Date	16 November 2022
Team ID	PNT2022MID17351
Project Name	Iot Based Smart Crop Protection System For Agriculture

Exception Handling

Dealing Errors

Error is the wrongs that can make a program go wrong.

An error may produce an incorrect output or may terminate the execution of the program or even may cause the system to crash.

Types of Errors

Errors are of two types:

1. Compile-time error

2. Run time error

1. Compile-time error

All syntax errors will be detected and displayed by the java compiler. S these errors are known as compile-time errors. Whenever the compiler displays an error, it will not create the .class file. So it is necessary that we fix all the errors before we can successfully compile and run the program.

Most of the compile time errors are due to spelling mistakes.The most common problems are :

Missing semicolon

Missing brackets in class and methods

Misspelling of identifiers and keywords

Missing double quotes in strings

Use of undeclared variables

And so on

2. Run time error

Sometimes , a program may compile successfully creating the .class file but may not run properly. Such programs may produce wrong results due to wrong logic. Most common run time errors are :

Dividing integer by zero

Accessing an element that is out of bounds of an array

Accessing a character that is out of bounds of a string
Trying to store a value into the array of an incompatible class or type
Attempting to use a negative size for an array

And many more

When such errors are encountered, java generates an error message and aborts the program

Exceptions

An Exception is a condition that is caused by a run time error in the program. When java interpreter encounters an error such as dividing an integer by zero, it create an exception object and throws it then take the correct action.This task is known as exception handling.

This mechanism performs following tasks :

1. Find the problem (Hit the exception).
2. Inform that an error has occurred (Throw the exception).
3. Receive the error information (Catch the exception).
4. Take corrective actions (Handles the exception).

The error handling code consists of two segments, one to detect errors and to throw exceptions and the other to catch exceptions and to take appropriate actions.

Exception Types

Built-in exceptions are the exceptions which are available in Java libraries. These exceptions are suitable to explain certain error situations. Below is the list of important built-in exceptions in Java.

It is thrown when an exceptional condition has occurred in an arithmetic operation.

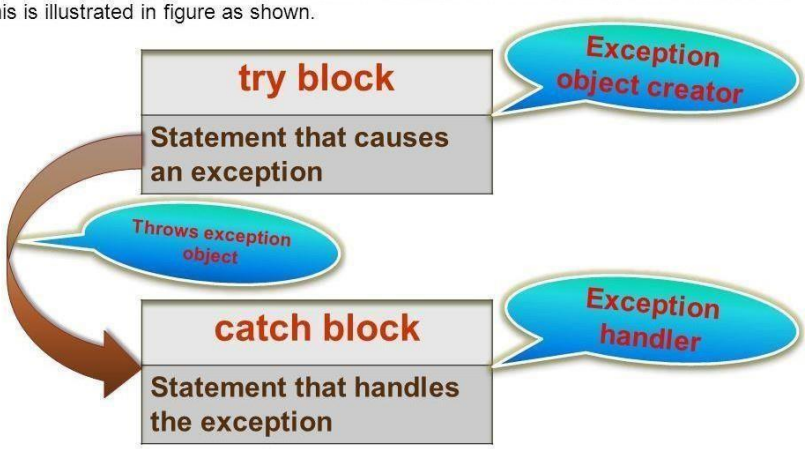
1. **ArrayIndexOutOfBoundsException**
   It is thrown to indicate that an array has been accessed with an illegal index. The index is either negative or greater than or equal to the size of the array.
2. **ClassNotFoundException**
   This Exception is raised when we try to access a class whose definition is not found
3. **FileNotFoundException**
   This Exception is raised when a file is not accessible or does not open.
4. **IOException**
   It is thrown when an input-output operation failed or interrupted
5. **InterruptedException**
   It is thrown when a thread is waiting , sleeping , or doing some processing , and it is interrupted.
6. **NoSuchFieldException**
   It is thrown when a class does not contain the field (or variable) specified
7. **NoSuchMethodException**
   It is thrown when accessing a method which is not found.

8. **NullPointerException**
   This exception is raised when referring to the members of a null object. Null represents nothing
9. **NumberFormatException**
   This exception is raised when a method could not convert a string into a numeric format.
10. **RuntimeException**
    This represents any exception which occurs during runtime.
11. **StringIndexOutOfBoundsException**
    It is thrown by String class methods to indicate that an index is either negative than the size of the string

Syntax of Exception Handling Code

Syntax of Exception Handling

The basic concepts of exception handling are throwing an exception and catching it. This is illustrated in figure as shown.



Java uses a keyword try to write a block of code that is likely to cause an error condition and "throw" an exception.

Catch block is defined by the keyword catch ,it catches the exception thrown by the try block and handle it properly

The catch block is added immediately after the try block.

Ex.

```
…….. ……..
try
{
Statement ;              // generates an exception
}
catch(Exception-type e)
{
Statements ;             // processing the exeption
}
…….. ……..
```

```
// Program to demonstrate exception handling mechanism
// Java program to demonstrate ArithmeticException class
Demo
{
   public static void main(String args[])
   { try
      { int a = 30, b = 0;
         int c = a/b;              // cannot divide by zero
         System.out.println ("Result = " + c);
      }
      catch(ArithmeticException e)
      {
         System.out.println ("Can't divide a number by 0");
```

```
    }
  }
}
```

Output:
Can't divide a number by 0

Creating our Own Exception

In java we can create our own exception class and throw that exception using throw keyword. These exceptions are known as **user-defined** or **custom** exceptions. In this tutorial we will see how to create your own custom exception and throw it on a particular condition.

To create our own exception :

first we have to create a class for exception which is extends from Exception class.
Then create another class having main method.
Now throw the new created exception class in the try block.

```
// program to demonstrate creating our own exception
        class MyException extends Exception
        { int a;
          MyException(int b)
        {
            a=b;
          }
          public String toString()
        {
            return ("Exception Number = "+a) ;
         }
        }

        class JavaException
        {
          public static void main(String args[])
        { try
        {
             throw new MyException(2);
             // throw is used to create a new exception and throw it.
         }
         catch(MyException e)
        {
           System.out.println(e) ;
         }
        }
        }
```

Output:

Exception Number = 2