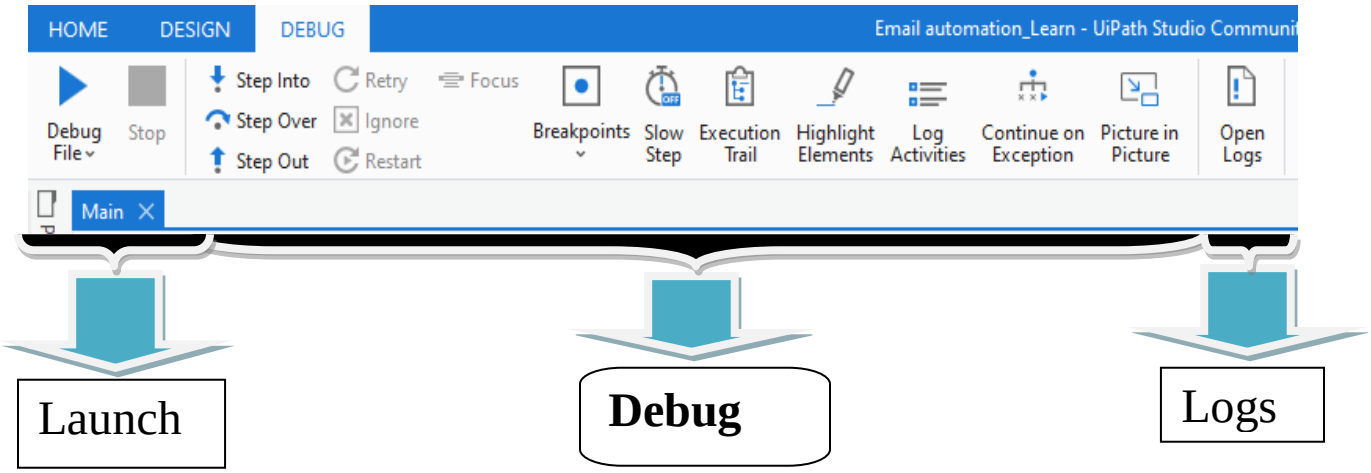
ERROR & EXCEPTION HANDLING

DEBUGGING IN UIPATH:

The Execute Tab has mainly 3 options



The **Launch** Section has 3 options:

- **Run**: Run is used to Execute the Project.
- **Stop**: Stop is used to stop the Execute your Project.
- **Debug**: Debug Option Is used to start Debugging the Project.

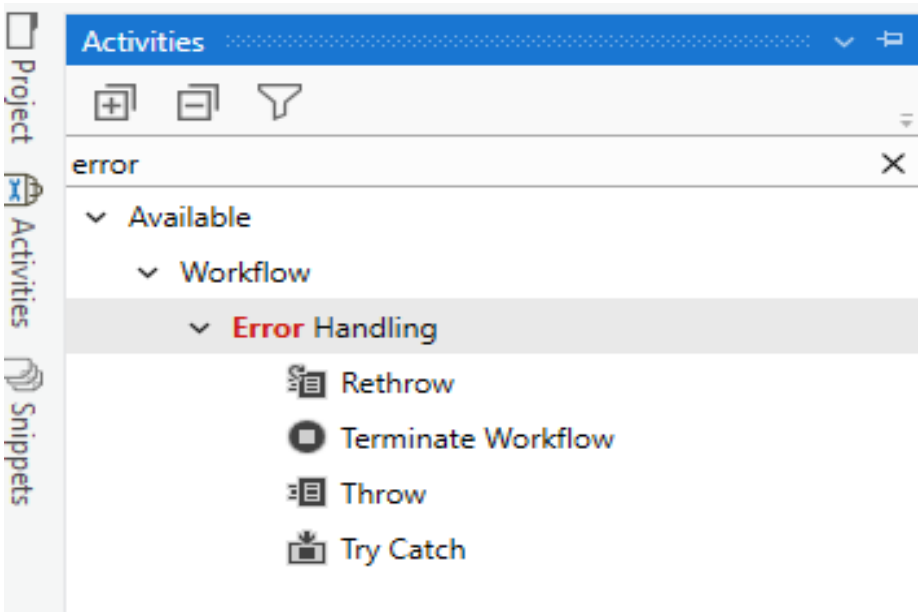
The **Debug** Section has 8 options:

1. **Step into/Step over/Step out**: This option is used to check the step by step execution.
2. **Break point**: used to define a breakpoint while debugging the project.
3. **Slow Step**: Reduce the speed of Execution
4. **Execution Trail**: When enabled, it shows the exact execution path at debugging. As the process is executed, each activity is highlighted and marked in the **Designer** panel, showing you the execution as it happens.
5. **Highlight:** If enabled, UI elements are highlighted during debugging. The option can be used both with regular and step-by-step debugging.

6. **Log Activities:** If enabled, debugged activities are displayed as Trace logs in the **Output** panel. Note that Highlight Elements and Log Activities options can only be toggled before debugging, and persist when reopening the automation project. This is not applicable for invoked workflows unless these files are opened in the Designer panel.
7. **Continue on Exception:** This debugging feature is disabled by default. When disabled in the ribbon, it throws the execution error and stops the debugging, highlights the activity which threw the exception, and logs the exception in the Output panel. If a Global Exception Handler was previously set in the project, the exception is passed on to the handler.
8. **Picture in Picture**: If enabled, the process starts in a separate session whenever you select **Run** or **Run File**, **Debug** or **Debug File**. If **Picture in Picture** is disabled, debugging and execution is performed in the current session.

Open Logs: Used to open the execution logs of the project.

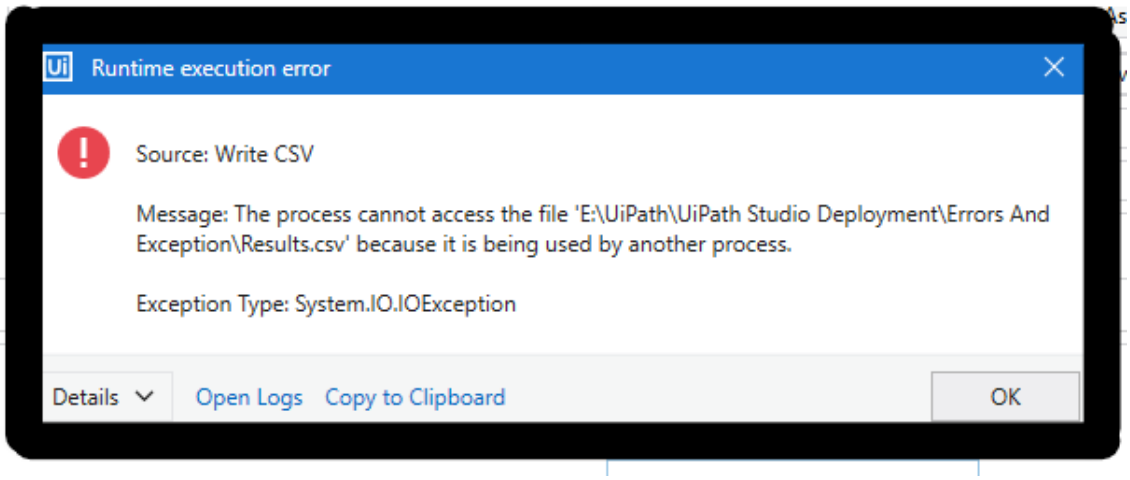
Exception handling in UiPath:



The Error Handling activity provides four choices i.e.

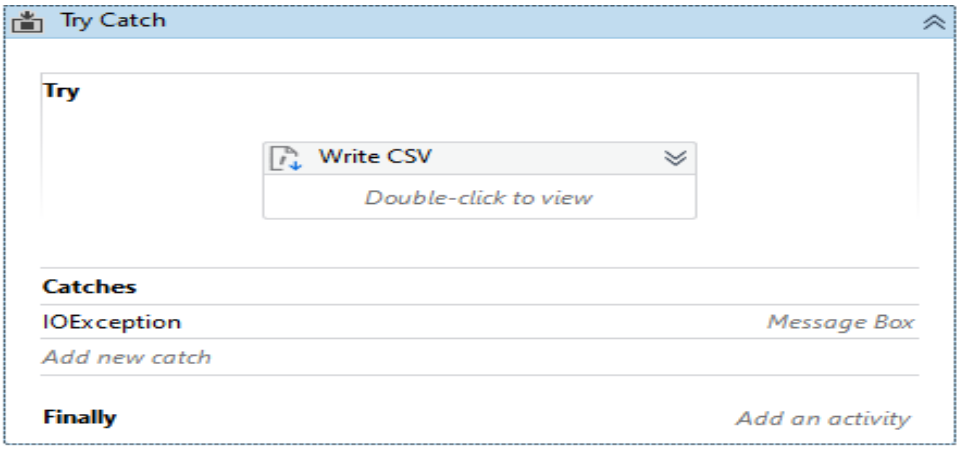
1. **1. Try Catch:** When you want to test anything and manage any exceptions that arise, you utilize activity. So, everything you want to test may go in the try Catch part, and if an error happens, it can be handled in the try catch block based on your input to the catch section. Aside from the try-catch, we have a finally section where we list the actions that must be completed.
2. **Rethrow:** UiPath to make sure activities occur before exception is thrown.
3. **Throw:** Used to Simulate an Exception.
4. **Terminate Workflow:** workflow is used to terminate the workflow the moment the task encounters an error.

1. TryCatch Error: When your files will be open on execute the project, it will show an error your files open.



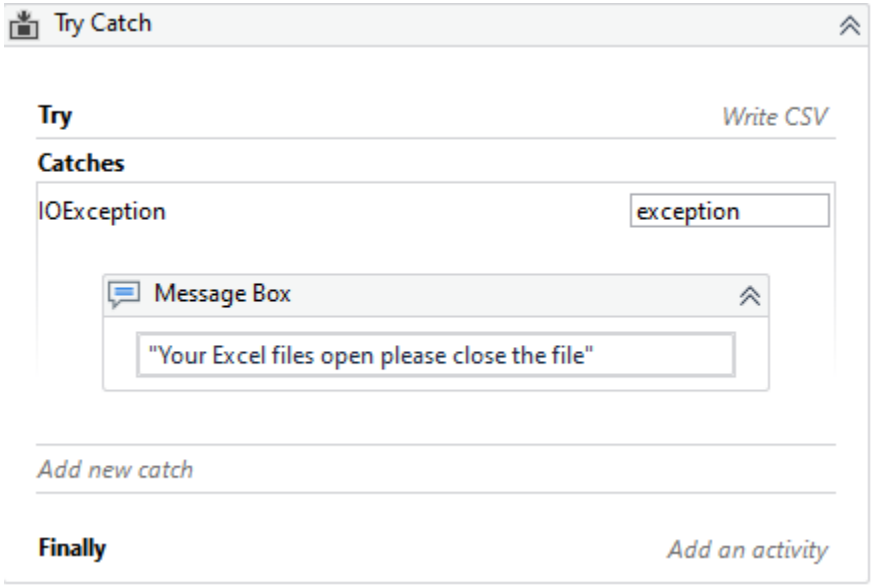
Solution:

1. Drag and drop the Try Catch activity
2. Write CSV files into the try section.



3. Catch block:

- Exception: Systeam.IO.Exception
- Add the message box: "Your Excel file open please close the file"

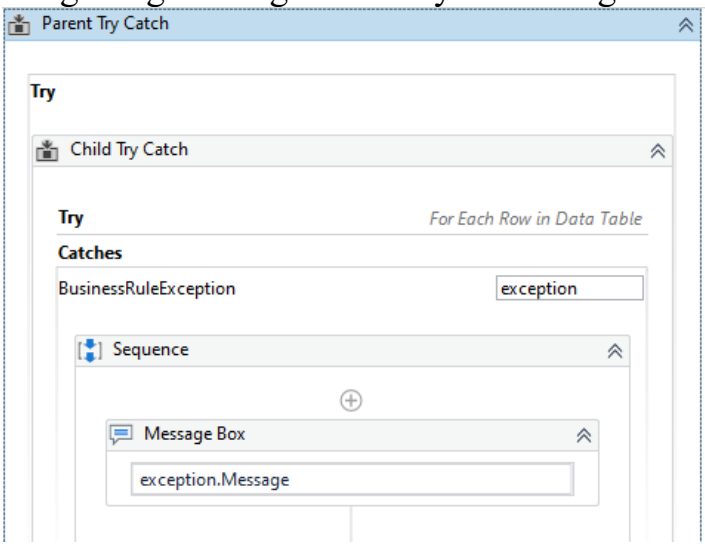


"Try Catch" activity demands a at least one catch to be implemented.

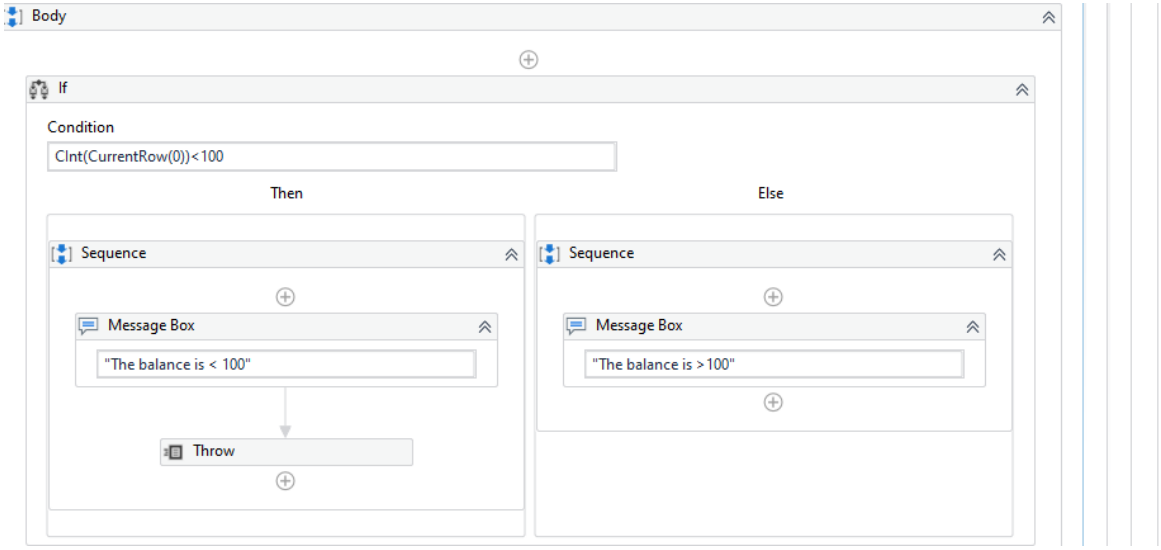
THROW & RETHROW ACTIVITY:

We are here going to handle two types of exceptions, One is generic exceptions and another being a null reference exception.

A generic exception and whenever it occurs we will suppress it and will log it using "Log Message" Activity or Message box"



We will implement, A null reference exception and whenever it occurs we will try to throw it.

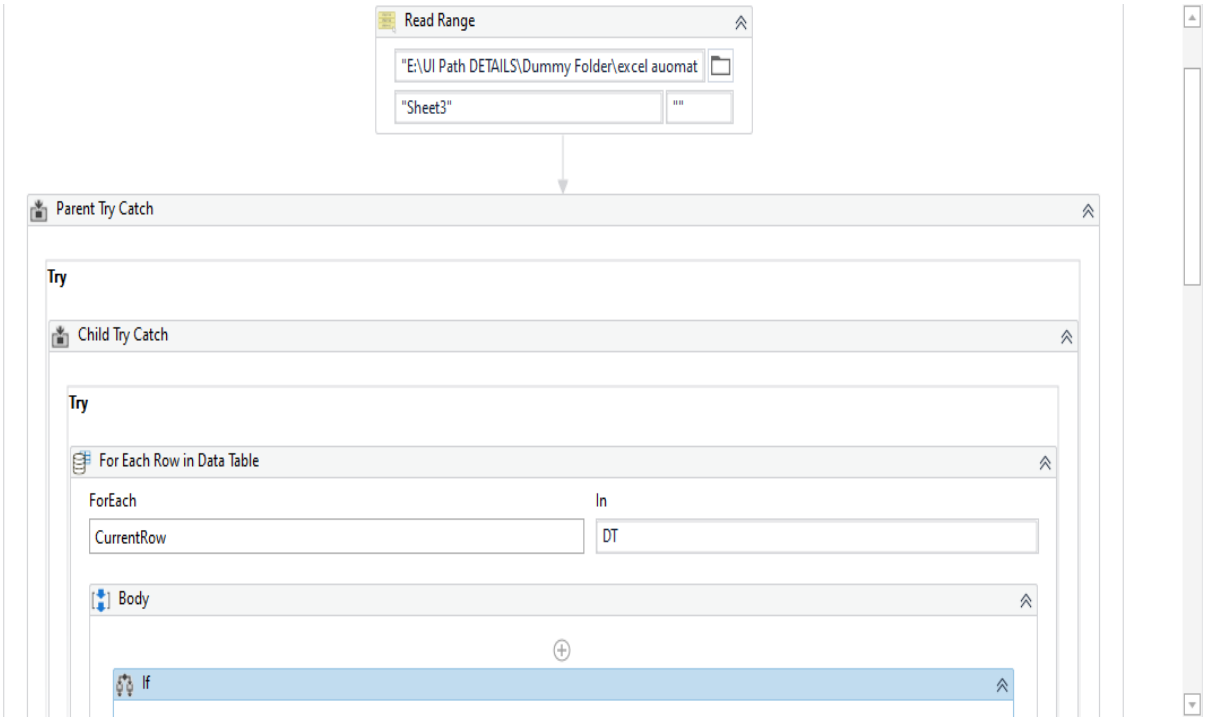


Use Of UiPath For Implementation:

1. For the sake of this example, we will build a robot, which we will create an Excel sheet that the name of Amount.

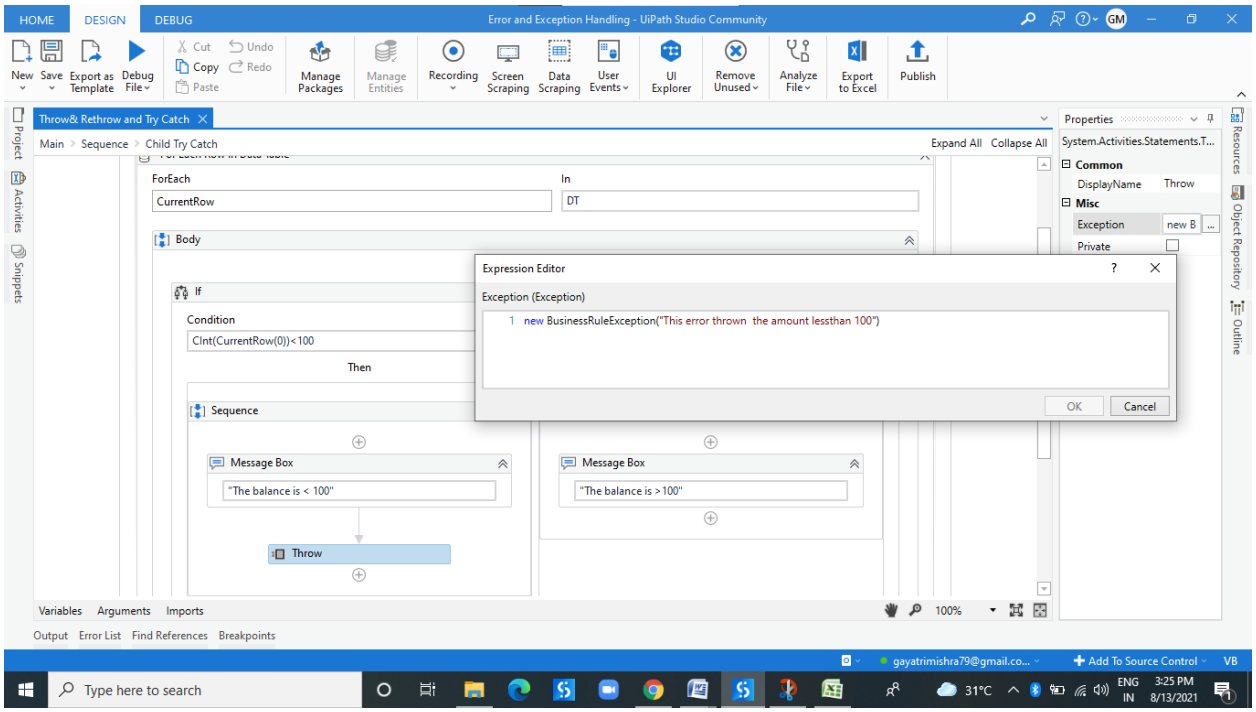
A	B	C
ammount		
100		
200		
50		
500		
85		
400		
101		

2. Use the Read Range (Workbook) Activity to read the entire book.
3. Drag and drop the Try Catch activity & for each row activity into the try block, each data read and take is used for execution.

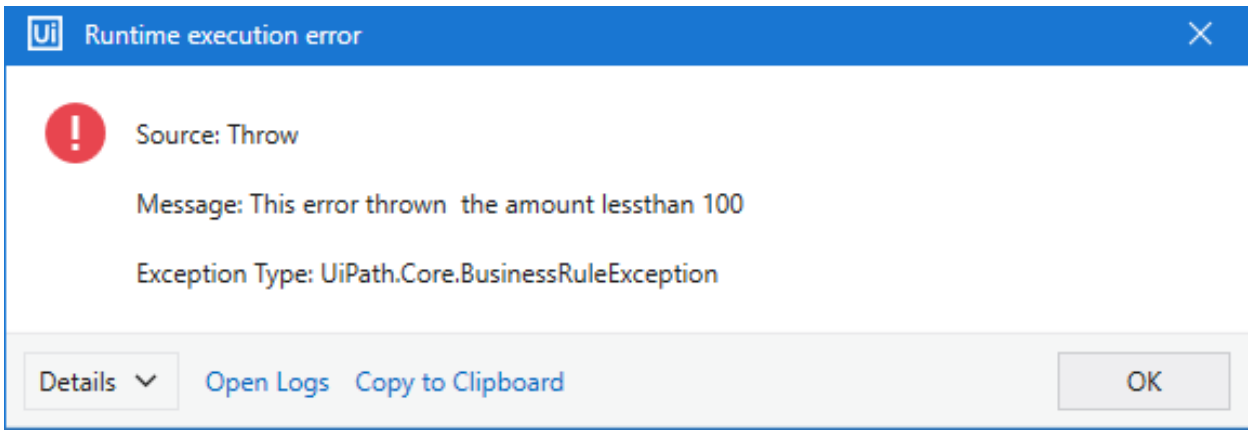


4. We implemented the IF activity for condition (If your amount is less/greater than the 100, then the message will be displayed).

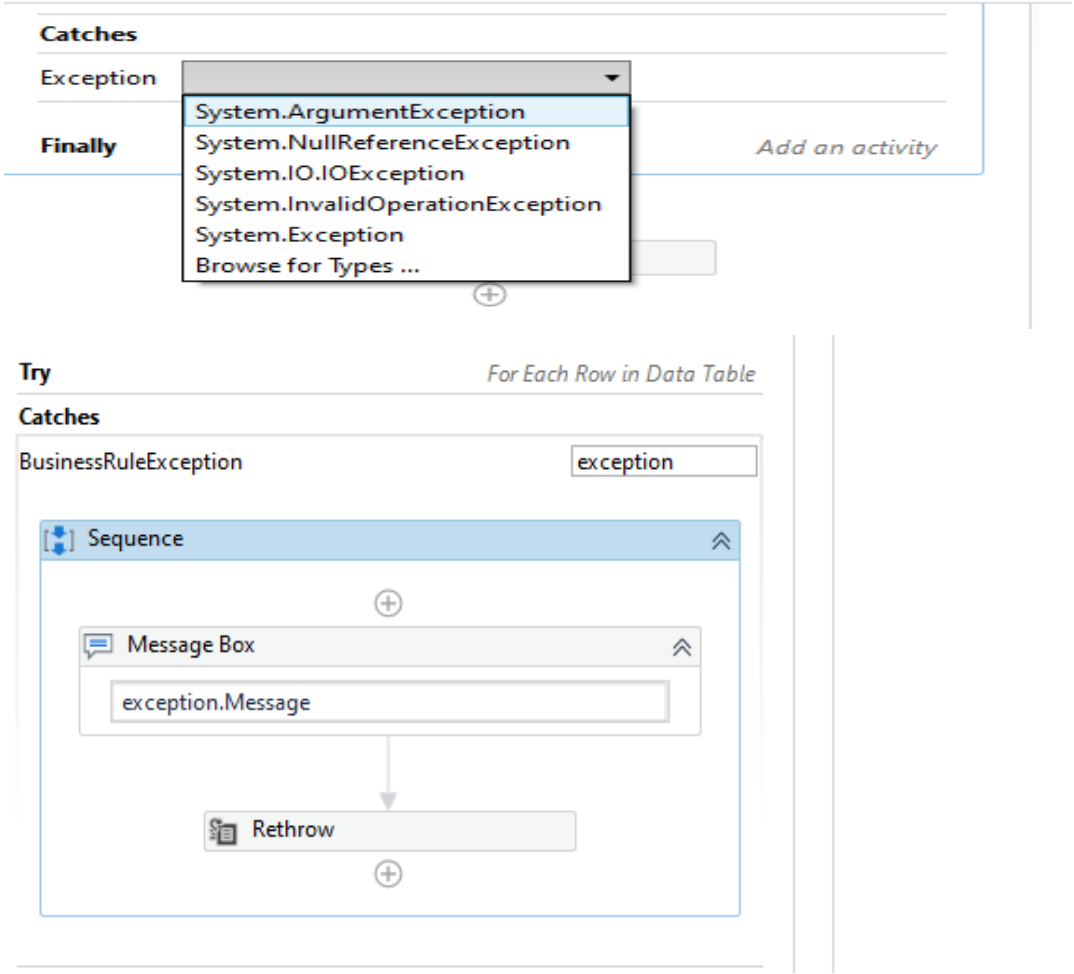
Condition: Cint (current row(0))<100.



- When Your Amount is less than 100, then it will throw the your custom message.



- In Catch Block, It then shows the Exception Type.



- ❖ Throw activity in UiPath is to generate your own exception. It can be system Exception or Business Exception.
- ❖ Rethrow activity in UiPath takes an existing exception that has been encountered and regenerated it at a high-level. It is used in catch block only.
- You have three options to prevent making such mistakes.
    - Open the main window/application and close all other windows.
    - If you want to simulate a click action, alter the Input Method to Simulate Click.
    - You can ensure that the correct selection values are present so that UiPath is directed to the correct window and the automation is performed.

There are quiet high numbers of exceptions you can observe in UiPath, It depends on the scenarios, most common are:

- **System.IO.FileNotFoundException**: If file is not there on specified path.
- **UiPath.Core.SelectorNotFoundException** is one of the common exceptions when it comes to automating user interfaces.
- **System.NullReferenceException**: This error usually occurs when using a variable with no set value.
- **System.ArgumentException**, **System.ArgumentNullException**
  The exception that is thrown when one of the arguments of invoked workflow is not valid, null respectively.

REFERENCE:

1. https://www.youtube.com/c/UiPathExpoHub
2 . https://www.edureka.co/blog/error-handling-in-UiPath.
3. https://excelcult.com/how-to-use-rethrow-in-UiPath

For any RPA Implementation/Resources in your Organization please reach out to rpa@gxplabs.com