**Checked Exceptions in Java**

Checked exceptions are those exceptions that are checked by the java compiler itself at compilation time and are not under runtime exception class hierarchy. If a method throws a checked exception in a program, the method must either handle the exception or pass it to a caller method.

Checked exceptions must 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.

The exceptions that are checked by Java compiler at compilation time is called checked exception in Java. All the exceptions except RuntimeException, Error, and their subclasses are checked exceptions.

**List of Checked Exceptions in Java**

* ClassNotFoundException
* InterruptedException
* InstantiationException
* IOException
* SQLException
* IllegalAccessException
* FileNotFoundException, etc

**Unchecked Exceptions (Runtime Exceptions) in Java**

Unchecked exceptions in Java are those exceptions that are checked by JVM, not by java compiler. They occur during the runtime of a program. All exceptions under runtime exception class are called unchecked exceptions or runtime exceptions in Java.

* We can write a Java program and compile it. But we cannot see the effect of unchecked exceptions and errors until we run the program. This is because Java compiler allows us to write a Java program without handling unchecked exceptions and errors.

**List of Unchecked Exceptions in Java**

Some important examples of runtime exceptions are given below:

* ArithmeticException
* ClassCastException
* NullPointerException
* ArrayIndexOutOfBoundsException
* NegativeArraySizeException
* ArrayStoreException
* IllegalThreadStateException
* SecurityException, etc.

**Difference between Checked and Unchecked Exceptions in Java**

There are many important differences between checked and unchecked exceptions in java. They are as follows:

1. The classes that directly inherit Throwable class except RuntimeException and Error are called checked exceptions whereas, classes that directly inherit RuntimeException are called unchecked exceptions.

2. Checked exceptions are checked and handled at compile-time whereas, unchecked exceptions are not checked and handled at compile time. They are checked at runtime.

3. Examples of checked exceptions are IOException, SQLException, ClassNotFoundException, etc whereas, examples of unchecked exceptions are ArithmeticException, ClassCastException, NullPointerException, IllegalArgumentException, etc.

4. When a checked exception occurs in a method, the method must either catch the exception or pass exception to its caller method. But in the case of unchecked exception, Java compiler does not force to catch exception or to declare it in a throws clause.

5. Checked exceptions in java extends Exception class whereas, unchecked exceptions extends RuntimeException class.

6. A checked exception happens when there is a chance of higher failure rate. whereas unchecked exceptions occur mostly due to programming mistakes/errors.

**Exception**

Java runtime system (JVM) was responsible for identifying exception class, creating its object, and throwing that object.

JVM automatically throws system-generated exceptions. All those exceptions are called implicit exceptions.

If we want to throw an exception manually or explicitly, for this, Java provides a keyword **throw**.

**Throw Keyword in Java**

**Throw in Java** is a keyword that is used to throw a built-in exception or a custom exception explicitly or manually. Using throw keyword, we can throw either checked or unchecked exceptions in java programming.

When an exception occurs in the try block, throw keyword transfers the control of execution to the caller by throwing an object of exception.

Only one object of exception type can be thrown by using throw keyword at a time. Throw keyword can be used inside a method or static block provided that exception handling is present.

The syntax to throw an exception manually in java is as follows:

**Syntax:**

throw exception\_name;

where exception\_name is a reference to an object of [Throwable class](https://www.scientecheasy.com/2020/05/throwable-in-java.html/) or its subclass.

For example:

throw new ArithmeticException();

or,

ArithmeticException ae = new ArithmeticException();

throw ae;

throw new NumberFormatException();

**Key points:**

1. The object of Throwable class or its subclasses can be created using new keyword or using a parameter inside catch clause.

2. Instances of classes other than Throwable class or its subclasses cannot be used as exception objects.

The throw keyword in Java is used for explicitly throwing a single exception. This can be from within a method or any block of code. Both checked and unchecked exceptions can be thrown using the throw keyword.

When an exception is thrown using the throw keyword, the flow of execution of the program is stopped and the control is transferred to the nearest enclosing try-catch block that matches the type of exception thrown. If no such match is found, the default exception handler terminates the program.

**Throws keyword in Java** is used in the method declaration. It provides information to the caller method about exceptions being thrown and the caller method has to take the responsibility of handling the exception.

Throws keyword is used in case of checked exception only because if we are not handling runtime exceptions (unchecked exceptions), Java compiler does not give any error related to runtime exceptions. If an error occurs, we are unable to do anything.

When the code generates a checked exception inside a method but the method does not handle it, Java compiler detects it and informs us about it to handle that exception.

In this case, compulsorily, we must handle that checked exception otherwise we will get an error flagged by Java compiler.

To prevent this error flagged by the compiler, we need to handle exceptions using throw clause. There are two ways to handle the exception:

1. By using [try-catch block](https://www.scientecheasy.com/2020/05/java-try-catch-block.html/)  
2. By using throws keyword

**package** basic.com;

**public** **class** ThrowTest {

**static** **void** m1() {

**throw** **new** ArithmeticException("hello");

}

**public** **static** **void** main(String[] args) **throws** Exception {

**try** {

*m1*();

} **catch** (ArithmeticException ae) {

System.***out***.println(ae.getMessage());

}

}

}

**package** basic.com;

**import** java.io.IOException;

**public** **class** ThrowsNewExample {

**static** **void** display() **throws** IOException {

**throw** **new** IOException();

}

**public** **static** **void** main(String[] args) {

// **TODO** Auto-generated method stub

**try** {

*display*();

} **catch** (IOException e) {

// **TODO** Auto-generated catch block

System.***out***.println("cought");

}

}

}