Everything You Should Know About Apex Exceptions in Salesforce

Share the information with your friends

Why use exceptions?

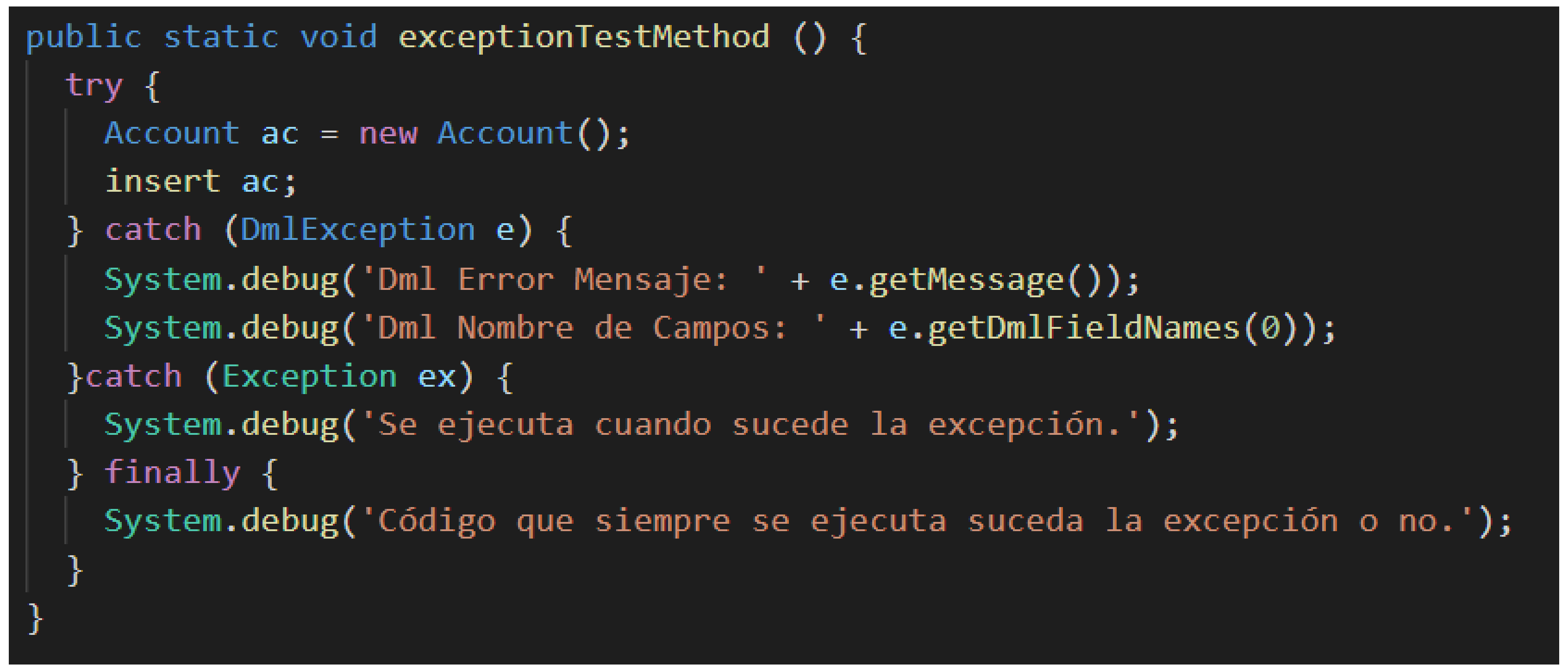
Exceptions are caused by errors encountered when executing code that interrupts its normal flow of execution.

They are used to recover from errors, manage them and ensure proper code execution.

Types of Exceptions

Native Exceptions:

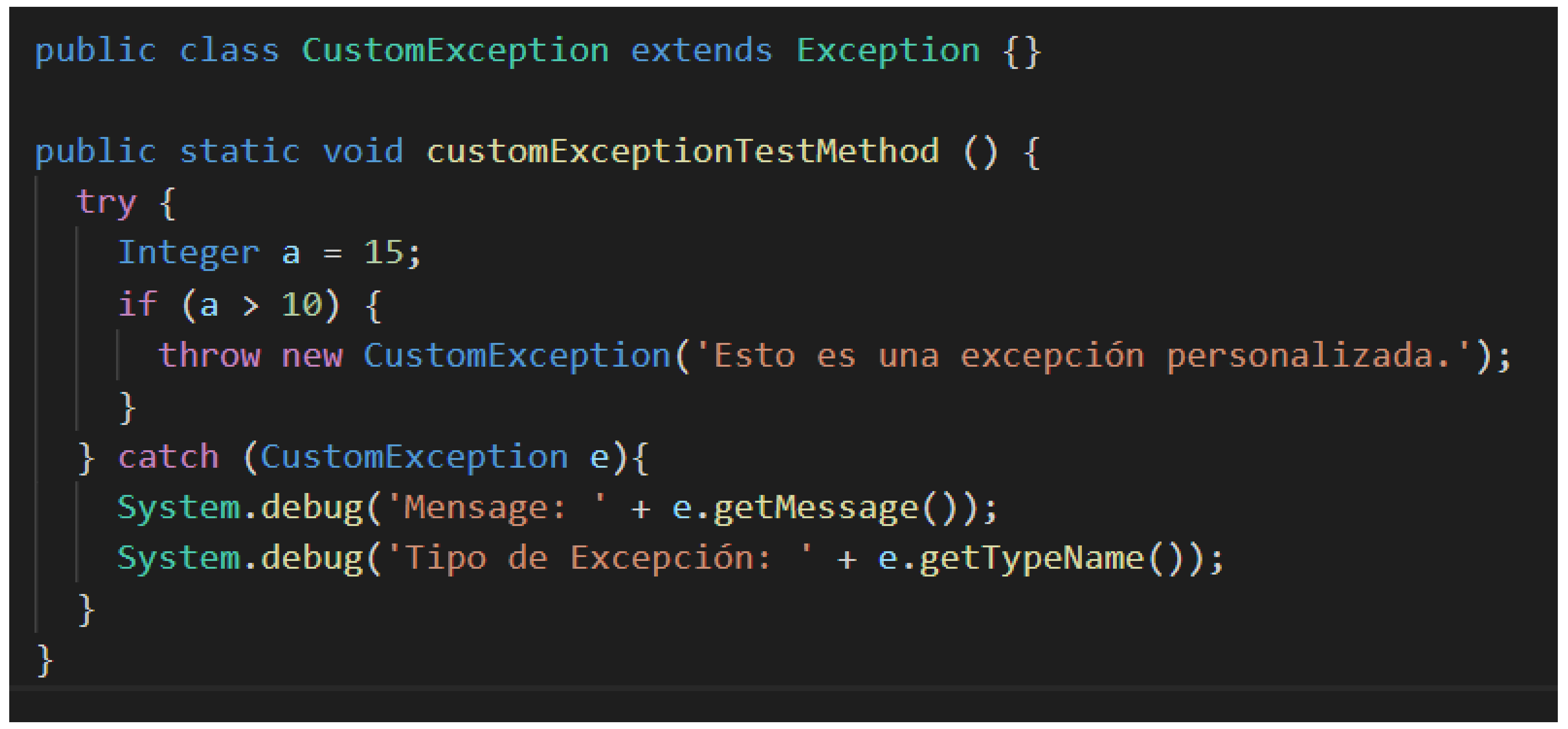
- Apex offers a native framework for handling exceptions by default.
- They happen when the flow of the program is automatically interrupted by the system.
- There can be multiple "Catch" blocks. However, the generic exception must be the last one.
- The Finally block is executed when the exception occurs or not. In addition, it is used regularly to clean variables or code.



```
public static void exceptionTestMethod () {
  try {
    Account ac = new Account();
    insert ac;
  } catch (DmlException e) {
    System.debug('Dml Error Mensaje: ' + e.getMessage());
    System.debug('Dml Nombre de Campos: ' + e.getDmlFieldNames(0));
  }catch (Exception ex) {
    System.debug('Se ejecuta cuando sucede la excepción.');
  } finally {
    System.debug('Código que siempre se ejecuta suceda la excepción o no.');
  }
}
```

Custom Exceptions:

- To interrupt the flow of the program and handle errors in any part of the code.
- To handle exceptions in more detail.
- To create a custom exception, it must be extended to the "Exception" class.
- The name of our custom class must end with the word "Exception". For example: "PurchaseException" or "CustomException".



```
public class CustomException extends Exception {}

public static void customExceptionTestMethod () {
  try {
    Integer a = 15;
    if (a > 10) {
      throw new CustomException('Esto es una excepción personalizada.');
    }
  } catch (CustomException e){
    System.debug('Mensage: ' + e.getMessage());
    System.debug('Tipo de Excepción: ' + e.getTypeName());
  }
}
```

Exceptions Methods:

Common Methods

- **getTypeName ():** Returns the type of exception.
- **getMessage ():** To get the error message.
- **getCause ():** Returns the cause of the exception.

- **getLineNumber ():** To return the line where the exception occurs.

Other Methods

- **getDmlFieldNames:** Name of the fields causing the error.
- **getDmlId:** Id of the failed records that cause the error.
- **getDmlMessage:** The message error for a specific failed registration.
- **getNumDml:** Returns the number of failed records.

Exceptions Considerations

Manageable Exceptions:

- **addError ():** To reverse DML operations and return the error in a trigger.
- **ApexPages.message (msg):** To display errors on a visualforce page.
- **Class "Messaging":** To return errors via e-mail.

Unmanageable Exceptions:

- **LimitExceptions** cannot be manageable.
- The **errors of Assertion** may not be manageable.