# ***EXCEPTION-HANDLING INTRIVIEW QUESTIONS***

**Q1.What is an Exception?**
Ans.An unwanted, unexpected event that disturbs normal flow of the program is called Exception.Example: FileNotFondException.

**Q2.What is the purpose of Exception Handling?**
Ans.The main purpose of Exception Handling is for graceful termination of the program.

**Q3.What is the meaning of Exception Handling?**
Ans. Exception Handling doesn't mean repairing an Exception, we have to define alternative way to continue rest of the code normally.
Example: If our programming requirement is to read the data from the file locating at London but at Runtime if London file is not available then we have to use local file alternatively to continue rest of program normally. This is nothing but Exception Handling.

**Q4.Explain Default Exception Handling Mechanism in java?**
Ans.If an exception raised, the method in which it's raised is responsible for the creation of Exceptions object by including the following information:

- Name of the Exception
- Description of the Exception
- Stack Trace
- After creating Exception object the method handover it to the JVM.
- JVM checks for Exception Handling code in that method.
- If the method doesn't contain any Exception handling code then JVM terminates the method abnormally and removes the corresponding entry from the stack.
- JVM identify the caller method and checks for Exception Handling code in that method. If the caller doesn't contain any exception handling code then JVM terminates that method abnormally and removes the corresponding entry from the stack.
- This process will be continue until main() method.
- If the main() method also doesn't contain exception handling code the JVM terminates that main() method and removes the corresponding entry from the stack.
- Just before terminating the program abnormally JVM handovers the responsibility of exception handling to the Default Exception Handler which is the component of JVM.
- Default Exception Handler just print exception information to the consol in the following format

Name of Exception: Description
Stack Trace (Location of the Exception)

**Q5.What is the purpose of try?**
Ans We should maintain all risky code inside the try block.
**Q6. What is the purpose of catch block?**
Ans.We have to maintain all Exception Handling code inside the catch block.
**Q7. Is try with multiple catch block is possible?**
Ans. The way of handling an exception is varied from exception to exception compulsory we have to write a separate catch block for every exception. Hence try will multiple catch block is possible and it is recommended to use.

```
Example:
try{
        //Risky code
}
catch(IOException e)
{
        //Hndling code for IOException
}
catch(ArithmeticException e)
{
        //handling code for AE
}
catch(NullPointerExcetpion e)
{
        // handling code for NPE
}
catch(Exception e)
{
        //default exception handling code
}
```

**Q8. If try with multiple catch block present is order of catch blocks important in which order we have to take?**
Ans. If try with multiple catch block present then the order of catch block is very important it should be from child to parent but not from parent to child.

**Q9. What are various methods to print Exception information? and differentiate them.**

Ans.

Throwable class defines the following method to print exception or error information .
**1. printStackTrace()** :- This method print exception information in the following format.

**Name of the Exception: Description**
**StackTrace**

**2.toString():**- This method print exception information in the following format.

**Name of the Exception: Description**

**3.getMessage():**- This method prints only description of the exception.

**Description**

**Q10.If an exception rised inside catch block then what will happen?**
Ans. If an exception raised inside catch block and it is not part of any try block then it is always abnormal termination.

**Q11. Is it possible to take try, catch inside try block?**
Ans. Yes, It is possible to take try, catch inside try block. That is nesting of try catch is possible.

**Q12.Is it possible to take try, catch inside catch block?**
Ans. Yes, It is possible to take try, catch inside catch block.

**Q13. Is it possible to take try without catch?**
Ans. Yes, it is possible to take try without catch but compulsory finally block should be available.

**Q14. What is the purpose of finally block?**
Ans. The main purpose of finally block is, to maintain the cleanup code. This block will execute always.

**Q15. Is finally block will be execute always?**
Ans. Yes finally block will be executed always irrespective of whether exception raised or not raised whether exceptions are handled or not handle. There is one situation where the finally block won't be executed if the JVM is going to be shutdown.

**Q16. In which situation finally block will not executed?**
Ans. There is one situation where the finally block won't be executed if we are using system.exit(0) explicitly then JVM itself will be shutdown and there is no chance of executing finally block.

**Q17. If return statement present inside try is finally block will be executed?**
Ans. Yes, if return statement present inside try, then also finally block will be executed. finally block will dominate return statement also.

**Q18. What is the difference between final, finally and finalize()?**
Ans. **final:-** final is a modifier applicable for **variables, methods** and **classes**. **final variable** means constant and reassignment is not possible. **final method** means implementation is final in the child classes we can't override**. final class**means it won't participate in inheritance and child class creation is not possible.
**finally:-** It is a **block** associated with try catch to **maintain cleanup code**. Finally block will be executed always irrespective of whether exception is raised or not raised or whether the exception is handle or not handle.
**finalize():**- It is a **method**, Garbage collector always calls this method just before destroying any object to perform cleanup activities.

**Q19. Is it possible to write any statement between try-catch and finally?**
Ans. No, it is not possible to write any statement between try catch and finally. If we will try to write any statement between them then we will get compile time error.

**Q20. Is it possible to take two finally blocks for the same try?**
Ans. No, it is not possible to take two finally blocks for the same try. If we try to take then we will get compile time error.

**Q21. Is syntax try-finally-catch is valid ?**
Ans. No, this syntax is not valid. It should be like try-catch-finally then only code will compile.

**Q22. What is the purpose of throw?**
Ans. Sometimes we can create Exception object explicitly and we can handover that exception object to the JVM explicitly by throw keyword.
The purpose of throw keyword is to handover our created exception object explicitly to the JVM.

**Example1:**

```
class Test{
        public static void main(String[] args){
                System.out.println(10/0);
        }
        }
```

In this case ArithmeticException object created implicitly and handover to the JVM automatically by the main method.

**Example2:**

```
Class Test{
        Public static void main(String[] args){
                Throw new ArithmeticException("/by Zero");
                }
        }
```

In this case creation of an exception object and handover to the JVM explicitly by the programmer.

**Q23. Is it possible to throw an Error?**
Ans. Yes, It is possible to throw any Throwable type including Error.

**Q24. Is it possible to throw any java object?**
Ans. No, we can use throw keyword only for throwable objects otherwise we will get compile time error saying incompatible type.

**Q25. After throw is it allow to take any statement directly?**
Ans. After throw statement we are not allow to place any statement directly violation leads to compile time error saying Unreachable Statement.

**Q26. What is the purpose of throws?**
Ans. The main purpose of throws keyword is to delegate the responsibilities of exception handling to the caller. It requires in the case of checked exception.

**Q27. What is the difference between throw and throws?**
Ans. Sometimes we can create Exception object explicitly and we can handover that exception object to the JVM explicitly by throw keyword.The main purpose of throw keyword is to handover our created exception object explicitly to the JVM. The main purpose of throws keyword is to delegate the responsibilities of exception handling to the caller. It requires in the case of checked exception.

**Q28. What is the difference between throw and thrown?**
Ans. There is no terminology of thrown in java.

**Q29. Is it possible to use throws keyword for any java class?**
Ans. No, we can use throws keyword only for Throwable classes. Otherwise we will get compile time error saying Incompatible types.

**Q30. If we are taking catch block for an exception but there is no chance of rising that exception in try then what will happen?**
Ans. If there is no chance of raising an exception in try then we are not allow to write catch

block for that exception violation leads to compile time error. But this rule is applicable only for fully checked exception.

**Q31. Explain Exception Handling keyword?**

Ans. **Exception Handling keyword:**

Try :- To maintain Risky code.

Catch:- To maintain Exception Handling code.

Finally:- To maintain the clean up code.

Throw:- To handover our created exception object to the JVM explicitly.

Throws:- To delegate the responsibilities of Exception Handling to the caller.

**Q32. Which class act as root for entire java Exception hierarchy?**

Ans. Throwable class act as root for entire java Exception hierarchy.

**Q33. What is the difference between Error and Exception?**

Ans. Throwable class contain two child classes.

**Exception:-** These are mostly caused by our program and are recoverable.

**Error:**- These are not caused by our program, mostly caused by lake of system resources. These are non recoverable.

**Q34. What is difference between checked exception and unchecked exception?**

Ans. The exceptions which are checked by the compiler for smooth execution of the program at Runtime is called checked exception. Example: IOException, InterruptedException.The exceptions which are not checked by the compiler are called unchecked exception. Example: ArithmeticException,RuntimeException.

**Q35.What is difference between partially checked and fully checked Exception?**

Ans. A checked exception is said to be fully checked if and only if all the child classes also checked otherwise it is called partially checked exception.

Example:

IOException:- fully checked exception

Exception:- partially checked exception

Throwable:- partially checked exception

RuntimeException:- unchecked exception

**Q36. What is a customized Exception?**

Ans. Sometimes based on our programming requirement we have to create our own exception such type of exception are called customize Exception.

Example:

TooYoungException

TooOldException

InsufficientFundException

**Q37. Explain the process of creating the customized Exception.**

Ans. **Creating customized Exception:**

```
Class TooYoungException extends RuntimeException{
  TooYoungExcetpion(String desc){
            Super(desc);
            }
}
Class TooOldException extends RuntimeException
{
            TooOldException(String desc){
super(desc);
```

```
            }
}
Class custExcepiton{
            Public static void main(String[] args){
Int age=Integer.parseInt(args[0]);
If(age>60)
{
Throw new TooYoungException("Please wait some more time, definitely you will get best
match");
            }
            Else if(age<18)
            {
Throw new TooOldException("Your age is already crossed of marriage, no chance to getting
marriage");
            }
            Else
            {
                        System.out.println("Congratulation! You will get match details soon
by your email");
            }
}
```

**Q38. Explain control flow in try, catch, finally.**

```
Ans.      Try{
            Statement1;
            Statement2;
Statement3;
}
Catch(X e){
Statement4;
}
Finally{
Statement5;
}
Statement6;
```

Case1:
If there is no Exception then output is
Statement1
Statement2
Statement3
Statement5
Statement6
Normal termination
Case2:
If an exception raised at statement2 and corresponding catch block has matched then output is
Statement1
Statement4
Statement5
Statement5
Normal termination
Case3:
An exception raised at statement2 and corresponding catch has not matched then output is

Statement1
Statement5
Abnormal termination
Case4:
An exception occurs at statement4 it always Abnormal termination but before that finally block will be executed and output is
Statement1
Statement2
Statement5
Abnormal termination
Case5:
If an exception raised at statement5 or statement6, it is always abnormal termination.

**Q39. Can you give the most common occurred exception in your previous project.**

Ans. NullPointerException, ArrayIndexOutofBoundException, StackOverFlowError, ClassCastException, NoClassDefFoundError, ExceptionInitilizerError, IllegalArgumentException, NumberFormatException, IllegalStateException, AssertionError.