**Exception:-**

* Unwanted or unexpected event that disturbs the normal flow of program is known as exception.
* It helps maintain the normal, desired flow of the program even when unexpected events occur.
* Defining alternative way to continue rest of program will excute.

**Default Exception Handling by Java:-** **Whenever an exception has occurred inside a method, the method creates an object known as an exception object and transfers it to the runtime system(JVM)**. The exception object includes the exception name and description, and the present program states where an exception happened.

**Example:-**

**public** **class** DefalutException2 {

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

System.***out***.println("m1 method called ");

*m2*();

}

**private** **static** **void** m2() {

System.***out***.println("m2 method called ");

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

}

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

DefalutException2.*m1*();

System.***out***.println(10/0);//

}

}

**Output:-**

m1 method called

m2 method called

Hello

Exception in thread "main" java.lang.ArithmeticException: / by zero

At Exception\_Handling/DefaultException.DefalutException2.main(DefalutException2.java:18)

**Hierarchy of Exception in java:-**

****

**Checked Exception vs Unchecked Exception:-**

|  |  |
| --- | --- |
| Checked Exception | Unchecked Exception |
| Checked exceptions occur at compile time. | Unchecked exceptions occur at runtime. |
| The compiler checks a checked exception. | The compiler does not check these types of exceptions. |
| These types of exceptions can be handled at the time of compilation. | These types of exceptions cannot be a catch or handle at the time of compilation, because they get generated by the mistakes in the program. |
| They are the sub-class of the exception class. | They are runtime exceptions and hence are not a part of the Exception class. |
| Here, the JVM needs the exception to catch and handle. | Here, the JVM does not require the exception to catch and handle. |
| Examples of Checked exceptions:   * File Not Found Exception * No Such Field Exception * Interrupted Exception * No Such Method Exception * Class Not Found Exception | Examples of Unchecked Exceptions:   * No Such Element Exception * Undeclared Throwable Exception * Empty Stack Exception * Arithmetic Exception * Null Pointer Exception * Array Index Out of Bounds Exception * Security Exception |

**Differences between fully checked and partially checked exceptions?**

A Checked Exception is said to be fully checked exception if and only if all its child classes also checked.

**Example:**

IOException

InteruptedException

A Checked Exception is said to be partially Checked exception if and only if some of its child classes are Unchecked.

**Example:**

Exception

**Note:**

The only possible partially checked exceptions in java are

1. Exception

2. Throwable

**Try-Catch Block:-**

* The try statement allows you to define a block of code to be tested for errors while it is being executed.
* In try block write risky code.
* The catch statement allows you to define a block of code to be executed, if an error occurs in the try block.

**Example:-**

public class Test1 {

    public static void main(String[] args) {

        System.out.println("Statement-1");

        try {

        System.out.println(10/0);

        }

        catch (ArithmeticException e) {

            System.out.println(10/2);

        }

        System.out.println("Statement-3");

    }}

**Control flow inside try catch:-**

public class Test {

    public static void main(String[] args) {

        try {

            System.out.println("statemet-1");

            System.out.println(("statemet-2"););

            System.out.println("statement-3");

        } catch (Exception e) {

            System.out.println("statement-4");

        }

        System.out.println("statement-5");

    }

}

**Case-1:-**

If there no exception found the program

**Output:-**

1,2,3,5 statements will be excute.

**Case-2:-**

If an exception raised in statement 2

**Note:-** Inside the try block any have Exception raised even you handle the exception rest of try block never go be execute.

**Output:-**

1, 4, 5 statements will be excute.

**Case-3:-**

If an exception raised in statement-2 but corresponding exception no catch block not matched.

**Output:-**

1 and Exception came(abnormal termination).

**Case-4:-**

If an exception raised at statement-4 and 5

**Output:-**

Abnormal termination

**Methods in Exception:-**

1. e.printStackTrace():-Complete information of exception then go for this method.

It shows Name of exception: Description

Stack Trace

Example:-

public class Methods {

    public static void main(String[] args) {

        try {

            System.out.println(10/0);

        } catch (Exception e) {

            e.printStackTrace();

            System.out.println(e);

        } }}

Output:-

java.lang.ArithmeticException: / by zero

at Exception\_Handling/TryCatch.Methods.main(Methods.java:5)

java.lang.ArithmeticException: / by zero

1. e.toString():-If we want to only name and Description then go for this method.

Example:-

public class Methods {

    public static void main(String[] args) {

        try {

            System.out.println(10/0);

        } catch (Exception e) {

//          e.printStackTrace();

            System.out.println(e);

            System.out.println(e.toString());

        }}}

Output:-

java.lang.ArithmeticException: / by zero

java.lang.ArithmeticException: / by zero

1. System.out.println(e.getMessge):If we want to only Description.

Example:-

public class Methods {

    public static void main(String[] args) {

        try {

            System.out.println(10/0);

        } catch (Exception e) {

//          e.printStackTrace();

//          System.out.println(e);

//          System.out.println(e.toString());

            System.out.println(e.getMessage());

}}}

Output:-

/ by zero

* All the three methods present **Throwable** class.

**One Try Multiple Catch:-**

Case-1

public class TryMultileCatch2 {

    public static void main(String[] args) {

        try {

            System.out.println(10/0);

        }catch (ArithmeticException e) {

                System.out.println("ArithmeticException ");

            }

         catch (Exception e) {

        System.out.println("exception");

}}}

Note:-In this programme no exception will raised. Because in 1st catch blocked exception handle second catch is handle another exception of programme.

Case-2

public class TryMultipleCatch1 {

    public static void main(String[] args) {

        try {

            System.out.println(10/0);

        } catch (Exception e) {

        System.out.println("exception");

        }

    catch (ArithmeticException e) {

        System.out.println("ArithmeticException ");

    }}}

Note:-In this programme exception will raised Because in 1st catch blocked already exception handle second catch unneccesary.

**Finally Block:-**

* The[**finally block**](https://www.geeksforgeeks.org/g-fact-24-finalfinally-and-finalize-in-java/) in java is used to put important codes such as clean up code. The finally block executes whether exception rise or not and whether exception handled or not.
* **e.g.** closing the file or closing the connection.
* **JVM**  will responsible for finally block execute or not.
* Whenever catch block not handle the exception 1st finally block will be excute then throw exception.
* One method is present **System.exit()**
* When ever **System.exit()** called **JVM** will **shutdown** and **finally** block not excute.

class Case1 {

    public static void main(String[] args)

    {

        try {

            System.out.println("inside try block");

                        System.out.println(18 / 2);}

        catch (ArithmeticException e) {

            System.out.println("Arithmetic Exception");

        }

        finally {

            System.out.println(

                "finally : i execute always.");

}}}

Output

inside try block

17

finally : i execute always.

Case 2: When the exception rises and handled by the catch block

class Case2 {

    public static void main(String[] args)

    {

        try {

            System.out.println("inside try block");

            System.out.println(34 / 0);

        }

        catch (ArithmeticException e) {

            System.out.println(

                "catch : exception handled.");

        }

        finally {

          System.out.println("finally : i execute always.");

}}}

Output:-

inside try block

catch : exception handled.

finally : i execute always.

**Case 3:-**When exception rise and not handled by the catch block

class Case3 {

    public static void main(String[] args)

    {

        try {

            System.out.println("Inside try block");

System.out.println(34 / 0);

        }

        catch (NullPointerException e) {

            System.out.println(

                "catch : exception not handled.");

        }

        finally {

            System.out.println(

                "finally : i will execute always.");

        }

       }

}

Output

Inside try block

finally : i will execute always.

Exception in thread "main" java.lang.ArithmeticException: / by zero

    at Case3.main(File.java:10)

**Various Combination try catch finnaly:-**

|  |  |  |
| --- | --- | --- |
| Case-1 | Case-2 | Case-3 |
| **try** {    } **catch** (Exception e) {    } | **try** {    } **catch** (ArithmeticException e) {  // **TODO**: handle exception  }**catch** (Exception e) {  // **TODO**: handle exception  } | **try** {    } **catch** (Exception e) {  // **TODO**: handle exception  }  **catch** (Exception e) {  // **TODO**: handle exception  } |
| **Correct** | **Correct** | **CE:Exception e has alredy caught** |

|  |  |  |
| --- | --- | --- |
| Case-4 | Case-5 | Case-6 |
| **try** {    } **catch** (Exception e) {    }  **finally** {  // **TODO**: handle finally clause  } | **try** {    } **catch** (ArithmeticException e) {  // **TODO**: handle exception  }  **try** {    } **catch** (Exception e) {  // **TODO**: handle exception  } | **try** {    }  **finally** {  // **TODO**: handle finally clause  } |
| **Correct** | **Correct** | **Correct** |

|  |  |  |
| --- | --- | --- |
| Case-7 | Case-8 | Case-9 |
| **try** {    } | **catch** (ArithmeticException e) {  // **TODO**: handle exception  } | **finally** {  // **TODO**: handle finally clause  } |
| **CE:try without catch or finally** | **CE:catch without finally** | **CE:finally without try** |

|  |  |  |
| --- | --- | --- |
| Case-10 | Case-11 | Case-12 |
| **try** {    }  **finally** {  // **TODO**: handle finally clause  }  **catch** (Exception e) {    } | **try** {    }  **try** {    } **catch** (Exception e) {    }  **finally** {  // **TODO**: handle finally clause  } | **try** {    } **catch** (ArithmeticException e) {  // **TODO**: handle exception  }**catch** (Exception e) {  // **TODO**: handle exception  }  **finally** {  // **TODO**: handle finally clause  } |
| **CE:catch without try** | **CE:try without catch or finally** | **Correct** |

|  |  |  |
| --- | --- | --- |
| Case-13 | Case-14 | Case-15 |
| **try** {    } **catch** (Exception e) {  // **TODO**: handle exception  }  **finally** {    }  **finally** {  } | **try** {    }  System.out.println("hii");  **catch** (Exception e) {  // **TODO**: handle exception  } | **try** {    } **catch** (Exception e) {  // **TODO**: handle exception  }  System.out.println("Hii");  **catch** (Exception e) {  // **TODO**: handle exception  } |
| **CE: finally without try** | **CE:try without catch or finally**  **CE:catch without try** | **CE:catch without try** |

|  |  |  |
| --- | --- | --- |
| Case-16 | Case-17 | Case-18 |
| **try** {    } **catch** (Exception e) {  // **TODO**: handle exception  }  System.out.println("Hii");  **finally** {    } | **try** {**try** {    } **catch** (ArithmeticException e) {  // **TODO**: handle exception  }  }**catch** (Exception e) {  // **TODO**: handle exception  } | **try** {**try** {    } **catch** (Exception e) {  // **TODO**: handle exception  }  }**catch** (Exception e) {  // **TODO**: handle exception  } |
| **CE: finally without try** | **Correct** | **CE:catch without try** |

|  |  |  |
| --- | --- | --- |
| Case-19 | Case-20 | Case-22 |
| **try** {**try** {    }    }  **catch** (Exception e) {  // **TODO**: handle exception  } | **try** {    } **catch** (Exception e) {  // **TODO**: handle exception **try** {    } **catch** (Exception e2) {  // **TODO**: handle exception  }  } | **try** {    } **catch** (Exception e) {  // **TODO**: handle exception  }  **finally** {**try** {    } **catch** (Exception e) {  // **TODO**: handle exception  }  **finally** {  System.out.println("hii");  }  } |
| **CE:try without catch or finally** | **Correct** | **Correct** |

|  |
| --- |
| Case-22 |
| **try**  System.out.println("try");  **catch** (Exception e)  System.out.println("catch");  **finally**  System.out.println("Finnaly"); |
| **CE: compulsory using curly braces** |

UserDefined Exception:-

* Java provides opportunity to the program to create own exception.
* It is mandatory requirement that the class should be sub class of exception class

Throw:-

* **Throw** keyword use for to handover our created exception object to the jvm manually for that purpose we can use **throw** keyword.
* This keyword used to throw an exception from a method on any block of code.We can throw both checked and unchecked exception.

Example:-

 class TestThrow1 {

    public static void validate(int age) {

        if(age<21) {

            //throw Arithmetic exception if not eligible to vote

            throw new ArithmeticException("Person is not eligible for Bajaj Card");

        }

        else {

            System.out.println("Person is eligible to vote!!");

        }

    }

    public static void main(String args[]){

        validate(13);

        System.out.println("rest of the code...");

  }

}

**OutPut:-**

Exception in thread "main" java.lang.ArithmeticException: Person is not eligible for Bajaj Card

at TestThrow1.validate(Test.java:6)

at TestThrow1.main(Test.java:15)

**Case-1:-**

**public** **class** Throw\_e\_Test3 {

**static** ArithmeticException *e* =**new** ArithmeticException();

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

**throw** *e*;

}

}

**Output:-**

Exception in thread "main" java.lang.ArithmeticException

at Exception\_Handling/Throw\_and\_Throws.Throw\_e\_Test3.<clinit>(Throw\_e\_Test3.java:4)

**Case-2:-**

**public** **class** Throw\_e\_Test3 {

**static** ArithmeticException *e* ;

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

**throw** *e*;

}

}

**Output:-**

Exception in thread "main" java.lang.NullPointerException: Cannot throw exception because "Throw\_and\_Throws.Throw\_e\_Test4.e" is null

at Exception\_Handling/Throw\_and\_Throws.Throw\_e\_Test4.main(Throw\_e\_Test4.java:6)

**Note: if refers null throw exception null pointer exception**

Throws Exception:-

* The throws is added to the method signature to late the caller no about the exception the called method can through. It responsibility of caller to the handle the exception using (or try catch block) throws.
* It use for only checked Exception.

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **Final** |  | **Finally** |  | | **finalize** | |  | |
| 1 | It is keyword |  | it is Block |  | | It is a method | |  | |
| 2 | It is applicable with  variable,method,class |  | it is use with try catch block or try block |  | | It is a method can override for any object | |  | |
|  | | |  | |  | |  | |