**Exception handling** in java – is an unexpected occurrence that happens during the execution of a program. It disrupts the regular flow of the flow of the program, and an exception terminates the program execution.

When an exception occurs, we receive a system-generated message. However we can handle the exception in java to provide a meaningful message to the user rather than a system-generated error.

**Why does an exception occur** – exception is a mechanism in java used to handle runtime errors such that we can maintain the program’s regular flow of execution. If an exception occurs, the remaining code is not executed. As the result, java compiler creates an exception object, and this exception object directly jumps to the default catch mechanism. Thus, the program is terminated after a default message is printed on the screen.

**Advantages of exception handling** – the primary advantage of exception handling is that it keeps the program’s regular flow. By using the exception handling mechanism, all the statements in our program will be executed, and the normal flow of the program will be sustained as well as a user-friendly message will be generated rather than a system-generated error massage.

**Exception vs. error**:

**Error** – program does not cause errors the majority of the time. Instead these are caused by a lack of system resources. Ex>- OutOfMemory.

Unrecoverable.

**Exception** – is an unwanted event that occurs during the program execution. It interrupts the regular flow of the program. Ex>- IOException, ArithmaticException, NullPointerException.

Recoverable.

**Keywords in exception handling** - there are 5 keywords:

1. **Try** – we can write code that might throw an exception in the try block. A try block in java must be followed by a catch block or finally block.
2. **Catch** – is used to handle any exceptions that might occur in our software. We can only use it after the try block. For a single try block we can use multiple catch blocks.
3. **Finally** – is used to clean up code or release resources that are being used by the program. Whether or not the exception are dealt, the finally block is always executed.
4. **Throw** – is used to throw an exception. It has the ability to throw exceptions explicitly. Using throw keyword, we can throw either checked or unchecked exceptions.
5. **Throws** – is used to declare an exception. Using the throws keyword, we can only declare checked exceptions. When we declare an exception, it is the programmer’s responsibility to write the exception handling code.

**Types of exception** in java –

1. **Checked** – except for error and runtime exception classes, all other classes that inherit the throwable class are the checked exceptions. At compile time, the compiler checks he checked exceptions. If the compiler does not handle the exception, we will get a compile time error in the case of checked exceptions. These exceptions can be handled using try-catch blocks or by declaring the exception with throws keyword.

**How to handle checked exception** –

2 ways to handle checked exceptions

* Using try-catch – to handle the checked exception, we can use try-catch. Handling exceptions is always a good idea because we should provide a meaningful message to the user for each exceptions type so that the user can easily understand the exception message.
* Using throws keyword

1. **Unchecked** – exception refers to all classes that inherit form runtime exception. The compiler does not check unchecked exceptions at compile time. However, they are checked at runtime. In unchecked exception, we do not get a compile-time error if the programmed does not handle the exception. The majority of the time, this exception happens due to incorrect data entered by the user (divide by zero). Unchecked exceptions include ArithmeticException, NullPointerException, ArrayIndexOutOfBoundException, etc.

**How to handle unchecked exception** – using try catch we can handle the unchecked exception. The try statement allows users to specify a block of code that will be checked for errors. In the contrast, the catch block will capture the provided exception object to performthe necessary operation.

**Multiple catch block** – one or more catch blocks can be used after try block. If we want to perform different tasks in response to different exceptions, we can use multiple catch blocks.

Note that only one exception can occur at a time and only one catch can be executed at a time.

All the catch block must be executed in the order specified.

**Explicitly throw an exception** - throw keyword is used to explicitly throw an exception. Using the throw keyword, we can throw either checked or unchecked exceptions. The throw is mostly used to throw user defined exceptions.

**Finally block** – is used to execute important code such as closing a connection stream etc. whether or not the exceptions are handled, the finally block is always executed.

**Usage** – 3 possible scenarios where we can use the finally block.

* Exception does not occur – prog. runs without throwing and exception, and the finally block is executed after the try block.
* Exception occurs and not handled by the catch block – the prog. throws an exception but is not handled by the catch block, and the prog. terminates abnormally.
* Exception occurs and handled by the catch block – the prog. Throws an exception but is not handled by the catch block. So the finally block executes after the try block and the prog. Terminates abnormally.

**User defined exception** – java allows us to create our own exception, which is essentially a derived class of exception. To create our own exception, we must first create a class that extends the exception class and represents user defined exceptions. We must pass the string to constructor if the superclass, which is obtained by calling the getMessage() function if the newly created object.