Exception Handling in Java, Need of Exception Handling, How to implement exception handling, about try, catch and finally block.

K.L.Madhavi

Exception Handling

What is Exception?

In Java, an exception is an event that occurs during the execution of a program that disrupts the normal flow of instructions. These exceptions can occur for various reasons, such as invalid user input, file not found, or division by zero. When an exception occurs, it is typically represented by an object of a subclass of the java.lang.Exception class.

Exception Handling

If there is any exception in our program then automatically the rest of the code will not be executed and the program will be terminated.

To avoid this abnormal termination of program,we use exception handling.

It is a mechanism to handle runtime errors such as ClassNotFoundException, IOException, SQLException, RemoteException, etc..

Need of Exception Handling

1. Preventing Program Crashes:

•When an exception occurs (e.g., division by zero, file not found), the program terminates abruptly if not handled.

•Exception handling allows you to gracefully catch these exceptions, preventing crashes and allowing your program to continue running or terminate in a controlled manner.

2. Error Recovery:

•With exception handling, you can implement error recovery mechanisms.

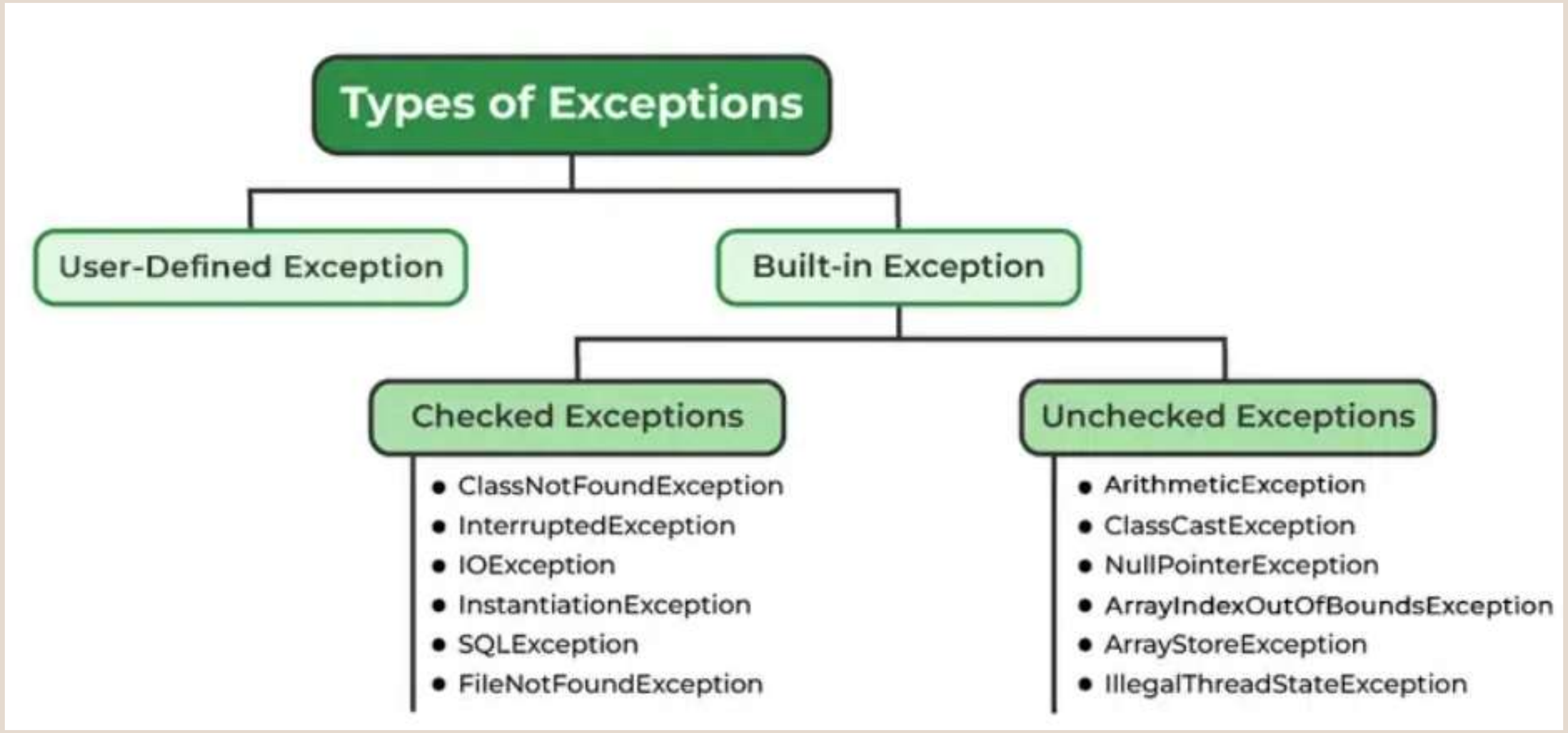
•For example, if a file is not found, you can prompt the user for a different file or retry the operation.

3. Meaningful Error Messages:

•Exception handling allows you to provide meaningful error messages to the user or log them for debugging purposes.

•This helps in identifying and fixing the root cause of the problem.

Types of Exceptions



1. Built-in Exceptions

Built-in exceptions are the exceptions that are available in Java libraries. These exceptions are suitable to explain certain error situations.

i.Checked Exceptions: Checked exceptions are called compile-time exceptions because these exceptions are checked at compile-time by the compiler.

→**IOException:** An exception is thrown when an input/output operation fails, such as when reading from or writing to a file.

→**SQLException:** It is thrown when an error occurs while accessing a database.

→**ParseException:** Indicates a problem while parsing a string into another data type, such as parsing a date.

→**ClassNotFoundException:** It is thrown when an application tries to load a class through its string name using methods like Class.forName(), but the class with the specified name cannot be found in the classpath.

ii.Unchecked Exceptions: The unchecked exceptions are just opposite to the checked exceptions. The compiler will not check these exceptions at compile time. In simple words, if a program throws an unchecked exception, and even if we didn't handle or declare it, the program would not give a compilation error.

→**NullPointerException:** It is thrown when trying to access or call a method on an object reference that is null.

→**ArrayIndexOutOfBoundsException:** It occurs when we try to access an array element with an invalid index.

→**ArithmeticException:** It is thrown when an arithmetic operation fails, such as division by zero.

→**IllegalArgumentException:** It indicates that a method has been passed an illegal or inappropriate argument.

2. User-Defined Exceptions:

Sometimes, the built-in exceptions in Java are not able to describe a certain situation. In such cases, users can also create exceptions, which are called 'user-defined Exceptions'.

Java Exception Keywords

Keyword	Description
try	The "try" keyword is used to specify a block where we should place an exception code. It means we can't use try block alone. The try block must be followed by either catch or finally.
catch	The "catch" block is used to handle the exception. It must be preceded by try block which means we can't use catch block alone. It can be followed by finally block later.
finally	The "finally" block is used to execute the necessary code of the program. It is executed whether an exception is handled or not.
throw	The "throw" keyword is used to throw an exception.
throws	The "throws" keyword is used to declare exceptions. It specifies that there may occur an exception in the method. It doesn't throw an exception. It is always used with method signature.

Syntax of try,catch and finally blocks

```
try
{
    // Code that may throw an exception
}
catch (Exception e)
{
    // Exception handling code
}
Finally
{
    // Cleanup code
}
```

Example Programs

i.
```
Class Ex
{
  void display()
  {
        try
        {
            int a=5/0;
         }
        catch(ArithmeticException e)
        {
             System.out.println(e);
         }
       System.out.println("Exception handled…");
    }
}
```

```
Class Except
{
    public static void main(String args[])
    {
       Ex obj=new Ex();
       obj.display();
     }
}
```

Output:

```
Java.lang.ArithmeticException:/ by Zero
Exception handled…
```

ii.

```
Class Ex
{
  void display()
  {
         try
         {
              int a[]=new int[5];
              a[6]=5;
          }
         catch(ArrayIndexOutOfBoundsException e)
         {
               System.out.println(e);
          }
        System.out.println("Exception handled…");
    }
}
```

```
Class Except
{
     public static void main(String args[])
     {
         Ex obj=new Ex();
         obj.display();
      }
}
```

Output:

```
Java.lang.ArrayIndexOutOfBoundsException:6
Exception handled…
```

iii.

```
Class Ex
{
  void display()
  {
         try
         {
              String Str=null;
              System.out.println(Str.length());
          }
         catch(NullPointerException e)
         {
               System.out.println(e);
          }
        System.out.println("Exception handled…");
    }
}
```

```
Class Except
{
     public static void main(String args[])
     {
         Ex obj=new Ex();
         obj.display();
      }
}
```

Output:

```
Java.lang.NullPointerException
Exception handled…
```

thank you