1)What is exception in java?

Exception in java is an abnormal condition raised while executing program In Java, an exception is an event that disrupts the normal flow of the program. It is an object which is thrown at runtime. If in a program we got exception in some line then next line of code will not execute. This exception must be handled properly. If it is not handled, program will be terminated abruptly. When an exception occurs within a method, it creates an object. This object is called the exception object. It contains information about the exception, such as the name and description of the exception and the state of the program when the exception occurred.

2)What is Exception Handling in java?

Exception Handling is a mechanism to handle runtime errors that the regular flow of of application can be presented such as ClassNotFoundException, IOException, SQLException, RemoteException, etc.

Exception Handling is the technique of handling unexpected failures that could occur in a program so that the program does not terminate and normal execution flow is maintained. Consider an example program where we have 6 statement blocks as shown in the below image. Statements 1 and 2 execute successfully. While executing the statement 3 block, an unexpected error occurred. Now the Java program checks if there is any exception-handling mechanism present. If an exception handling block is present, then the exception is handled gracefully and the program continues with the execution of the statement 4 block. If not, the program gets terminated.

3)Tell me the Hierarchy of about Java Exception Classes?

 Exceptions in Java are hierarchical and follow inheritance to categorize different kinds of exceptions. The parent class of all exceptions and errors is Throwable. This class has 2 child classes - Error and Exception.

1. Errors are those abnormal failures that are not in the scope of recovery for the application. It is not possible to anticipate errors or recover from them. Errors could be due to failures in hardware, out-of-memory, JVM crash, etc.
2. Exception is divided into 2 categories - Checked Exception and Unchecked Exception.
   * Checked Exceptions are those that can be anticipated in the program at the time of compilation and we should try to handle this otherwise it leads to a compilation error. IllegalAccessException, ClassNotFoundException, and FileNotFoundException are some of the types of checked exceptions. Exception is the parent class of all checked exceptions.
   * Unchecked exceptions are those exceptions that occur during runtime and are not possible to identify at the time of compilation. These exceptions are caused due to mistakes in application programming - for example, we try to access an element from an array index that is out of bounds from the length of the array, while trying to run operations on null objects and so on. RuntimeException, ArrayIndexOutOfBoundException, NullPointerException, ClassCastException, NumberFormatException, IllegalArgumentException, etc are some of the types of unchecked exceptions. The parent class of all runtime/unchecked exceptions is RuntimeException.

Ex: ClassNotFoundException and IOException

**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

In Java, exceptions could be handled in the following ways:

1. **try-catch block:** The try section holds the code that needs to be normally executed and it monitors for any possible exception that could occur. The catch block “catches” the exception thrown by the try block. It could consist of logic to handle failure scenario or the catch block could simply rethrow the exception by using the “throw” keyword.
2. **finally block:** Regardless of whether an exception has occurred or not, if we need to execute any logic, then we place it in the final block that is usually associated with the try-catch block or just with the try block. The final block is not executed when System.exit(0) is present in either the try or catch block.

**Q.**What is checked Exception? Or Compiletime Exception?

Cheked Exception is identify by the jvm at compile time. Exception that occur at compile time is called ad checked excetion.

For example, if you use **FileReader** class in your program to read data from a file, if the file specified in its constructor doesn't exist, then a FileNotFoundException occurs, and the compiler prompts the programmer to handle the exception IOException, SQLException,

Q.What is meant by ClassNotFoundException in Java?

ClassNotFoundException is a checked exception in Java that **occurs when the JVM tries to load a particular class but does not find it in the classpath**

**Q.When FileNotFoundExceptionOccur?**

FileNotFoundException is a checked exception in Java that occurs when an attempt to open a file denoted by a specified pathname fails. This exception is thrown by the FileInputStream , FileOutputStream , and RandomAccessFile constructors when a file with the specified pathname either does not exist or is inaccessible

**Q. What are runtime exceptions in Java/Unchecked Exception?**

Runtime exceptions are those exceptions that occur at the run time of the program execution. These exceptions are not noticed by the compiler at the compile time and hence the program successfully gets compiled. Therefore, they are also called unchecked exceptions. All subclasses of the java.lang.RunTimeException class and java.lang.Error class belongs to runtime exceptions. Examples of runtime exceptions include NullPointerException, NumberFormatException, ArrayIndexOutOfBoundException, StackOverflowError, ClassCastException, ArithmeticException, ConcurrentModificationException, etc.

Q.When NullPointerException is occurred?

NullPointerException is thrown when program attempts to use an object reference that has the null value. NullPointerException is raised in an application when we are trying to do some operation on null where an object is required. Some of the common reasons for

Q. **How to avoid the NullPointerException?**

To avoid the NullPointerException, we must ensure that all the objects are initialized properly, before you use them. When we declare a reference variable, we must verify that object is not null, before we request a method or a field from the objects.

Q.When ArrayIndexOutOfBoundException occur?

**An array Index Out Of Bounds Exception is thrown when a program attempts to access an element at an index that is outside the bounds of the array**. This typically occurs when a program tries to access an element at an index that is less than 0 or greater than or equal to the length of the array.

### 8. How do you handle checked exceptions?

Checked Exceptions can be handled by either using a try-catch block or by using the throws clause in the method declaration. If these exceptions are not handled properly, then the program would fail to compile.

**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.

**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 block or finally block.

**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, statement3 is not executed. Once a try block throws an exception, remaining statements will not be executed. Control comes directly to catch block.

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

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

**Ex:throw InstanceOfThrowableType;**

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

|  |  |  |
| --- | --- | --- |
| Final | Finally | finalize() |
| final is a keyword in Java which is used to make a variable or a method or a class as unchangeable. | finally is a block in Java which is used for exception handling along with try and catch blocks. | finalize() method is a protected method of java.lang.Object class which is used to perform some clean up operations on an object before it is removed from the memory. |
| The value of a variable which is declared as final can’t be changed once it is initialized. | finally block is always executed whether an exception is occurred or not and occurred exception is handled or not. | This method is called by garbage collector thread before an object is removed from the memory. |
| A method declared as final can’t be overridden or modified in the sub class and a class declared as final can’t be extended. | Most of time, this block is used to close the resources like database connection, I/O resources etc soon after their use. | This method is inherited to every class you create in Java. |

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

|  |  |  |
| --- | --- | --- |
| **Throw** | **Throws** | **Throwable** |
| throw is a keyword in Java which is used to throw an exception manually. | throws is also a keyword in java which is used in the method signature to indicate that this method may throw mentioned exceptions. | Throwable is a super class for all types of errors and exceptions in Java. This class is a member of java.lang package. |
| Using throw keyword, you can throw an exception from any method or block. But, that exception must be of type **java.lang.Throwable** class or it’s sub classes. | The caller to such methods must handle the mentioned exceptions either using try-catch blocks or using throws keyword. | Only instances of this class or it’s sub classes are thrown by the java virtual machine or by the throw statement. |

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

**5) What are the differences between StackOverflowError and OutOfMemoryError In Java?**

|  |  |
| --- | --- |
| **StackOverflowError** | **OutOfMemoryError** |
| It is related to Stack memory. | It is related to heap memory. |
| It occurs when Stack is full. | It occurs when heap is full. |
| It is thrown when you call a method and there is no space left in the stack. | It is thrown when you create a new object and there is no space left in the heap. |
| It occurs when you are calling a method recursively without proper terminating condition. | It occurs when you are creating lots of objects in the heap memory. |
| How to avoid? Make sure that methods are finishing their execution and leaving the stack memory. | How to avoid? Try to remove references to objects which you don’t need anymore. |

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

The throw keyword allows a programmer to throw an exception object to interrupt normal program flow. The exception object is handed over to the runtime to handle it. For example, if we want to signify the status of a task is outdated, we can create an OutdatedTaskException that extends the Exception class and we can throw this exception object as shown below:

**if** (task.getStatus().equals("outdated")) {

**throw** **new** OutdatedTaskException("Task is outdated");

}

The throws keyword in Java is used along with the method signature to specify exceptions that the method could throw during execution. For example, a method could throw NullPointerException or FileNotFoundException and we can specify that in the method signature as shown below:

**public** **void** **someMethod**() **throws** NullPointerException, FileNotFoundException {

// do something

}

Q.Explain try ,catch, finally , throw, throws?

|  |  |
| --- | --- |
| **Keyword** | **Description** |
| Try | The "try" keyword is used to specify a block where we should place an exception code. It means we can't use try block alone. The try block must be followed by either catch or finally. |
| Catch | The "catch" block is used to handle the exception. It must be preceded by try block which means we can't use catch block alone. It can be followed by finally block later. |
| Finally | The "finally" block is used to execute the necessary code of the program. It is executed whether an exception is handled or not. |
| Throw | The "throw" keyword is used to throw an exception. |
| throws | The "throws" keyword is used to declare exceptions. It specifies that there may occur an exception in the method. It doesn't throw an exception. It is always used with method signature. |

Q.Explain hierarchy of Exception?

Hierrarchy of exception

Java.lan is a class where u have an object and throwable is an class which is extended from java.lan object class

And again throwable class has two subclass error and exception Throwable is parent class and and its child classs is error and exception

Any kind of exception and error received in ur program will be created an ur instance of throwable class It will store information about exception and error that will receive

1)Error is that something can not be recover this is error is not related our program this related our memory area.This error is out of our program ex outofMemeory This can not be handled in program this is an error