

try-catch Block

⇒ try and catch are 2 different keywords which are used in Exception handling
⇒ try :
→ try is the block which is used to store the risky code (in which exception can occur)
→ Syntax :-
try
{
 //risky code
}
catch :
→ catch is the block which is used to handle the exception
→ Syntax :-
try
{
 //risky code
} catch(ExceptionClassName ref_variable_name)
{
 //exception handling code
}
⇒ NOTE : try and catch block should be used together.
⇒ Syntax :-
try
{
 //risky code
} catch(ExceptionClassName ref_variable_name)
{
 //handling code
}

⇒ Working / Flow or try-catch Block

```
import java.util.Scanner;
public class Main{
    public static void main(String[] args)
    {
        Scanner scanner = new Scanner(System.in);
        try
        {
            System.out.println("Enter no 1");
            int no1 = scanner.nextInt();
            System.out.println("Enter no 2");
            int no2 = scanner.nextInt();
            int res = no1 / no2;
            System.out.println(res);
        }
        catch(Exception e)
        {
            System.out.println(e);
        }
        System.out.println("-----App Finished Successfully-----");
    }
}
```

Case 1 : no exception
-----App Started-----
Enter no 1
100
Enter no 2
0
Result : 25
-----App Finished Successfully-----

```
import java.util.Scanner;
public class Main{
    public static void main(String[] args)
    {
        System.out.println("-----App Started-----");
        Scanner scanner = new Scanner(System.in);
        try
        {
            System.out.println("Enter no 1");
            int no1 = scanner.nextInt();
            System.out.println("Enter no 2");
            int no2 = scanner.nextInt();
            int res = no1 / no2;
            System.out.println(res);
        }
        catch(Exception e)
        {
            System.out.println(e);
        }
        System.out.println("-----App Finished Successfully-----");
    }
}
```

Case 2 : Exception Occurred
-----App Started-----
Enter no 1
100
Enter no 2
0
Exception Occur
-----App Finished Successfully-----

⇒ What is the concept of catch() exception handling
→ Example 1 :-
try
{
 -----mysql database connection-----
 -----user values insert-----
} catch(Exception e)
{
 -----oracle database connection-----
 -----user values insert-----
}

→ Example 2 :-
try
{
 -----user input no1 and no2-----
 -----divide-----
} catch(Exception e)
{
 -----print you cannot divide by zero-----
 -----please enter another number-----
 -----if user divide by zero then replace zero by one-----
}

⇒ try-catch block combinations :-

```
try  
{  
}  
catch(Exception e)  
{  
}
```

```
try  
{  
}  
try  
{  
    -----another code-----  
} catch(Exception e)  
{  
}
```

```
try  
{  
}  
try  
{  
}  
catch(Exception e)  
{  
}
```

1. We use this type of combination when there are multiple exceptions
in try block.
2. If there are multiple catch blocks then first catch block should
have child exception class name or same level exception class name

```
try  
{  
    2-3 exceptions  
} catch(---)  
{  
} catch(---)  
{  
}
```

```
try  
{  
} catch(IOException e)  
{  
} catch(SQLException e)  
{  
} catch(SQLEXception e)  
{  
}
```

⇒ Multiple Exception Class Name in catch Block :

```
try  
{  
    2 or 3 exceptions  
} catch(IOException | ArithmeticException | SQLException e)  
{  
}
```

⇒ Different ways to print the exceptions :-
try
{
 1. e.printStackTrace();
 => It will print all three things
 2. System.out.println(e.getMessage());
 => It will print 1 and 2 (not 3)
 3. System.out.println(e.getLocalizedMessage());
 => It will print 2nd part

finally Block

⇒ NOTE : catch block will be executed only when there is exception in try block,
but if there is no exception in try block then catch block will not execute

⇒ finally is the block which is always executed whether there is exception or not

⇒ Syntax :-

```
try  
{  
}  
catch(---)  
{  
}  
finally  
{  
    // resource closing code  
}
```

⇒ Use : finally block is used to close the resources which are opened
For example - database connection is opened
file is opened

try-with-resource

⇒ try-with-resource is used when we want to pass the responsibility of closing the resources
try
{
}
finally
{
 // resource closing code
}

⇒ Syntax :-

```
try(Resource opening code)
{
}
catch(ExceptionClassName e)
{
}
```

Interview Questions

1. What is the purpose of the try and catch blocks in Java?
2. Can a try block exist without a catch block? If yes, how?
3. Can a block have multiple catch blocks?
4. Can we have multiple catch blocks for a single try block? Why would we use them?
5. Can we have multiple try blocks with one catch block?
6. What is the order of execution when multiple catch blocks are used?
7. What is the order of execution when multiple catch blocks are used?
8. Can we catch multiple exceptions in a single catch block? If yes, how?
9. What happens if a catch block catches an exception that is not thrown in the try block?
10. What happens if an exception occurs inside a catch block itself?
11. Can we nest try-catch blocks in Java? Provide a scenario.
12. What happens if both try and finally blocks have a return statement?
13. Is it possible that the code might run in catch, will the finally block execute?
14. Can a finally block exist without catch?
15. Is the finally block always executed in Java? When is it not executed?
16. What is try-with-resources in Java?
17. What is the difference between try-with-resources?
18. What are the advantages of using try-with-resources over try-finally?
19. What types of objects can be used in try-with-resources?
20. How does try-with-resources work internally?
21. What is the difference between try-with-resources and try-with-resources statement?
22. What happens if an exception is thrown while closing a resource?
23. What are suppressed exceptions in try-with-resources? How are they handled?
24. Can we use catch and finally along with try-with-resources?
25. What is the difference between try-with-resources and using a finally block for closing resources?

⇒ Ques:-
26. Can we have try-catch-finally together? Explain the execution order.
27. What happens if an exception is thrown in try and another in finally?
28. What happens if an exception is thrown in both try and catch blocks?
29. What happens if an exception is thrown in finally block?
30. What happens if a resource fails to initialize in try-with-resources?
31. Can we declare a try-with-resources block without a catch or finally block?
32. What is the difference between try-with-resources and try-finally?
33. What are the different ways to print an exception in Java?
34. What is the difference between printStackTrace(), getMessage(), and toString() methods?
35. When do we use printStackTrace() versus getMessage()
36. How can we print the details of an exception to the console?
37. What is the best practice for printing exceptions in production applications?