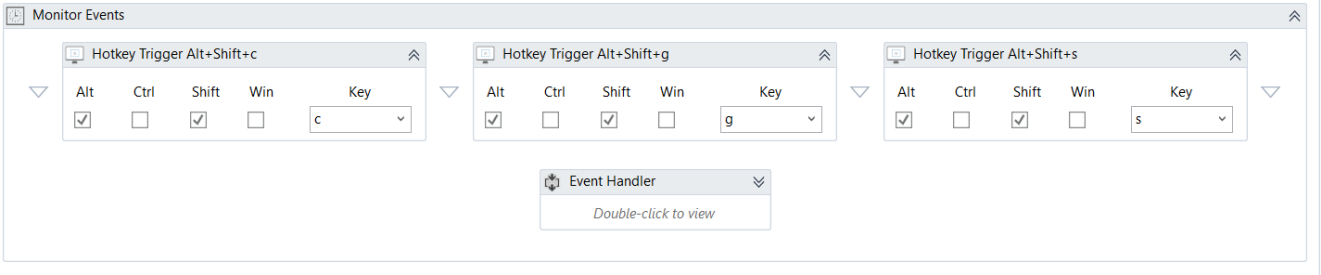
To exemplify how to use **Monitor Events**, **Hotkey Trigger** and **Get Event Info** activities, we implemented an automation with following functionality:

- Robot should continuously listen to hotkeys sent by user and perform following actions:
  - Alt+Shift+c → Open Calculator application (and continue to monitor other events)
  - Alt+Shift+g → Copies the text previously selected by user and searches it on Google. (and continue to monitor other events)
  - Alt+Shift+s → Stops monitoring events. (Other activities placed after MonitorEvents activity should be performed)

This is how the demo was built:

1. Open Studio and start new Process
2. Create Boolean variable (ContinueMonitor) and assign it the value True.
3. Drag a Monitor Events activity. We want the robot to listen to 3 user triggers:
   a. Alt+Shift+c
   b. Alt+Shift+g
   c. Alt+Shift+s

For this, we should add 3 Hotkey Trigger activities in the main container of Monitor Events activity, as shown below.



4. Set RepeatForever option as ContinueMonitor. If the Boolean variable remains True, the block executes every time is activated. If the Boolean variable becomes False, the activity executes only once.
5. Drag a Get Event Info activity and set its result as TriggerHotkey – a UiPath.Core.EventInfo variable.
   At runtime, the TriggerHotkey will store the information of the event which triggered the EventHandler, which will help us find which of the 3 triggers was activated.
6. Drag a Switch activity. Set TypeArgument as String and Expression as:

   TriggerHotkey.KeyEventInfo.KeyName.ToLower

   This expression returns the KeyName of the event which triggered the handler – in our case, this is the only differentiator, as KeyModifiers are the same in all three cases: Alt+Shift.

7. Click on “Add new case” and add “c” as “Case value” (without quotation marks).

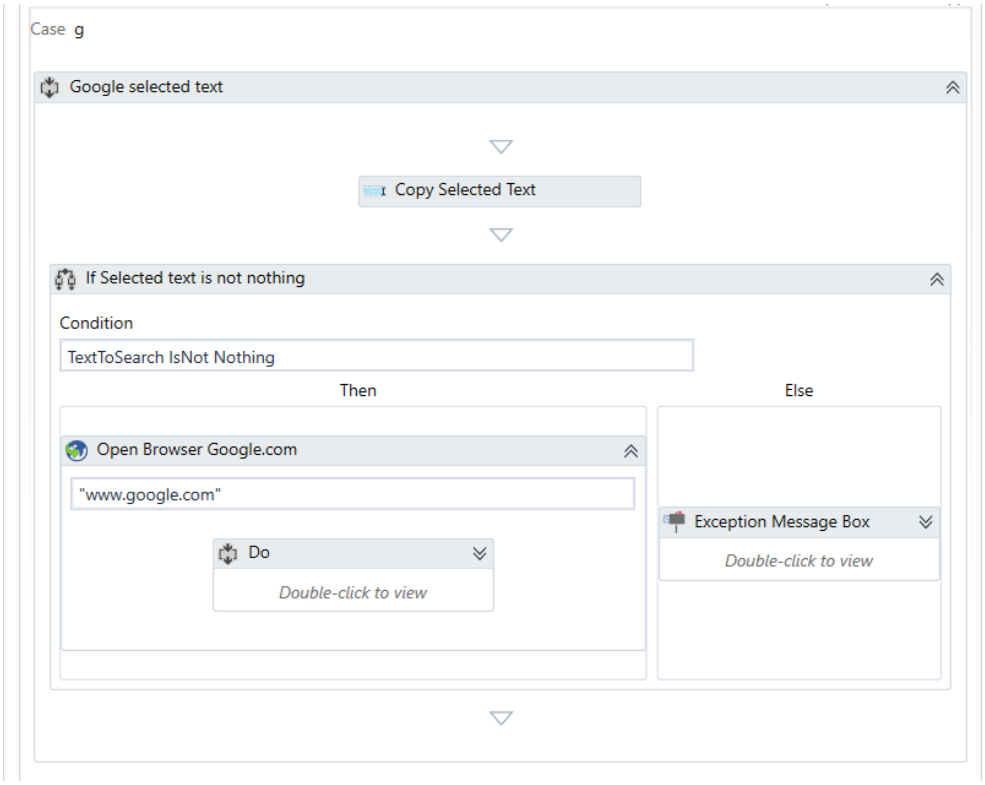
a. Drag an Open Application activity.
b. Set Input > Arguments as “calc.exe” and add a suitable selector for this application.

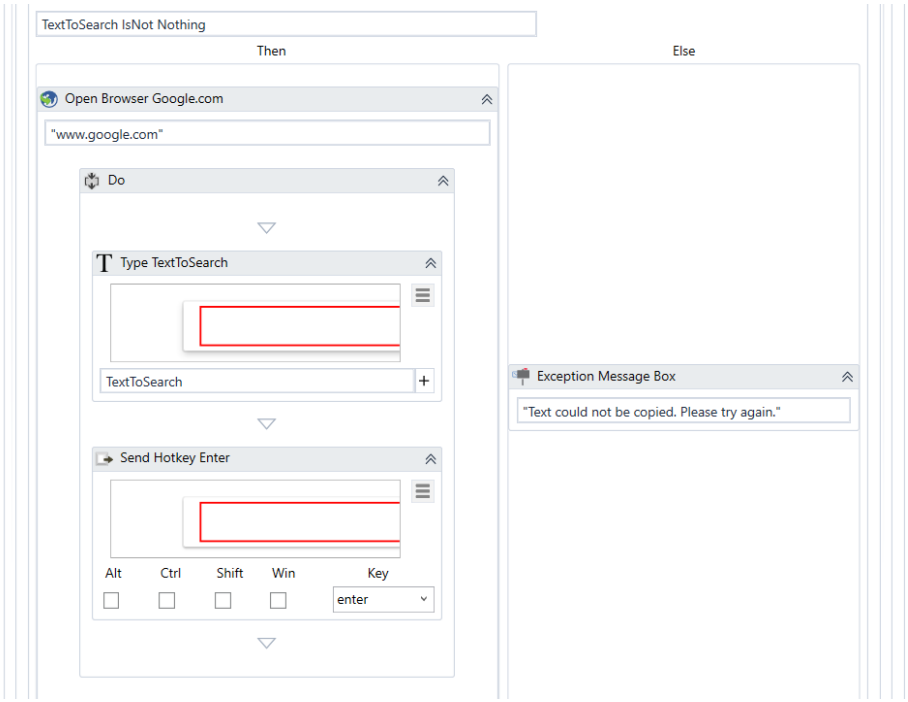
8. Add new case with value “g”.
   a. Drag Copy Selected Text. Assign its result to a string variable, in our case: TextToSearch.

   b. Because the user may forget to select the text before sending the hotkey, or may select text which cannot be copied, we set ContinueOnError to True and Timeout to 2000 (2s).

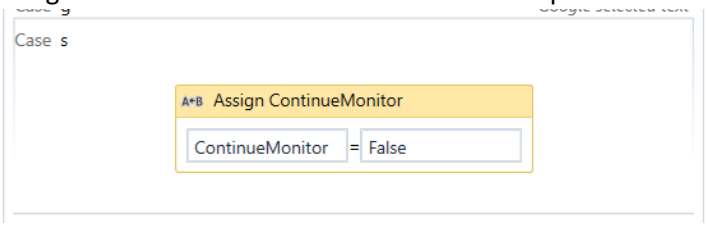
   c. Drag an If activity and set the condition to: TextToSearch IsNot Nothing

      Note: TextToSearch Scope should be the sequence for case “g”.
   d. For Then: Add an Open Browser activity and search the text selected.
   e. For Else: Add a message box.





9. Add new case with value “s”
    a. Assign False to ContinueMonitor – this will stop MonitorEvents execution.



10. This is how our workflow should look like: