw, but it needs to ensure that the scope of such checked exceptions in the child class is narrower as compared to the parent class.
* When the superclass throws an unchecked exception, the subclass method can throw either none or any number of unrelated unchecked exceptions.

You can understand [exception handling in method](https://www.geeksforgeeks.org/exception-handling-with-method-overriding-in-java/)￼[overriding](https://www.geeksforgeeks.org/exception-handling-with-method-overriding-in-java/) through code better.

Now when you are done with the most popular Java exception handling interview questions, we at Hackertrail are here to connect you to the right opportunities in your domain.

**7+**

* [INTERVIEW QUESTIONS](https://www.hackertrail.com/talent/tag/interview-questions/)
* [JAVA](https://www.hackertrail.com/talent/tag/java/)

[PREVIOUS](https://www.hackertrail.com/talent/tips/do-first-impressions-really-count/)

**[Do First Impressions Really Count?](https://www.hackertrail.com/talent/tips/do-first-impressions-really-count/)**

[NEXT](https://www.hackertrail.com/talent/backend/python/python-interview-questions-top-30-asked/)

**[Python Interview Questions: Top 30 Questions](https://www.hackertrail.com/talent/backend/python/python-interview-questions-top-30-asked/)**

YOU MAY ALSO LIKE



READ MORE

* 10 MINUTE READ

[**Top Java Spring Framework Interview Questions and Answers (Updated 2021)**](https://www.hackertrail.com/talent/backend/java/top-java-spring-interview-questions-and-answers/)

* OCTOBER 23, 2020

6+ Are you curious to learn about the Java Spring framework? Or are you looking for Java Spring interview questions for your next job…



READ MORE

* 6 MINUTE READ

[**Java Collections Interview Questions**](https://www.hackertrail.com/talent/backend/java-collections-interview-questions/)

* MARCH 2, 2021

0 Collections Framework forms the backbone of Java. Therefore, Java Collections interview questions hold immense importance for a career in…

1. [Introduction to Exception handling in Java](https://www.hackertrail.com/talent/backend/java/20-java-exception-handling-interview-questions-and-answers/#introduction-to-exception-handling-in-java)
2. [The Java Exception Hierarchy](https://www.hackertrail.com/talent/backend/java/20-java-exception-handling-interview-questions-and-answers/#the-java-exception-hierarchy)
3. [Exception Handling Interview Questions: Types of Exceptions in Java](https://www.hackertrail.com/talent/backend/java/20-java-exception-handling-interview-questions-and-answers/#exception-handling-interview-questions-types-of-exceptions-in-java)
5. [Exception Handling Interview Questions: Types of Exception Handling Keywords](https://www.hackertrail.com/talent/backend/java/20-java-exception-handling-interview-questions-and-answers/#exception-handling-interview-questions-types-of-exception-handling-keywords)
7. [Exception Handling Interview Questions: Catching Exceptions in Java](https://www.hackertrail.com/talent/backend/java/20-java-exception-handling-interview-questions-and-answers/#exception-handling-interview-questions-catching-exceptions-in-java)
8. [Exception Handling Best Practices](https://www.hackertrail.com/talent/backend/java/20-java-exception-handling-interview-questions-and-answers/#exception-handling-best-practices)

****

**TECHIES,  
WE'VE GOT YOU COVERED**

© 2021 TechKnowledgey Pte. Ltd

[**Interview Prep**](https://www.hackertrail.com/talent/)

[**For Candidates**](https://www.hackertrail.com/public/)

[**For Employers**](https://www.hackertrail.com/public/employer?src=generic)

[**About Us**](https://www.hackertrail.com/public/v2/about.html)