<!--Estudio Shonos-->

Exceptions;

<By=" Team Number 2"/>

}

Exceptions {

An exception is an abnormal situation that can occur
when we run a certain program. The way the programmer
treat it is what is generally known as management or management of the exception.
an exception is an abnormal situation that can occur
when we run a certain program. The way the programmer
treat it is what is generally known as management or management of the exception.

Some cases of anomalous situations that can be cited are:

- call a method on a "null" object.
- Try to divide a
- number by “0”.
- try to open a file that does not exist to read it.

}

Exception´s Constructors{

`Exception()`

First parameterless constructor

`Exception (String message)`

A second constructor, with a parameter of “String” type, which allows us to create the exception with a specific message.

The first thing that should be clear is that in Java, all exceptions that we can use or create our own, they must inherit from the "Exception" class of the java library

The use of exceptions in Java allows for error handling and control in a structured way, preventing the program from stopping abruptly and providing a form of proper error handling or recovery

}

Example of exception{

```java
package exception.example;

/**
 *
 * @author Adonny Calero,Jsons,DCCO-ESPE
 */
public class Exception{
    public static void main(String[] args) {
        try {
            int dividend = 10;
            int divider = 0;
            int result = dividend / divider; // We try to divide by zero, which throws an exception
            System.out.println("The result of the division is: " + result); // This line will not be executed
        } catch (ArithmeticException e) {
            System.out.println("An arithmetic error occurred: " + e.getMessage()); // We catch and display the exception message
        }
    }
}

```

}

About the example {

In the example an attempt is made to divide by zero, which is an invalid operation. This situation generates an exception of type “ArithmeticException”. The line of code "int result = dividend / divisor;" is where the exception is thrown.

- To handle this exception, we use a try-catch block. The try block contains the code that might throw an exception. In this case, the split operation is inside the try block. If an exception occurs inside the try block, it jumps to the corresponding catch block

In the catch block, you specify the type of exception you want to catch. In this example, we use catch(ArithmeticException e), which means that we are catching exceptions of type ArithmeticException. Inside the catch block, we can perform actions to handle the exception, such as displaying an error message

- In the example, trying to divide by zero throws an ArithmeticException exception, which is caught by the catch block. Inside the catch block, the message "An arithmetic error occurred: / by zero" is returned, using the getMessage() method of the exception object (e) to get the specific message of the exception.

}

<!--Estudio Shonos-->

Thanks {

<By="Team number 2"/>

}