# Exception Handling

Unexpected or unwanted events can occur either at compile-time or run-time in the application code. It could be due to the programming mistake as well.

Types of errors:

1. Compile-time error -> Syntax error
2. Run-time error -> Exceptions
3. Logical error

* All exceptions in java are classes.
* Keywords in exception handling

1. Try
2. Catch
3. Finally
4. Throw
5. Throws

* Try block must be associated with at least one catch block or finally block
* Whenever an exception occurs in a program, JVM will create an object in the corresponding exception class and pass the object to the exception handler, and then the exception handler code will be executed. Here catches block is called as an exception handler.
* Syntax

Try

{

===========Task==========

}

Catch (Exceptionclass Exceptionclassref)

{

}

* 2 types of exceptions

1. Checked exceptions – these are checked at compile time.
2. Unchecked exceptions. – this will happen at run time.

Diagram

Description automatically generated with medium confidence

import static java.lang.Integer.\*;

import static java.lang.System.\*;

public class Demo {

public static void main (String args[])

{

              int x = parseInt("210");

              int y = parseInt("345");

              out.print(x/y);

}

}

Text

Description automatically generated

The above all exceptions are unchecked exceptions, these exceptions are handled by System implicitly. We can handle the above exceptions explicitly to display user friendly error messages.

import static java.lang.Integer.\*;

import static java.lang.System.\*;

public class Demo {

public static void main (String args[])

{

              try

              {

              int x = parseInt("210");

              int y = parseInt("345");

              out.print(x/y);

              }

              catch (ArrayIndexOutOfBoundsException ex)

              {

                             System.err.println("Please pass all arguments");

              }

              catch (NumberFormatException ex)

              {

                             System.err.println("Please provide Integers");

              }

              catch (ArithmeticException ex)

              {

                             System.err.println("Divide by Zero Problem");

              }

**catch** (RuntimeException ex)

       {

             System.***err***.println("Please pass all arguments and second number should not be Zero");

       }

}

}

Upcasting : Subclass object passed to super class reference variable. It works everywhere.

Down casting: this need type casting.

**catch** (Exception ex)

       {

             ==================

       }

This catch block handles all exceptions because all exceptions are sub classes of exception class.

**catch** (RuntimeException ex)

       {

             =================

       }

This catch block handles all unchecked exceptions because all unchecked exceptions are sub classes of runtime exception class.

**catch** (InterruptedException ex)

       {

             ====================

       }

This catch block handles all Interrupted exceptions and its sub classes.

**catch** (IOException ex)

       {

             ====================

       }

This catch block handles all IO exceptions and its sub classes.

**catch** (InterruptedException | IOException ex)

       {

             ====================

       }

This catch block handles all Interrupted Exception and its subclasses, IO exceptions and its sub classes.

Handling multiple exceptions with single catch block concept introduced in JDK 1.7 Version in 2011.

# User defined Exceptions

## Checked Exceptions

* Try and catch block is required.

Refer code snippet: UserDefinedExceptionExample1

|  |
| --- |
| // When Input 10  **package** exceptionhandling;  **import** java.io.BufferedReader;  **import** java.io.IOException;  **import** java.io.InputStream;  **import** java.io.InputStreamReader;  **class** NegativeNumberException **extends** Exception  {    }  **class** cube  {  **public** **int** print(**int** x) **throws** NegativeNumberException  {  **if** (x > 0)  **return** x\*x\*x;  **else**  **throw** **new** NegativeNumberException();  }  }  **public** **class** UserDefinedExceptionExample1  {  **public** **static** **void** main(String[] args) **throws** IOException  {  BufferedReader br = **new** BufferedReader(**new** InputStreamReader(System.***in***));  **try**  {  **int** x = Integer.*parseInt*(br.readLine());  **int** y = **new** cube().print(x);  System.***out***.println(y);  }  **catch** (NegativeNumberException ex)  {  System.***out***.println(ex);  }  }  }  //Output - 1000 |

|  |
| --- |
| //When input -10  **package** exceptionhandling;  **import** java.io.BufferedReader;  **import** java.io.IOException;  **import** java.io.InputStream;  **import** java.io.InputStreamReader;  **class** NegativeNumberException **extends** Exception  {    }  **class** cube  {  **public** **int** print(**int** x) **throws** NegativeNumberException  {  **if** (x > 0)  **return** x\*x\*x;  **else**  **throw** **new** NegativeNumberException();  }  }  **public** **class** UserDefinedExceptionExample1  {  **public** **static** **void** main(String[] args) **throws** IOException  {  BufferedReader br = **new** BufferedReader(**new** InputStreamReader(System.***in***));  **try**  {  **int** x = Integer.*parseInt*(br.readLine());  **int** y = **new** cube().print(x);  System.***out***.println(y);  }  **catch** (NegativeNumberException ex)  {  System.***out***.println(ex);  }  }  }  //Output - exceptionhandling.NegativeNumberException |

## Unchecked Exceptions

* Try and catch block is required.

Refer code snippet: UserDefinedExceptionExample2

|  |
| --- |
| // When Input 10  **package** exceptionhandling;  **import** java.io.BufferedReader;  **import** java.io.IOException;  **import** java.io.InputStreamReader;  **class** NegativeNumberException1 **extends** RuntimeException  {    }  **class** cube1  {  **public** **int** print(**int** x) **throws** NegativeNumberException1  {  **if** (x > 0)  **return** x\*x\*x;  **else**  **throw** **new** NegativeNumberException1();  }  }  **public** **class** UserDefinedExceptionExample2  {  **public** **static** **void** main(String[] args) **throws** IOException  {  BufferedReader br = **new** BufferedReader(**new** InputStreamReader(System.***in***));    **int** x = Integer.*parseInt*(br.readLine());  **int** y = **new** cube1().print(x);  System.***out***.println(y);  }  }  //Output –  10  1000 |

|  |
| --- |
| // When Input -10  **package** exceptionhandling;  **import** java.io.BufferedReader;  **import** java.io.IOException;  **import** java.io.InputStreamReader;  **class** NegativeNumberException1 **extends** RuntimeException  {    }  **class** cube1  {  **public** **int** print(**int** x) **throws** NegativeNumberException1  {  **if** (x > 0)  **return** x\*x\*x;  **else**  **throw** **new** NegativeNumberException1();  }  }  **public** **class** UserDefinedExceptionExample2  {  **public** **static** **void** main(String[] args) **throws** IOException  {  BufferedReader br = **new** BufferedReader(**new** InputStreamReader(System.***in***));    **int** x = Integer.*parseInt*(br.readLine());  **int** y = **new** cube1().print(x);  System.***out***.println(y);  }  }  //Output  -10  Exception in thread "main" exceptionhandling.NegativeNumberException1  at exceptionhandling.cube1.print(UserDefinedExceptionExample2.java:18)  at exceptionhandling.UserDefinedExceptionExample2.main(UserDefinedExceptionExample2.java:28) |

## Difference between throw and throws

Throw:

It is used to throw an object of exception class to catch block.

Throws:

It is used to apply an exception to a method, and it is also used to handle an exception.