

Java try block must be followed by either **catch or finally block or both.**

**It is not necessary that each try block must be followed by a catch block. It should be followed by either a catch block or a finally block.** And whatever exceptions are likely to be thrown should be declared in the throws clause of the method.

A try block can be followed by one or more catch blocks. E

#### **Rule: For each try block there can be zero or more catch blocks, but only one finally block.**

#### **Rule: By default Unchecked Exceptions are forwarded in calling chain (propagated).**

* **error:** beyond your control e.g. you are unable to do anything if there occurs VirtualMachineError or StackOverflowError.

Points to remember

* At a time only one exception occurs and at a time only one catch block is executed.
* All catch blocks must be ordered from **most specific to most general**, i.e. catch for ArithmeticException must come before catch for Exception.

Let's see an example, to handle the exception **without maintaining the order of exceptions** (i.e. from most specific to most general).

1. **class** MultipleCatchBlock5{
2. **public** **static** **void** main(String args[]){
3. **try**{
4. **int** a[]=**new** **int**[5];
5. a[5]=30/0;
6. }
7. **catch**(Exception e){System.out.println("common task completed");}
8. **catch**(ArithmeticException e){System.out.println("task1 is completed");}
9. **catch**(ArrayIndexOutOfBoundsException e){System.out.println("task 2 completed");}
10. System.out.println("rest of the code...");
11. }
12. }
13. **Output:**
14. Compile-time error

Unlike Unchecked Exceptions, the propagation of exception **does not happen** in case of Checked Exception and its mandatory to use [throw keyword](https://www.geeksforgeeks.org/throw-throws-java/) here. Only unchecked exceptions are propagated.**Checked exceptions throw compilation error.**

1. If we throw an [unchecked exception](https://howtodoinjava.com/java/exception-handling/checked-vs-unchecked-exceptions-in-java/) from a method, it is not mandatory to handle the exception or declare in throws clause. For example, NullPointerException is an unchecked exception.

|  |
| --- |
| **Handling unchecked exceptions is not mandatory** |
| public class JavaExample  {      public static void main(String[] args)      {          method();      }        public static void method( ) {          throw new NullPointerException();      }  } |

2. But if we throw a checked exception using throw statement, we MUST either handle the exception in **catch block** or method much explicitly declare it using throws declaration. For example, FileNotFoundException is a checked exception.

|  |
| --- |
| **Checked exceptions must be handled** |
| public class JavaExample  {      public static void main(String[] args)      {          try          {              method();          }          catch (FileNotFoundException e)          {              e.printStackTrace();          }      }        public static void method( ) throws FileNotFoundException      {          throw new FileNotFoundException();      }  } |