Consultingwerk
software architecture and development

ABL Structured Error Handling

PUG CHALLENGE EXCHANGE AMERICAS

Mike Fechner, Director, Consultingwerk Ltd.
mike.fechner@consultingwerk.de

Consultingwerk Ltd.

- Independent IT consulting organization
- Focusing on **OpenEdge** and **.NET**
- Located in Cologne, Germany
- Customers in Europe, North America, Australia and South Africa
- Vendor of tools and consulting programs
- 25 years of Progress experience (V5 … OE11)
- Specialized in GUI for .NET, OO, Software Architecture, Application Integration

Agenda

- Traditional/Classic Error Handling recap
- Structured Error Handling
- Combining Structured and Classic Err. Handling
- STOP Conditions
- Error Handling best practices

Runtime „conditions“

- (Runtime) **Error**
- Application **Error**
- Stop Conditions
- Quit Conditions

Classic Error Handling

- Block oriented error handling
- Error handling directives (ON ERROR …)
- Intuitively used by experienced ABL developers
- Made more sense in TTY data entry

- NO-ERROR option on a number of statement
- ERROR-STATUS System Handle
- RETURN ERROR <return value>
- NO-UNDO for variables and temp-tables

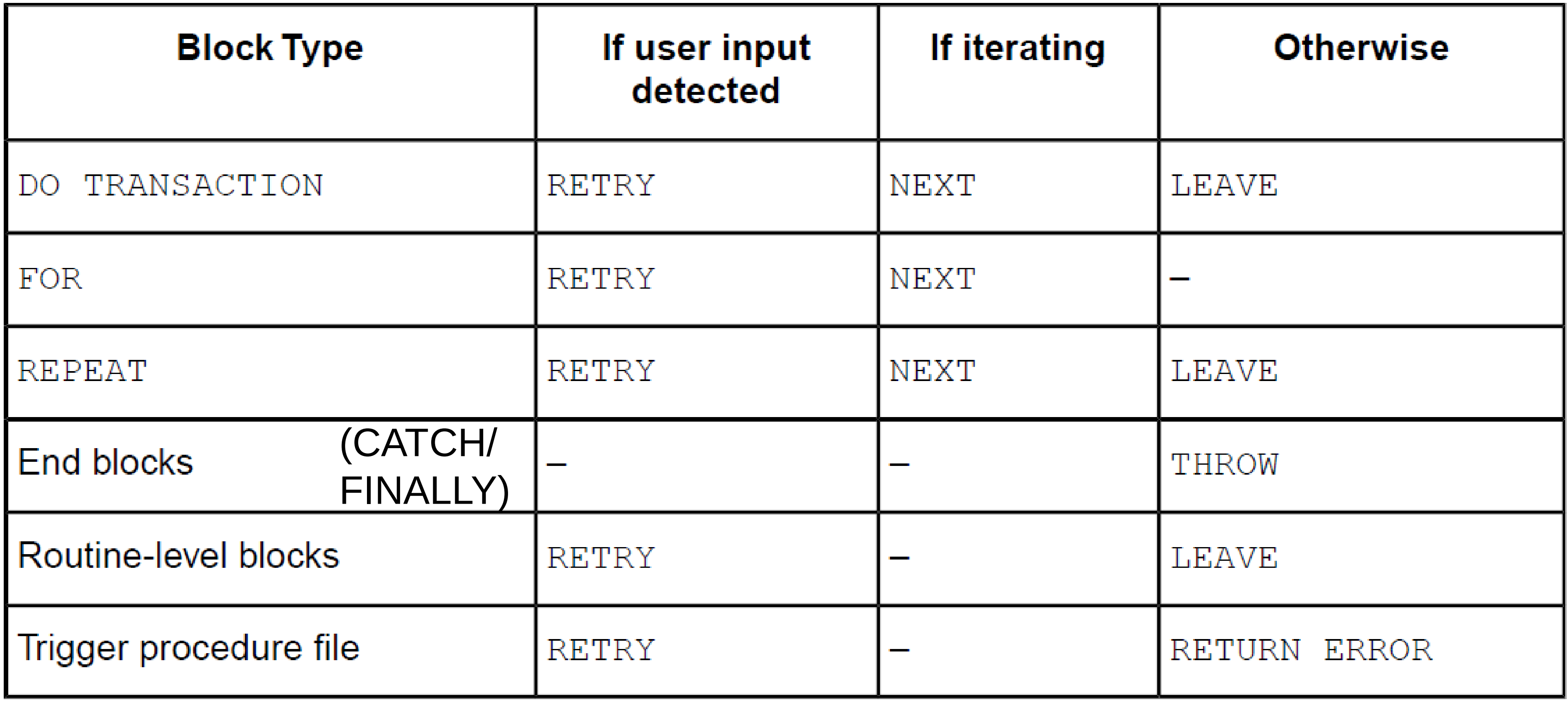
Error handling sequence in a block

-
- Stops execution of current block
- Display error message
- Undo transaction or sub-transaction (if present)
- Branch action (RETRY, LEAVE, NEXT)

Consultingwerk
software architecture and development

Default branching options per block type

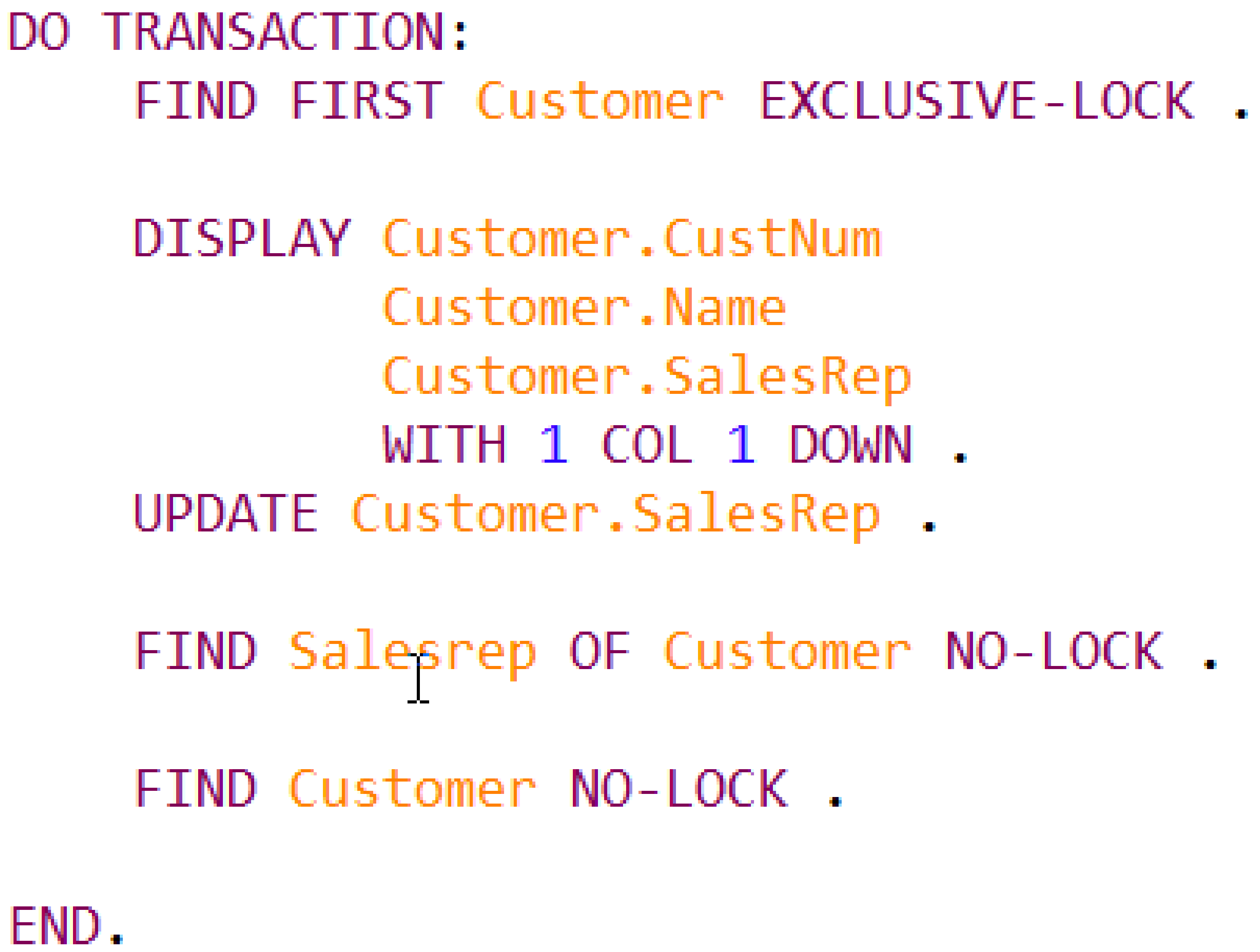
Table 2: Default branching on error



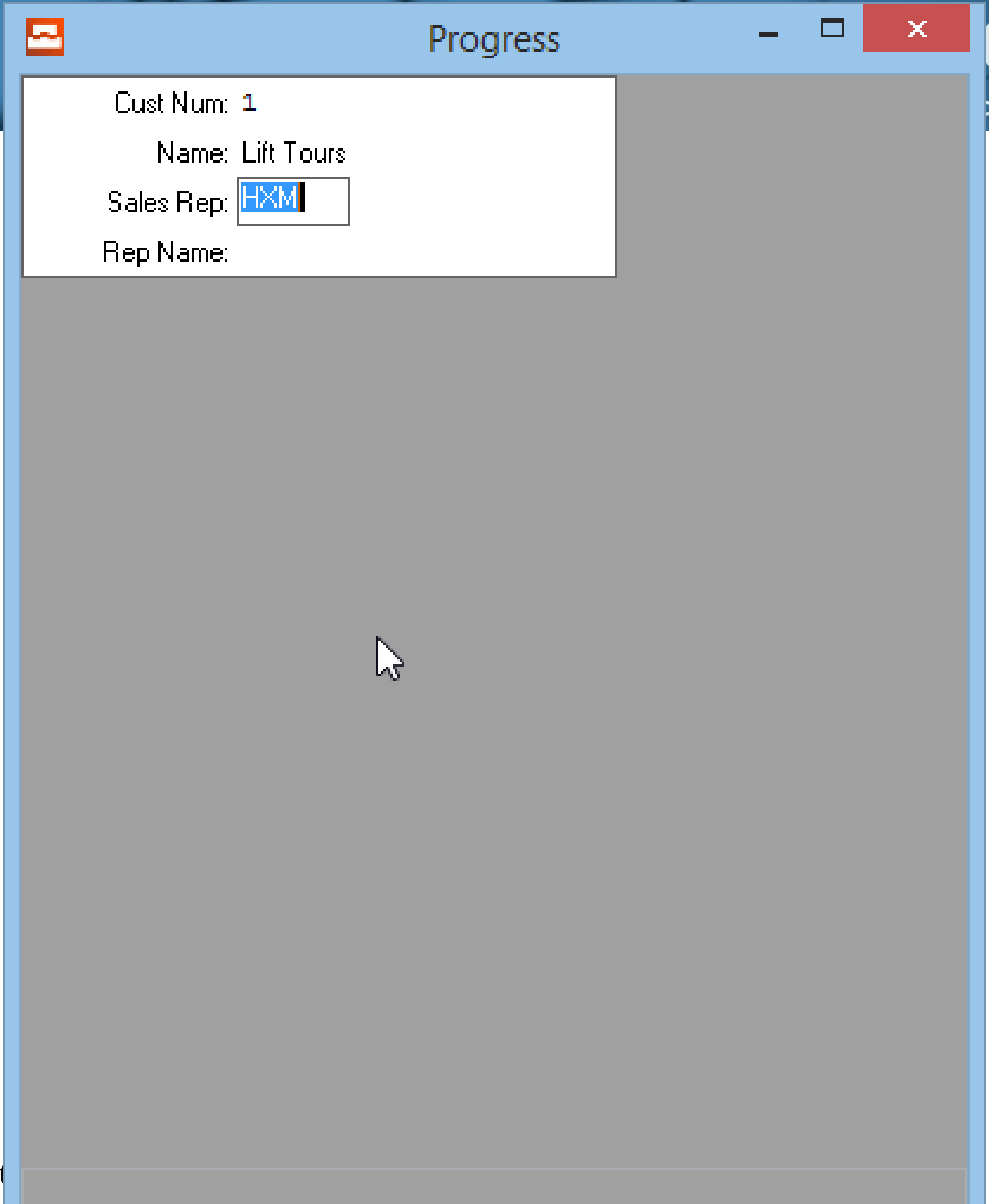
Block Type	If user input detected	If iterating	Otherwise
DO TRANSACTION	RETRY	NEXT	LEAVE
FOR	RETRY	NEXT	–
REPEAT	RETRY	NEXT	LEAVE
End blocks (CATCH/ FINALLY)	–	–	THROW
Routine-level blocks	RETRY	–	LEAVE
Trigger procedure file	RETRY	–	RETURN ERROR

Routine-level blocks

- .p file, .w file
- Internal procedure
- User-defined function
- Database trigger
- User-interface trigger
- Class method, including
  - Constructor
  - GET/SET
  - Destructor



```
DO TRANSACTION:
    FIND FIRST Customer EXCLUSIVE-LOCK .

    DISPLAY Customer.CustNum
            Customer.Name
            Customer.SalesRep
            WITH 1 COL 1 DOWN .
    UPDATE Customer.SalesRep .

    FIND Salesrep OF Customer NO-LOCK .

    FIND Customer NO-LOCK .

END.
```

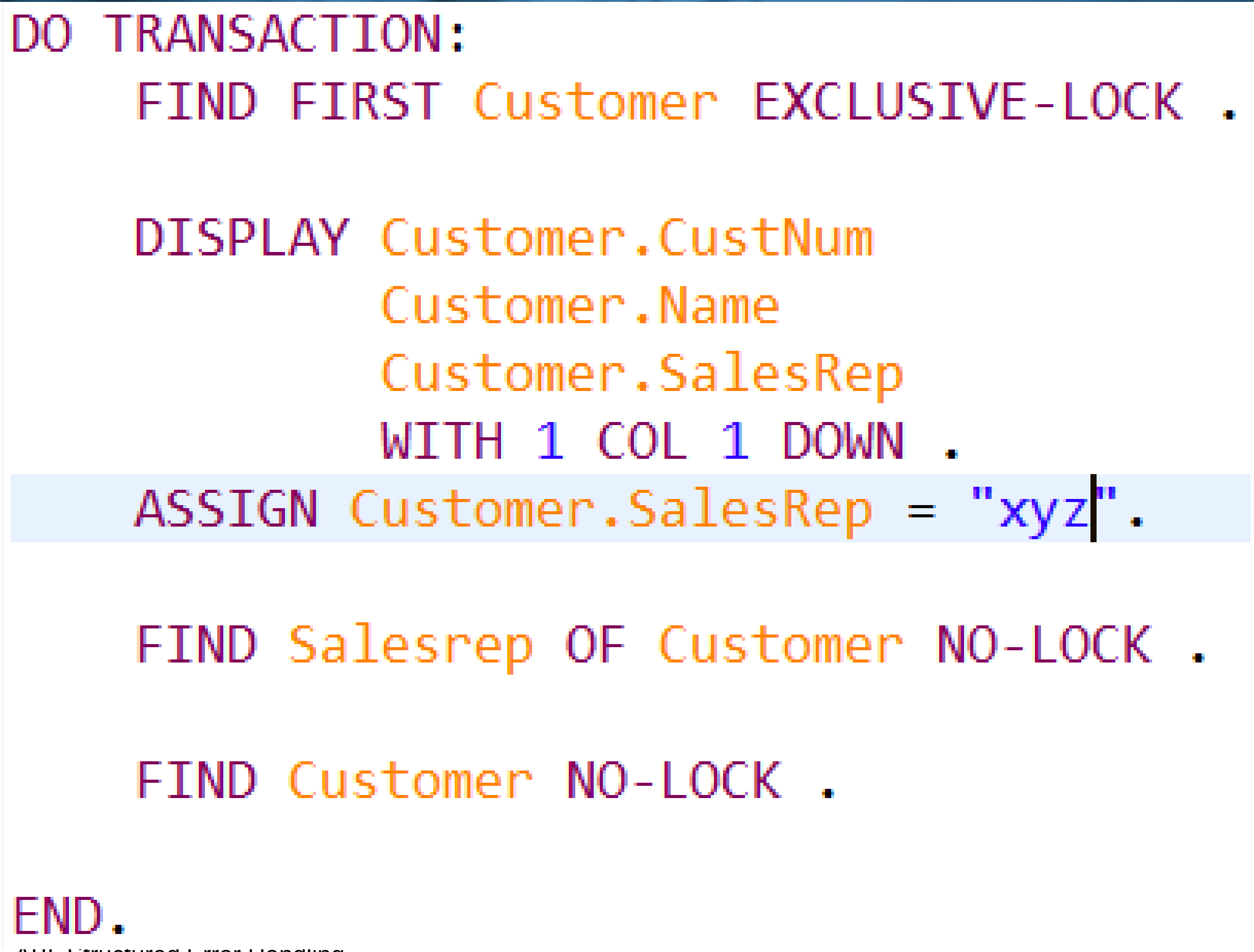


Cust Num: 1

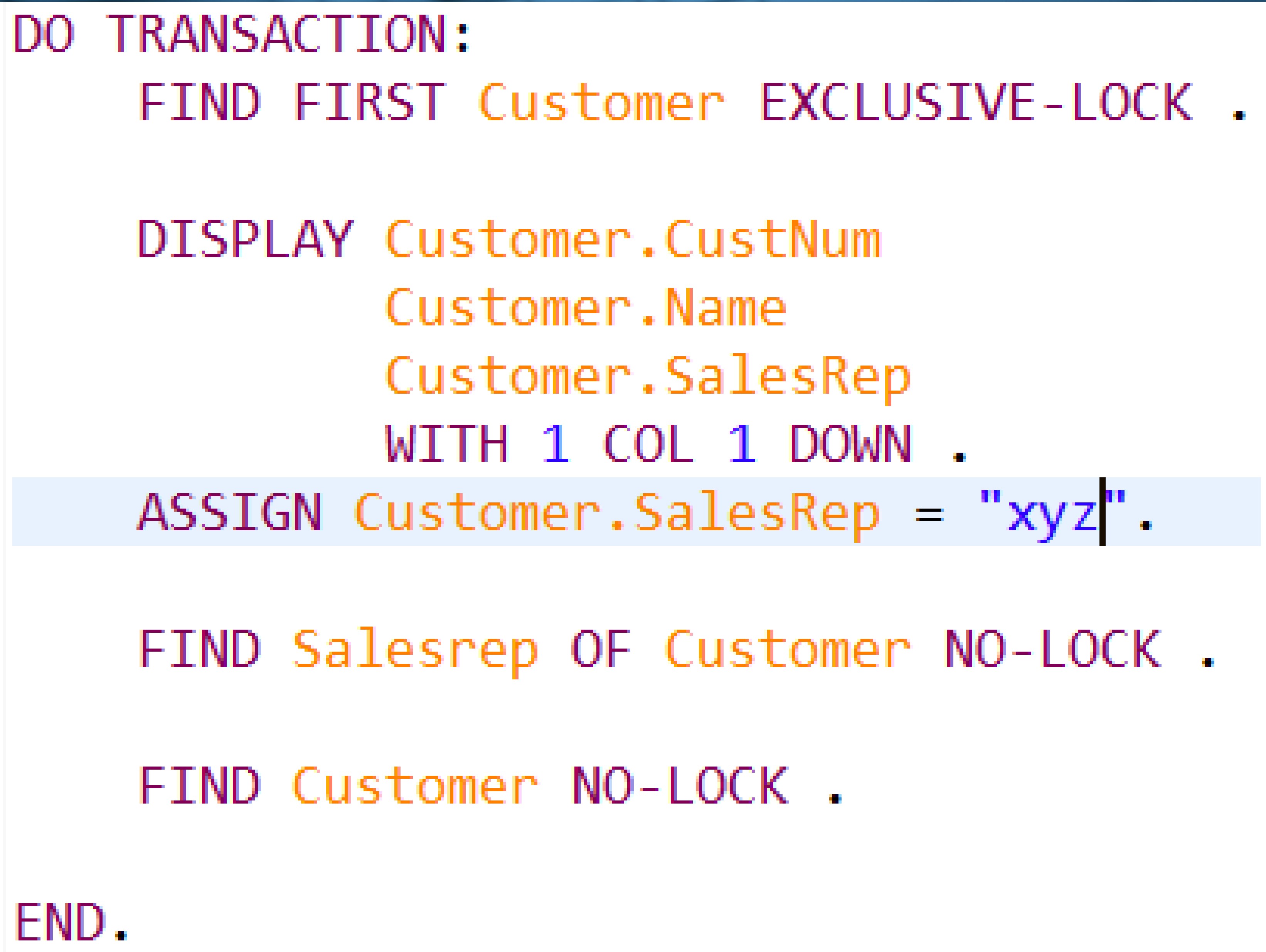
Name: Lift Tours

Sales Rep: HXM

Rep Name:



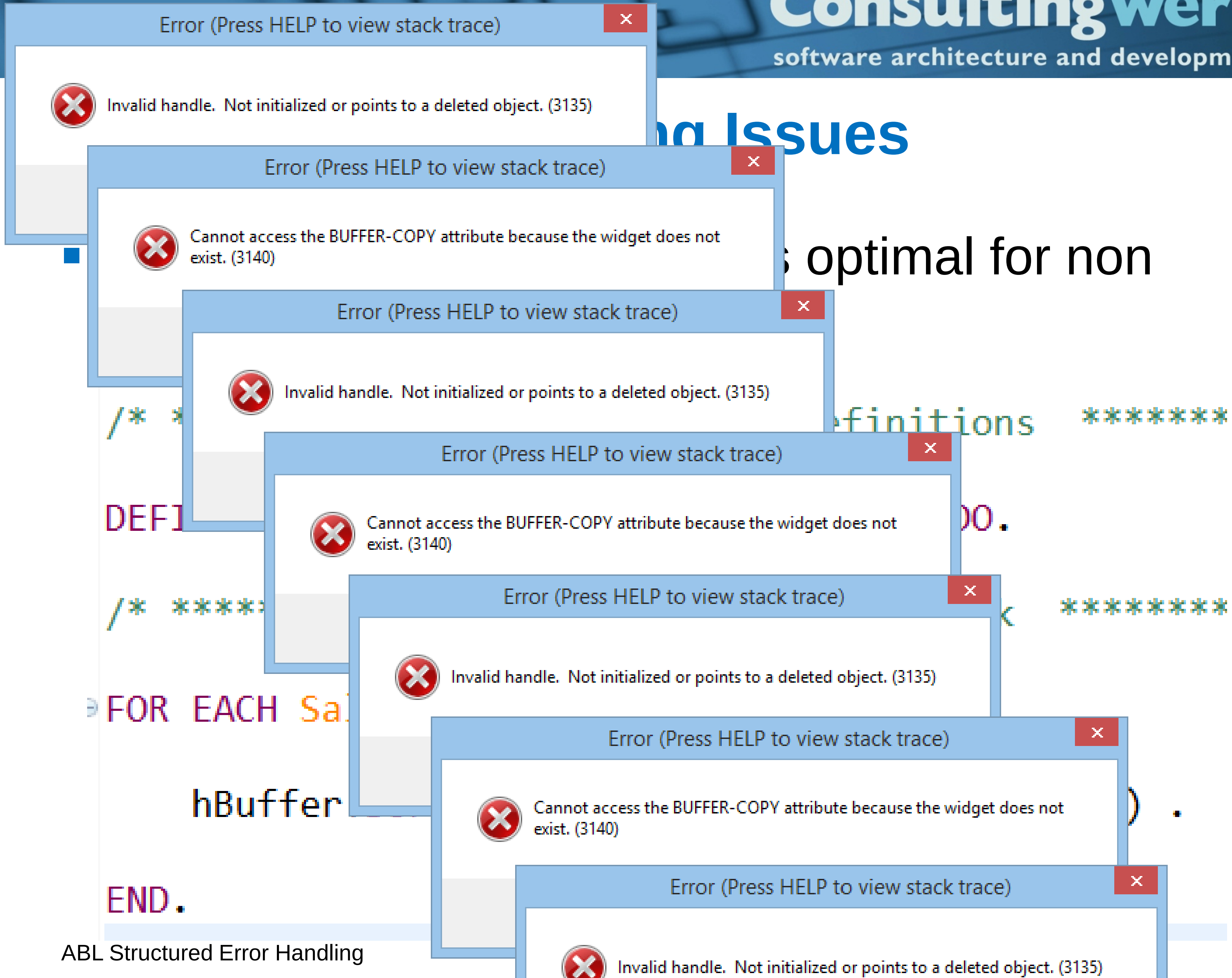
```
DO TRANSACTION:
    FIND FIRST Customer EXCLUSIVE-LOCK .

    DISPLAY Customer.CustNum
            Customer.Name
            Customer.SalesRep
            WITH 1 COL 1 DOWN .
    ASSIGN Customer.SalesRep = "xyz".

    FIND Salesrep OF Customer NO-LOCK .

    FIND Customer NO-LOCK .

END.
```



```
DO TRANSACTION:
    FIND FIRST Customer EXCLUSIVE-LOCK .

    DISPLAY Customer.CustNum
            Customer.Name
            Customer.SalesRep
            WITH 1 COL 1 DOWN .
    ASSIGN Customer.SalesRep = "xyz".

    FIND Salesrep OF Customer NO-LOCK .

    FIND Customer NO-LOCK .

END.
```

g Issues

- s optimal for non



```
/* *                    finitions   ********

DEFI                                OO.

/* *****                            *   ********

FOR EACH Sa

    hBuffer                              )   .

END.
```

Error (Press HELP to view stack trace)

Invalid handle. Not initialized or points to a deleted object. (3135)

Error (Press HELP to view stack trace)

Cannot access the BUFFER-COPY attribute because the widget does not exist. (3140)

Error (Press HELP to view stack trace)

Invalid handle. Not initialized or points to a deleted object. (3135)

Error (Press HELP to view stack trace)

Cannot access the BUFFER-COPY attribute because the widget does not exist. (3140)

Error (Press HELP to view stack trace)

Invalid handle. Not initialized or points to a deleted object. (3135)

Error (Press HELP to view stack trace)

Cannot access the BUFFER-COPY attribute because the widget does not exist. (3140)

Error (Press HELP to view stack trace)

Invalid handle. Not initialized or points to a deleted object. (3135)

Classic Error Handling Issues

- Calling statements with NO-ERROR leaves ERROR-STATUS behind
- ERROR-STATUS:ERROR and RETURN-VALUE need to be reset by
  - ERROR-STATUS:ERROR = FALSE NO-ERROR
  - RETURN "" .
- Otherwise confusing if (pending) error was already handled or not

Agenda

- Traditional/Classic Error Handling recap
- Structured Error Handling
- Combining Structured and Classic Err. Handling
- STOP Conditions
- Error Handling best practices

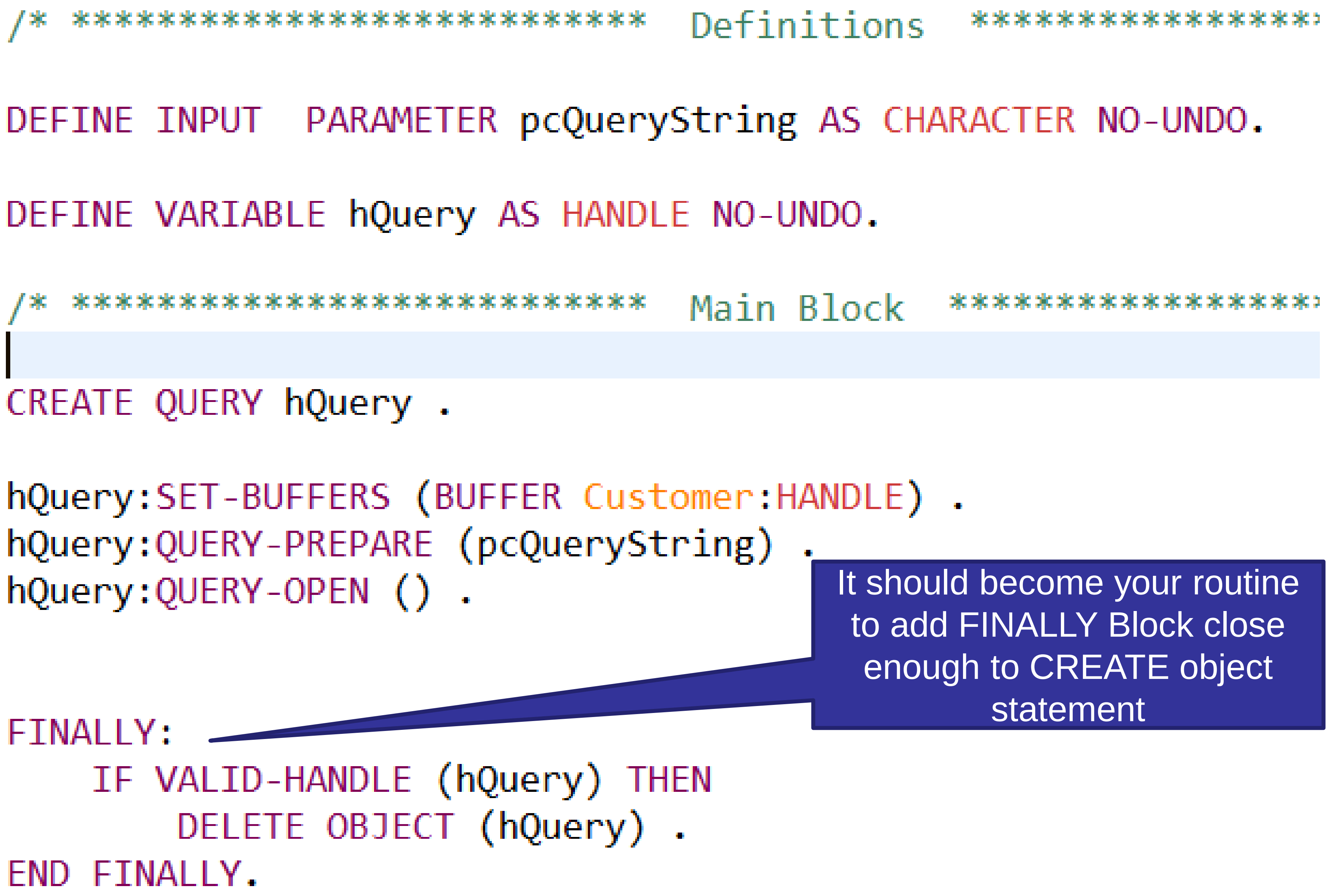
Structured Error Handling

- **Alternative** way of handling errors in the ABL
- Adds error handling constructs known form OO languages such as C# and Java to the ABL
- Based on OO-ABL elements
- Usable and useful with procedural code without exception (such as classic error handling can be used in OO)
- Available since 10.1C, enhanced in 10.2A (garbage collection), 11.3 (block-level default), 11.4 (serializable error classes)

FINALLY

- Clean up code block, can be attached to the end of every undoable block
- Executed when block succeeds, errors, executed per iteration of block (loops)
- Can be nested
- Available since 10.1C

- Must use! For everyone using dynamic queries etc… every code that requires clean up at runtime



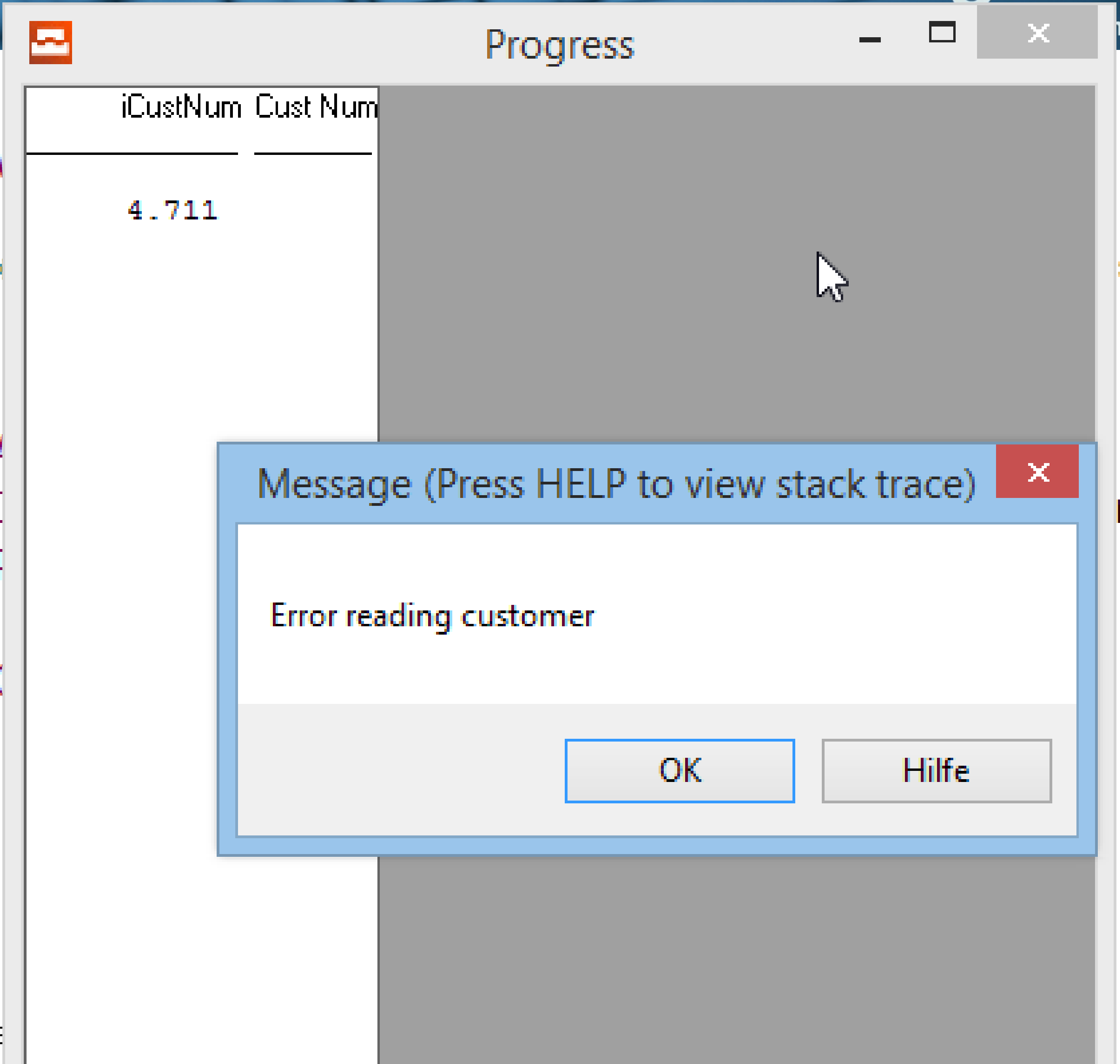
```
/* ***************************  Definitions  ******************

DEFINE INPUT  PARAMETER pcQueryString AS CHARACTER NO-UNDO.

DEFINE VARIABLE hQuery AS HANDLE NO-UNDO.

/* ***************************  Main Block  *******************

CREATE QUERY hQuery .

hQuery:SET-BUFFERS (BUFFER Customer:HANDLE) .
hQuery:QUERY-PREPARE (pcQueryString) .
hQuery:QUERY-OPEN () .


FINALLY:
    IF VALID-HANDLE (hQuery) THEN
        DELETE OBJECT (hQuery) .
END FINALLY.
```

It should become your routine to add FINALLY Block close enough to CREATE object statement

CATCH

- Block to handle runtime errors in the surrounding block
- When error occurs:
  - Block execution (iteration) stops
  - Transaction or sub-transaction undone
  - CATCH block execution
- CATCH Block allows to inspect the error
- Any undoable block is a TRY block (no explicit TRY block needed in the ABL)

```
DEFINE V

/* *****                                                    ****

REPEAT:
    UPDA
    FIND                                                    m .
    DISP

    CATC



    END
END.
```



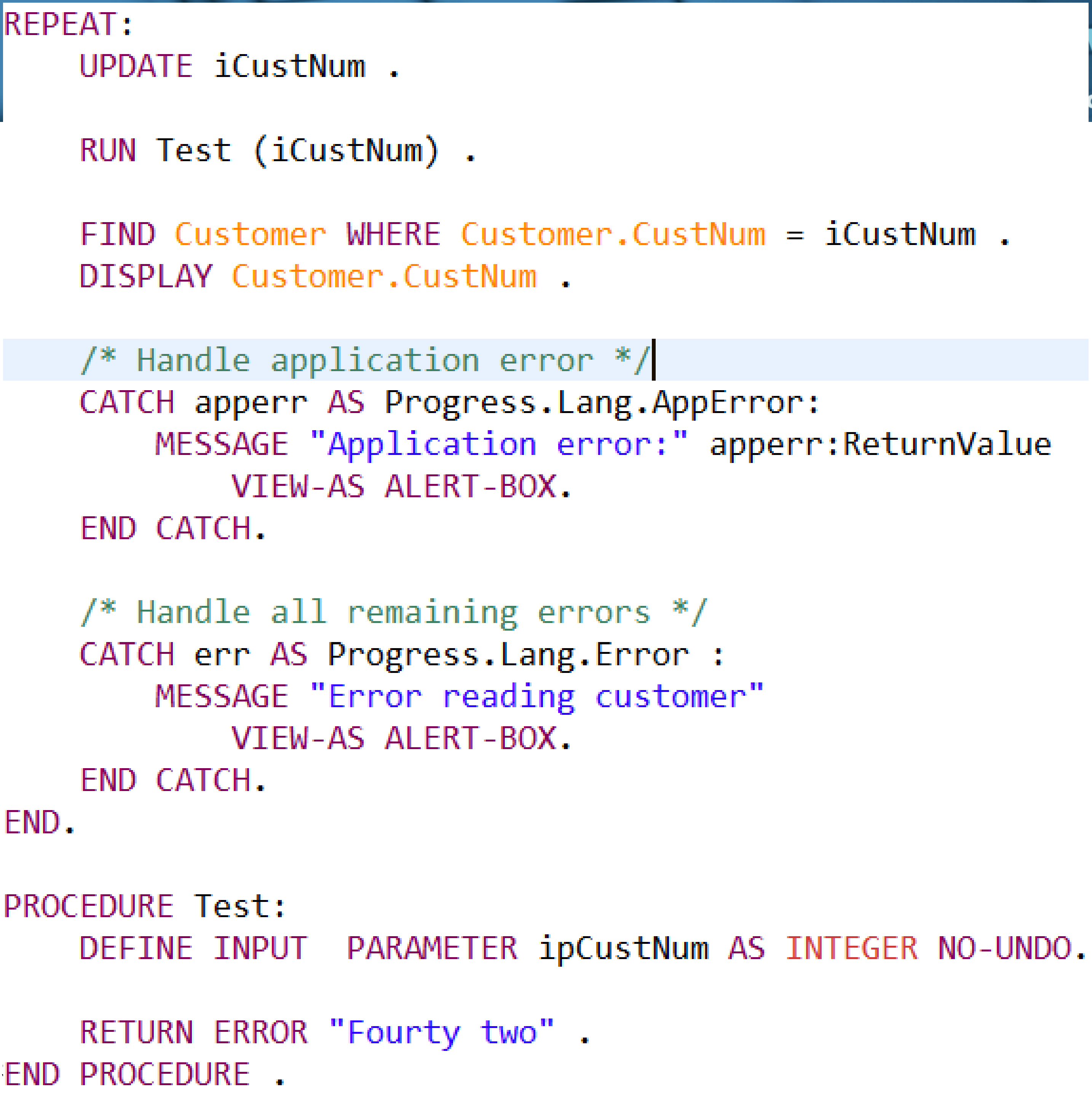
```
       iCustNum  Cust Num
  ----------   ---------
       4.711
```

Message (Press HELP to view stack trace)

Error reading customer

CATCH

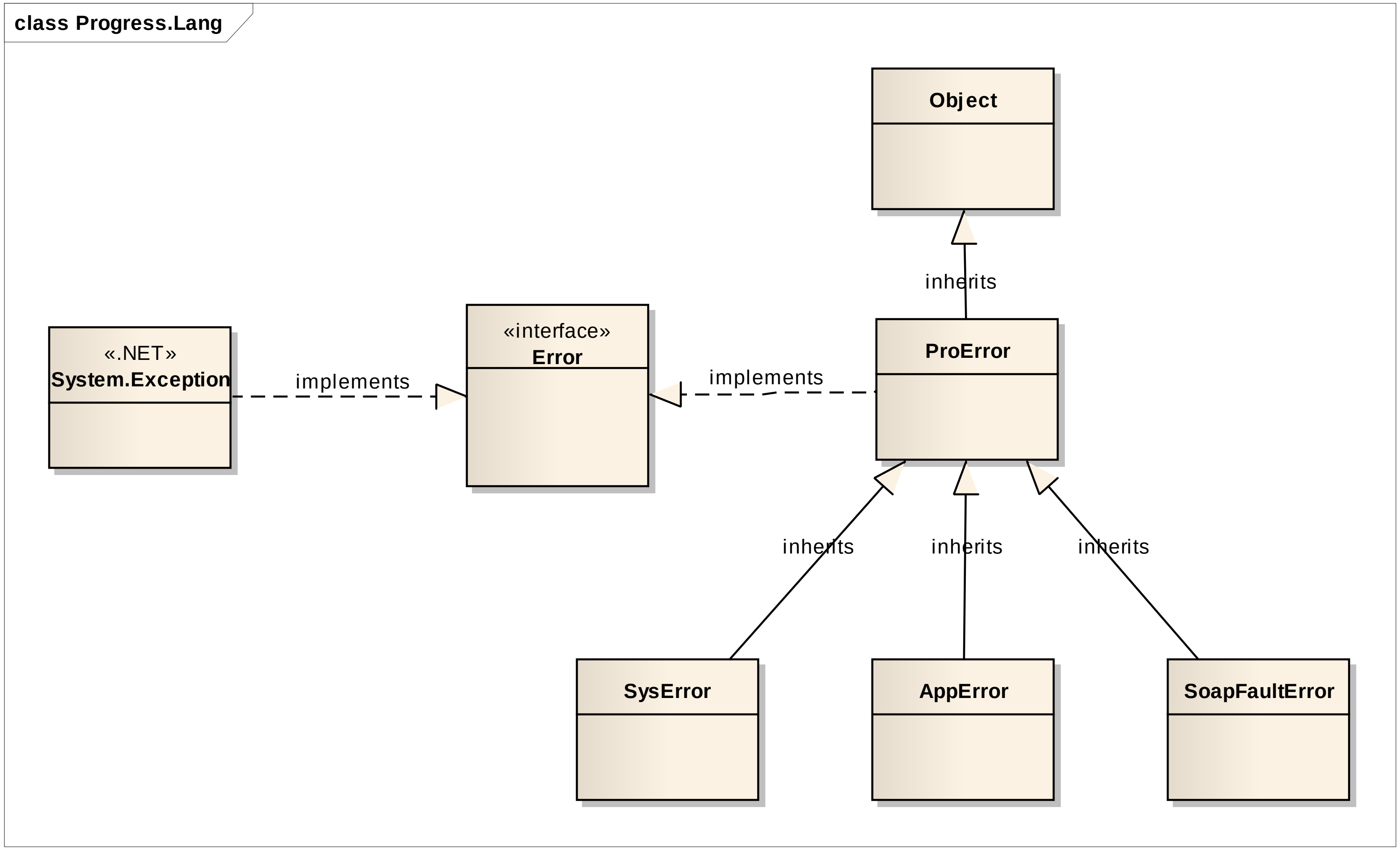
- CATCH block filters errors by type
- Multiple CATCH blocks can handle specific errors
- Should order CATCH blocks by specific to generic types
- First CATCH block matching the error that occurs handles it, following CATCH blocks will not handle the error

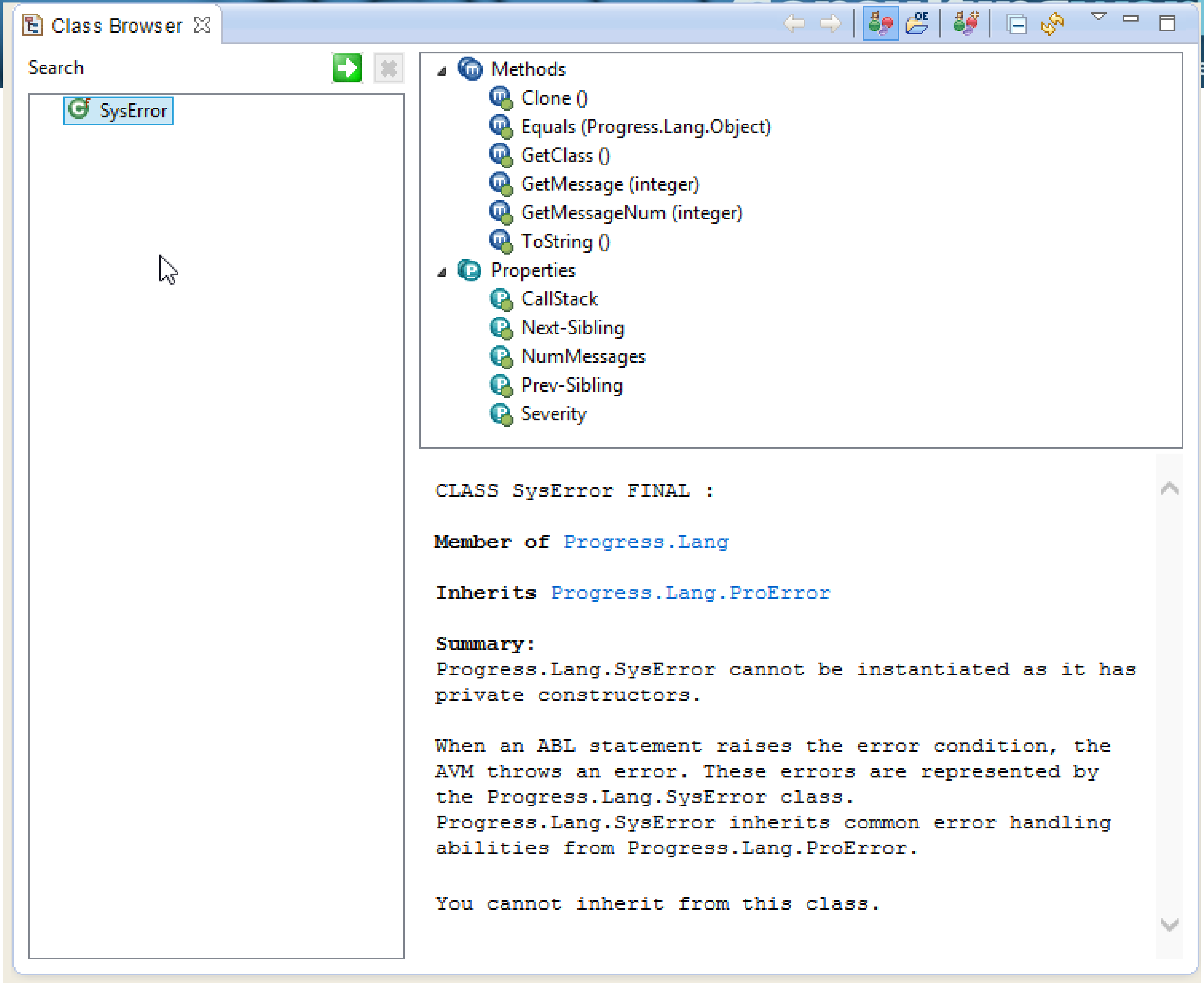


```
REPEAT:
    UPDATE iCustNum .

    RUN Test (iCustNum) .

    FIND Customer WHERE Customer.CustNum = iCustNum .
    DISPLAY Customer.CustNum .

    /* Handle application error */
    CATCH apperr AS Progress.Lang.AppError:
        MESSAGE "Application error:" apperr:ReturnValue
            VIEW-AS ALERT-BOX.
    END CATCH.

    /* Handle all remaining errors */
    CATCH err AS Progress.Lang.Error :
        MESSAGE "Error reading customer"
            VIEW-AS ALERT-BOX.
    END CATCH.
END.

PROCEDURE Test:
    DEFINE INPUT  PARAMETER ipCustNum AS INTEGER NO-UNDO.

    RETURN ERROR "Fourty two" .
END PROCEDURE .
```

Consultingwerk
software architecture and development

Error classes



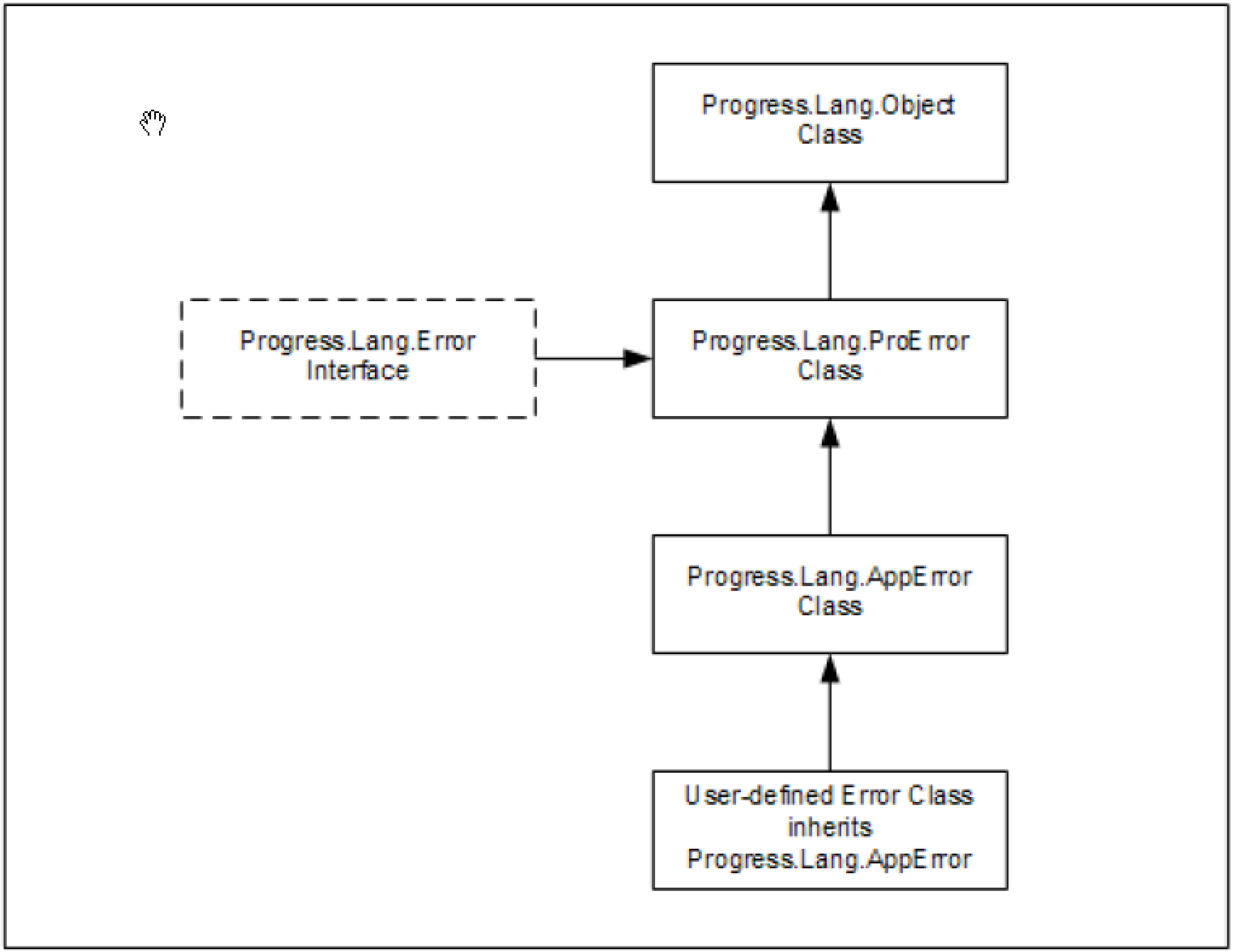


Methods
- Clone ()
- Equals (Progress.Lang.Object)
- GetClass ()
- GetMessage (integer)
- GetMessageNum (integer)
- ToString ()

Properties
- CallStack
- Next-Sibling
- NumMessages
- Prev-Sibling
- Severity

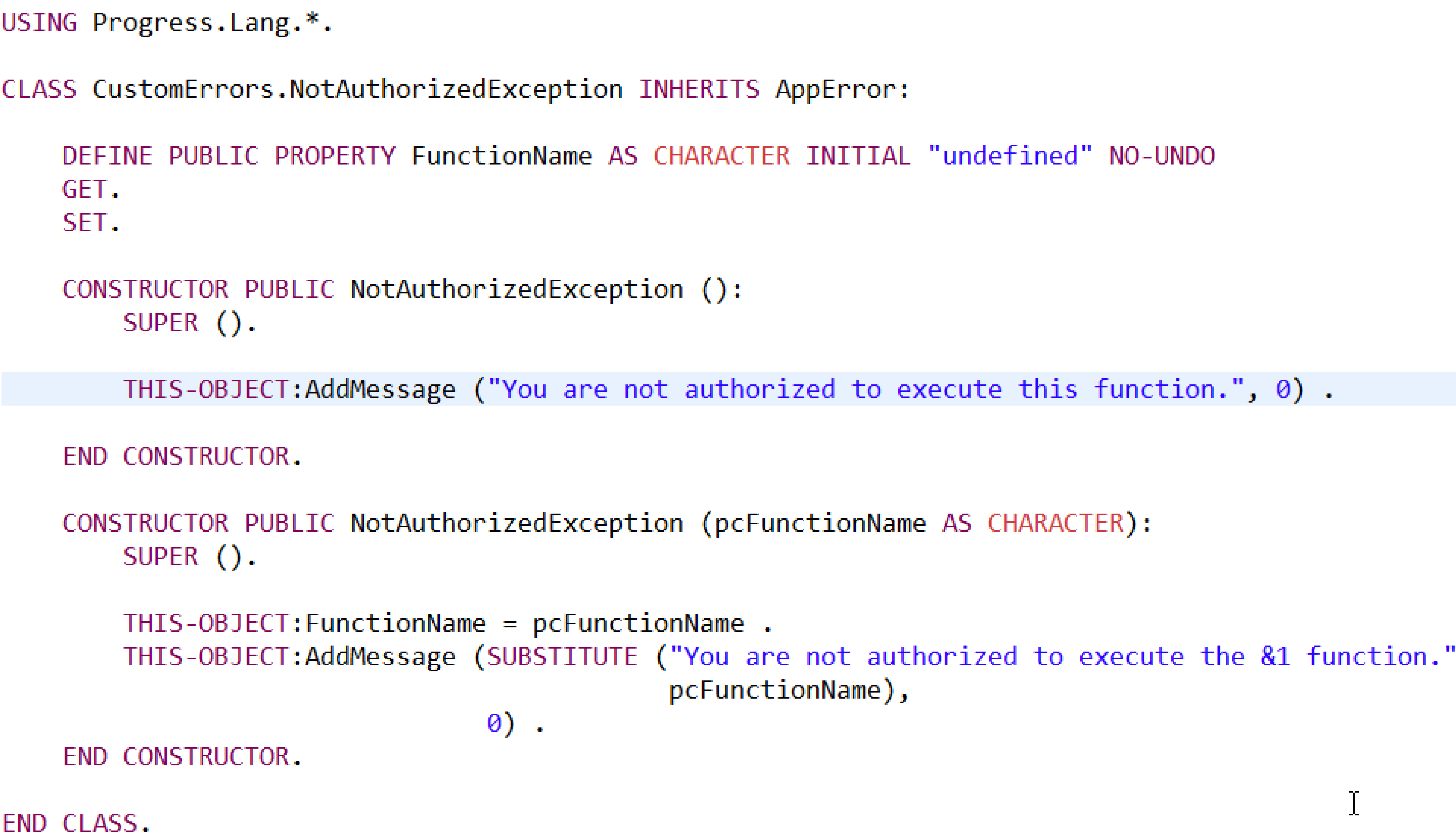
```
CLASS SysError FINAL :

Member of Progress.Lang

Inherits Progress.Lang.ProError

Summary:
Progress.Lang.SysError cannot be instantiated as it has
private constructors.

When an ABL statement raises the error condition, the
AVM throws an error. These errors are represented by
the Progress.Lang.SysError class.
Progress.Lang.SysError inherits common error handling
abilities from Progress.Lang.ProError.

You cannot inherit from this class.
```

Figure 1: Hierarchy of ABL error classes


Progress.Lang.Object Class
Progress.Lang.Error Interface
Progress.Lang.ProError Class
Progress.Lang.AppError Class
User-defined Error Class inherits Progress.Lang.AppError

Consultingwerk
software architecture and development

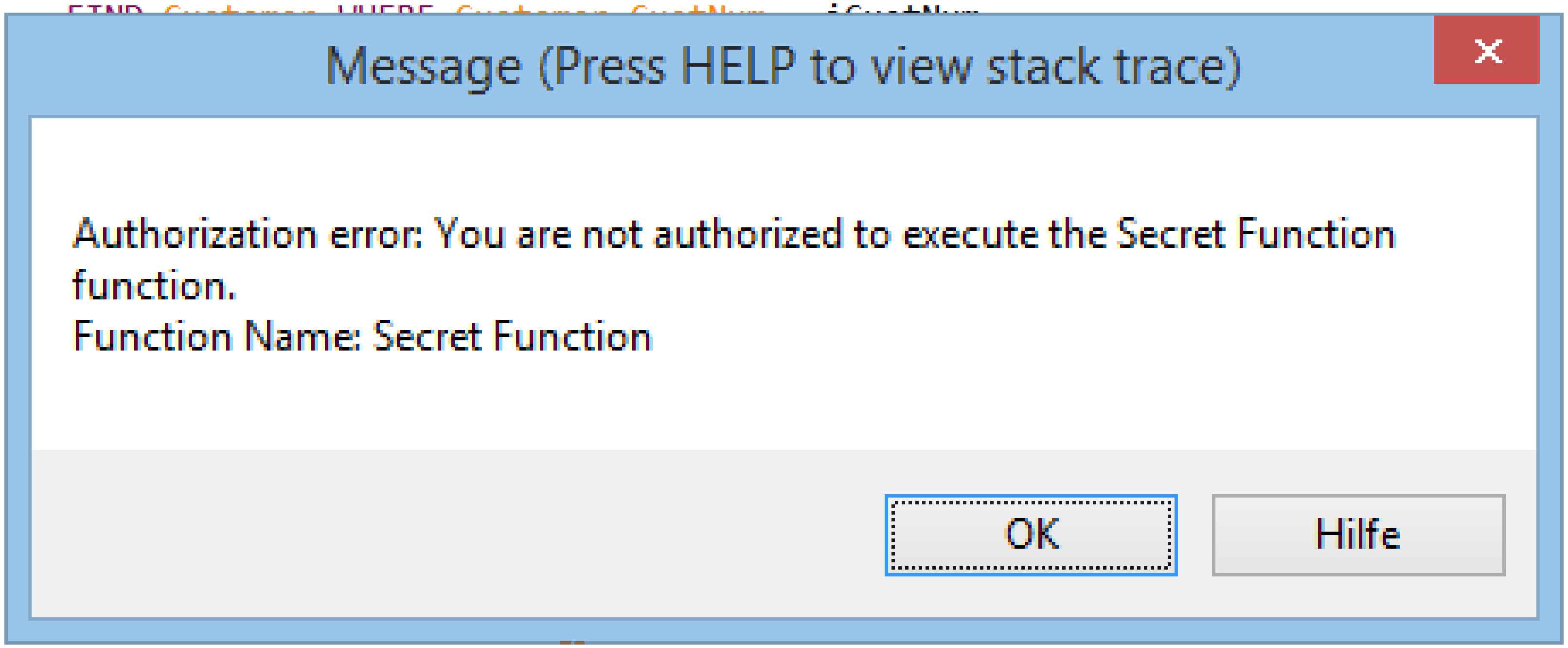
Custom Error Class



```
USING Progress.Lang.*.

CLASS CustomErrors.NotAuthorizedException INHERITS AppError:

    DEFINE PUBLIC PROPERTY FunctionName AS CHARACTER INITIAL "undefined" NO-UNDO
    GET.
    SET.

    CONSTRUCTOR PUBLIC NotAuthorizedException ():
        SUPER ().

        THIS-OBJECT:AddMessage ("You are not authorized to execute this function.", 0) .

    END CONSTRUCTOR.

    CONSTRUCTOR PUBLIC NotAuthorizedException (pcFunctionName AS CHARACTER):
        SUPER ().

        THIS-OBJECT:FunctionName = pcFunctionName .
        THIS-OBJECT:AddMessage (SUBSTITUTE ("You are not authorized to execute the &1 function."
                                            pcFunctionName),
                                0) .
    END CONSTRUCTOR.

END CLASS.
```

CATCH Custom Error

```
    RUN Test (iCustNum) .

    FIND Customer WHERE Customer.CustNum = iCustNum
```



Message (Press HELP to view stack trace)

Authorization error: You are not authorized to execute the Secret Function function.
Function Name: Secret Function

```
END.

PROCEDURE Test:
    DEFINE INPUT  PARAMETER piCustNum AS INTEGER NO-UNDO.

    IF USERID ("sports2000") <> "mikefe" THEN
        UNDO, THROW NEW NotAuthorizedException ("Secret Function") .
```

Demo

- Locate Consultingwerk.Exceptions.Exception in http://help.consultingwerkcloud.com/smartcomponent_library/release/

Raising errors … THROW

- DEFINE VARIABLE err AS <myerror> .
  err = NEW <myerror> (parameter) .
  err:CustomProperty = 42 .
  UNDO, THROW err .

- UNDO, THROW NEW <myerror> (parameter) .

- RETURN ERROR NEW <myerror> (parameter) .

AppError constructors

- NEW AppError ()
- NEW AppError (CHARACTER)
- NEW AppError (CHARACTER, INTEGER)

- **Attention:** AppError (CHARACTER) and AppError (CHARACTER, INTEGER) behave differently
- Signature suggest the INTEGER is for an optional parameter

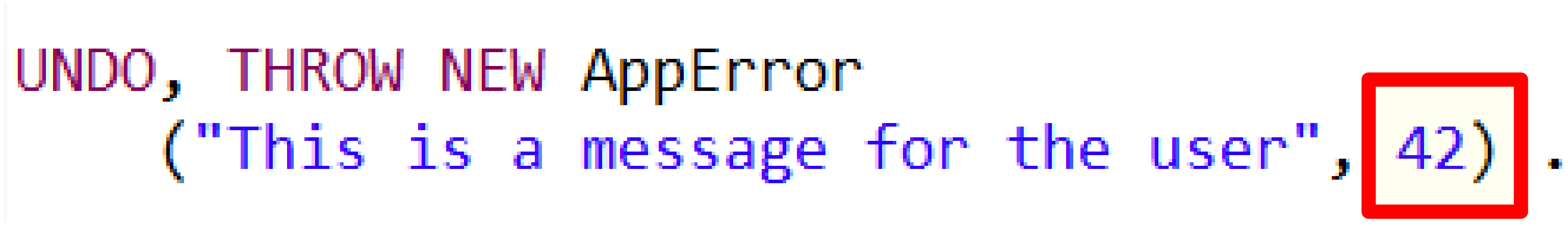
Consultingwerk
software architecture and development

AppError constructors

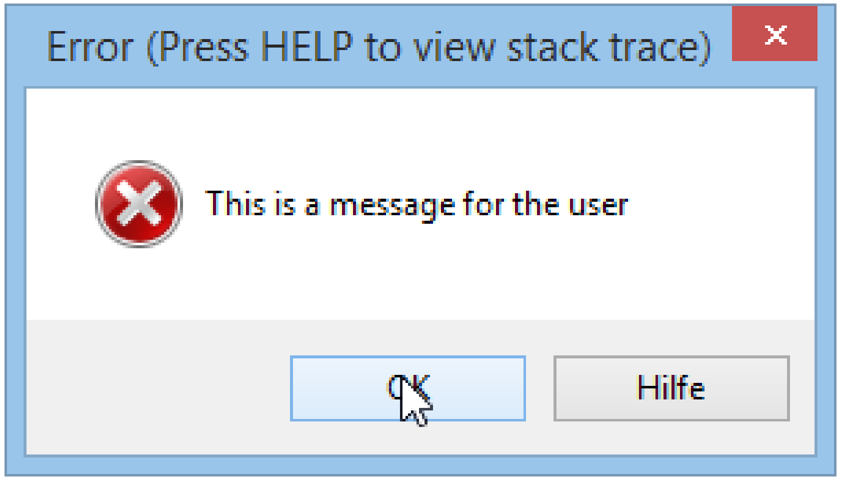
- UNDO, THROW NEW AppError
  („this text is ***NOT*** shown“) .
- UNDO, THROW NEW AppError
  („this text is shown to the user“, 42) .

- First statement initializes an AppError with a ReturnValue assigned (RETURN ERROR)
- Second statement initializes an AppError with a message and severity/number assigned!
- AVM only displays errors with message

Consultingwerk
software architecture and development



```
UNDO, THROW NEW AppError
    ("This is a message for the user", 42).
```



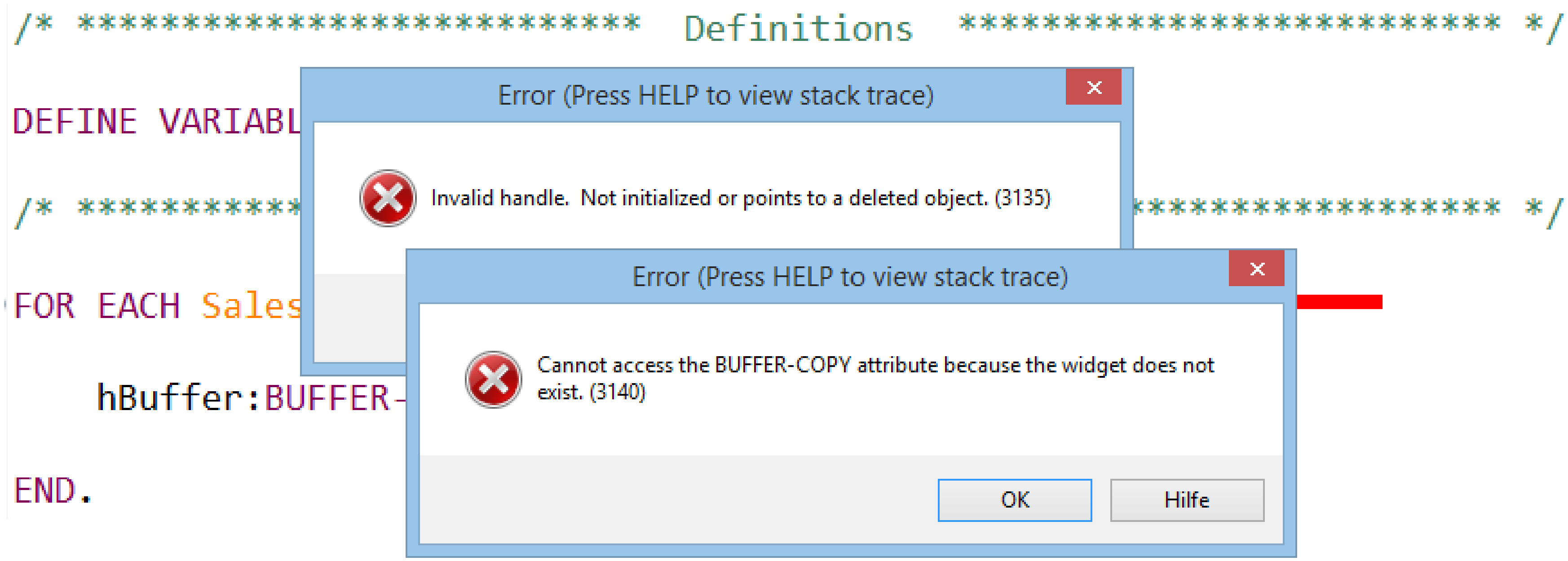
Error (Press HELP to view stack trace)

This is a message for the user

THROW Branch option on blocks …

- <block statement> ON ERROR UNDO, THROW:
- FOR EACH Customer ON ERROR UNDO, THROW

- When errors occur in the block, pass error for handling to next outer block
  – Next outer block may also throw
- Iterating block will **<u>NOT</u>** iterate any further
- First error terminates everything
- No more infinitive loops caused by errors!



```
/* ****************************  Definitions  **************************** */

DEFINE VARIABL

/* **********                                       ****************** */

FOR EACH Sales

    hBuffer:BUFFER-

END.
```

Error (Press HELP to view stack trace)

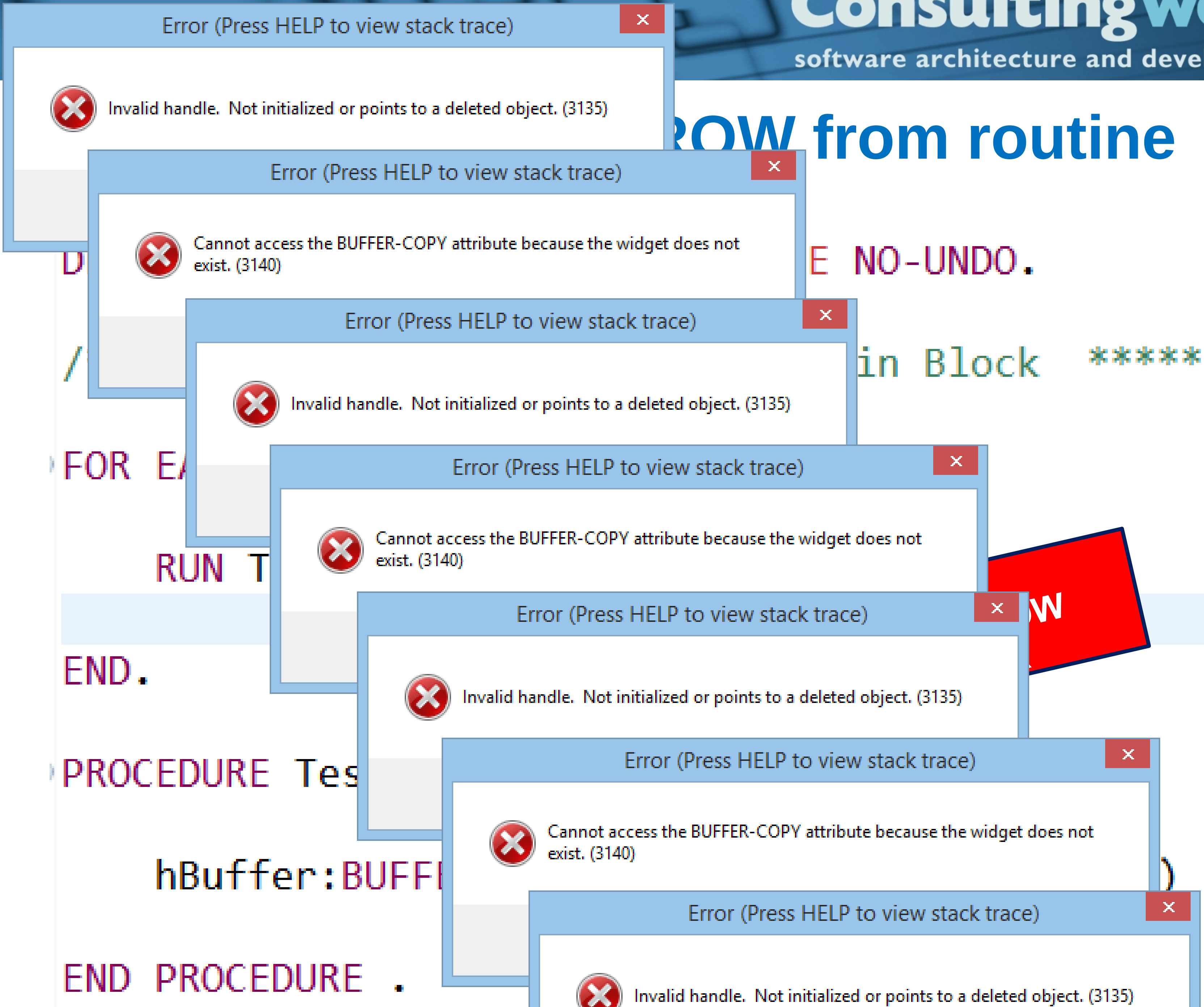
Invalid handle. Not initialized or points to a deleted object. (3135)

Error (Press HELP to view stack trace)

Cannot access the BUFFER-COPY attribute because the widget does not exist. (3140)

OW from routine

```
D                          E NO-UNDO.

/                          in Block  **********

FOR EA

    RUN T

END.

PROCEDURE Tes

    hBuffer:BUFF                               ) .

END PROCEDURE .
```



Error (Press HELP to view stack trace)

Invalid handle. Not initialized or points to a deleted object. (3135)

Error (Press HELP to view stack trace)

Cannot access the BUFFER-COPY attribute because the widget does not exist. (3140)

Error (Press HELP to view stack trace)

Invalid handle. Not initialized or points to a deleted object. (3135)

Error (Press HELP to view stack trace)

Cannot access the BUFFER-COPY attribute because the widget does not exist. (3140)

Error (Press HELP to view stack trace)

Invalid handle. Not initialized or points to a deleted object. (3135)

Error (Press HELP to view stack trace)

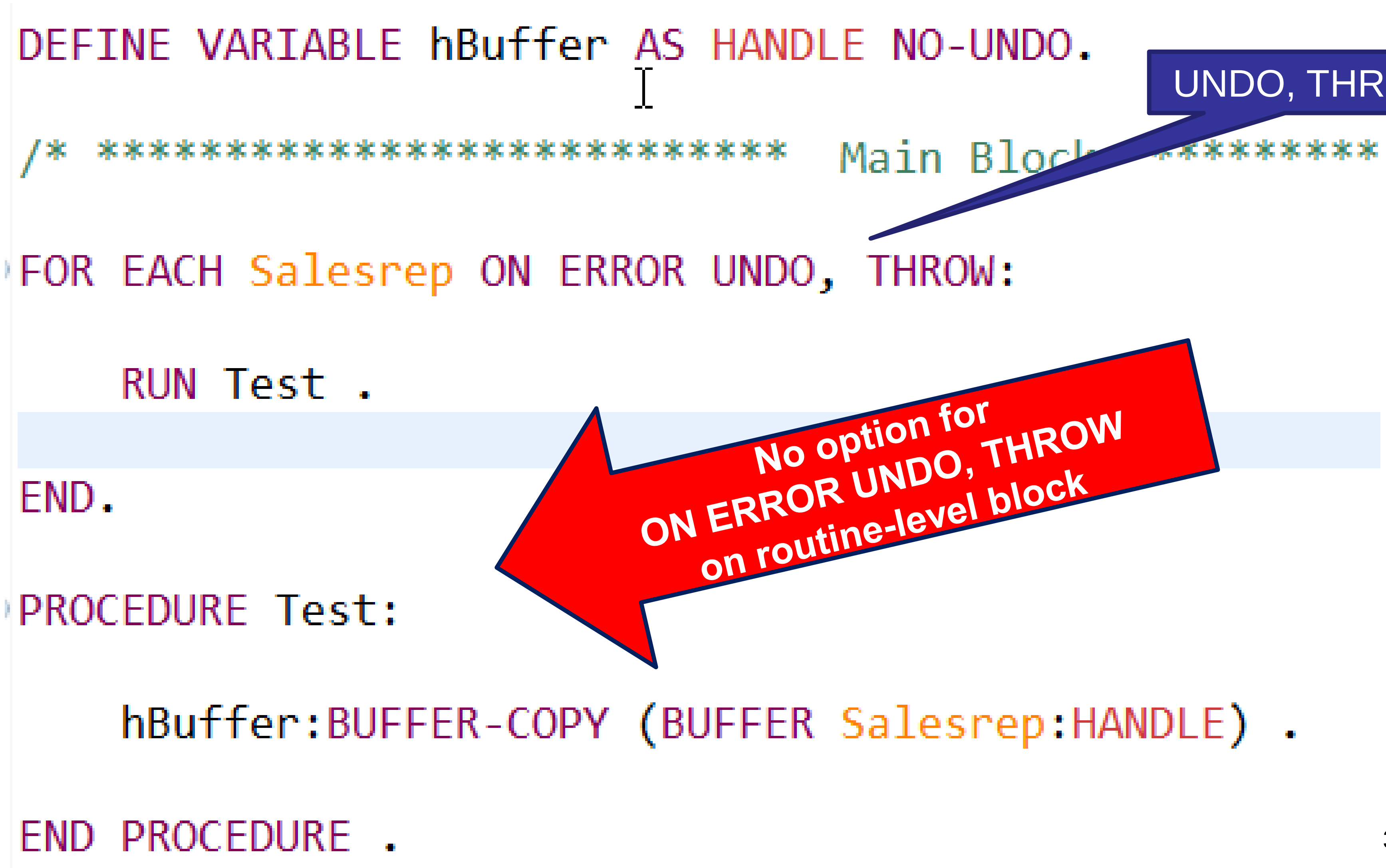
Cannot access the BUFFER-COPY attribute because the widget does not exist. (3140)

Error (Press HELP to view stack trace)

Invalid handle. Not initialized or points to a deleted object. (3135)

W

Problem: UNDO, THROW from routine



```
DEFINE VARIABLE hBuffer AS HANDLE NO-UNDO.

/* ****************************  Main Bloc  **********

FOR EACH Salesrep ON ERROR UNDO, THROW:

    RUN Test .

END.

PROCEDURE Test:

    hBuffer:BUFFER-COPY (BUFFER Salesrep:HANDLE) .

END PROCEDURE .
```

UNDO, THROW

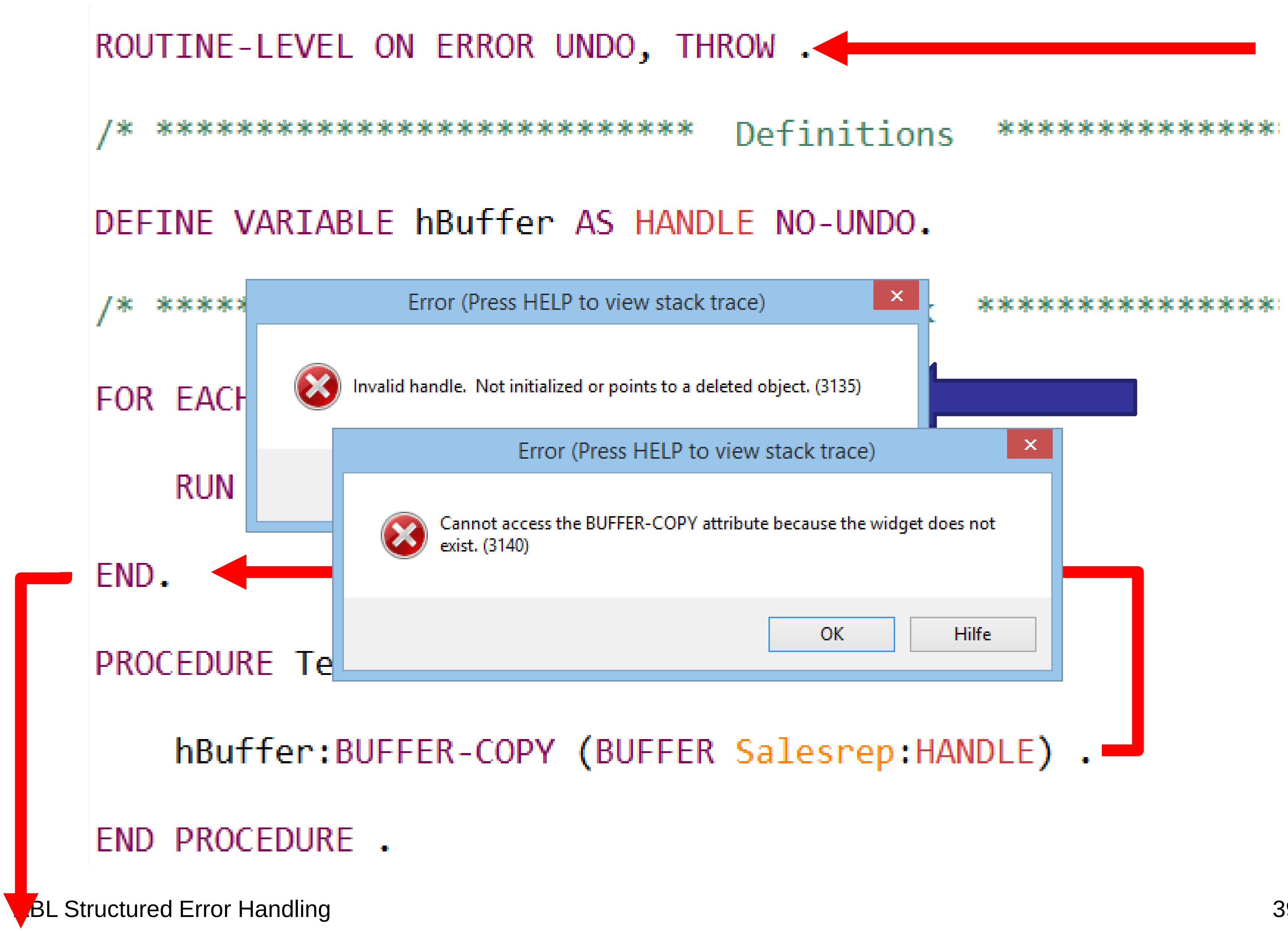
No option for
ON ERROR UNDO, THROW
on routine-level block

Consultingwerk
software architecture and development

ROUTINE-LEVEL ON ERROR UNDO, THROW

- Declarative statement, available since 10.1C
- Must be placed before any executable statements, including DEFINE‘s
- May be after USING

- Changes ERROR Branching option for ALL routine blocks in compile unit to ON ERROR UNDO, THROW
- No effect on loops, DO TRANSACTION, DO ON ERROR



```
ROUTINE-LEVEL ON ERROR UNDO, THROW .

/* ***************************   Definitions   ***************

DEFINE VARIABLE hBuffer AS HANDLE NO-UNDO.

/* *****                                  *  ***************

FOR EACH

    RUN

END.

PROCEDURE Te

    hBuffer:BUFFER-COPY (BUFFER Salesrep:HANDLE) .

END PROCEDURE .
```

Error (Press HELP to view stack trace)

Invalid handle. Not initialized or points to a deleted object. (3135)

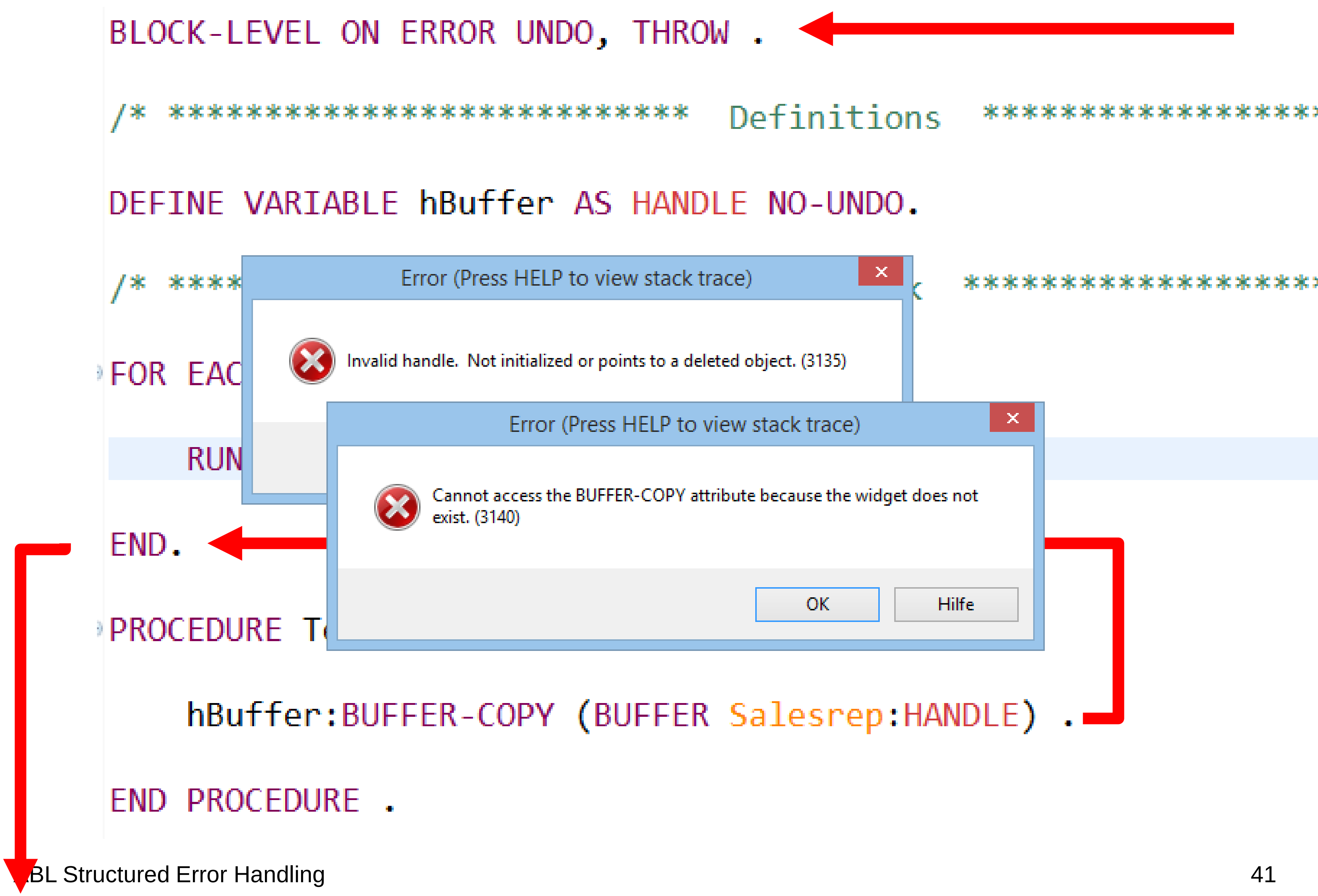
Error (Press HELP to view stack trace)

Cannot access the BUFFER-COPY attribute because the widget does not exist. (3140)

BLOCK-LEVEL ON ERROR UNDO, THROW

- Declarative statement, available since 11.3
- Must be placed before any executable statements, including DEFINE‘s
- May be after USING

- Changes default ERROR Branching option for ALL blocks in compile unit to ON ERROR UNDO, THROW
- May be redefined on loops, DO TRANSACTION, DO ON ERROR



```
BLOCK-LEVEL ON ERROR UNDO, THROW .

/* ****************************  Definitions  ******************

DEFINE VARIABLE hBuffer AS HANDLE NO-UNDO.

/* ****                                        *  ******************

FOR EAC

    RUN

END.

PROCEDURE T

    hBuffer:BUFFER-COPY (BUFFER Salesrep:HANDLE) .

END PROCEDURE .
```

Error (Press HELP to view stack trace)

Invalid handle. Not initialized or points to a deleted object. (3135)

Error (Press HELP to view stack trace)

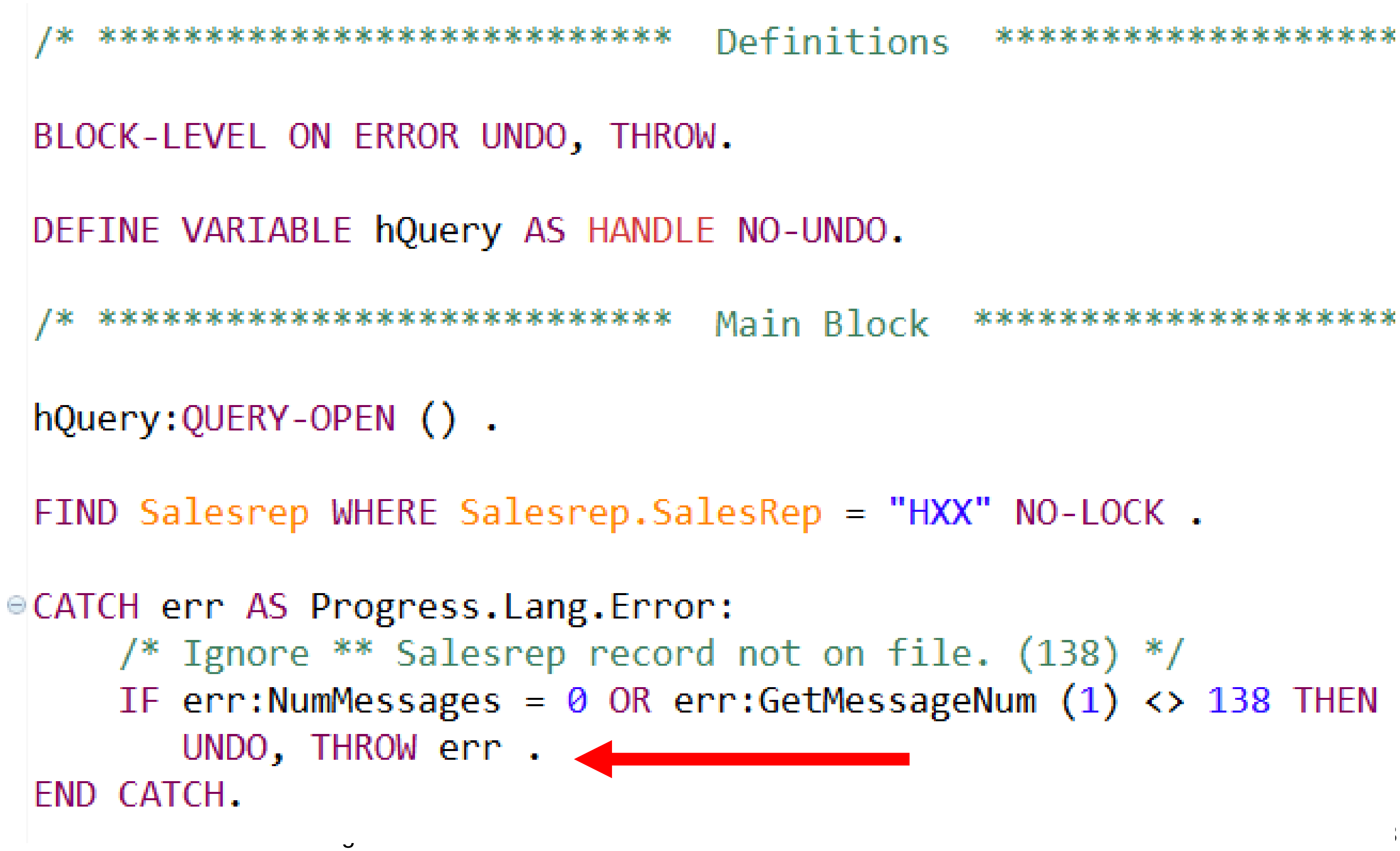
Cannot access the BUFFER-COPY attribute because the widget does not exist. (3140)

BLOCK-LEVEL ON ERROR UNDO, THROW

- Don’t get used to BLOCK-LEVEL ON ERROR when still deploying to 10.1C – 11.2
- Don’t switch between ROUTINE-LEVEL and BLOCK-LEVEL using PROVERSION preprocessor

- Error handling should be tested during development
- And behave 100% the same way at runtime
- Surprises in error handling are expensive …

Re-THROW



```
/* ***************************  Definitions  ************************

BLOCK-LEVEL ON ERROR UNDO, THROW.

DEFINE VARIABLE hQuery AS HANDLE NO-UNDO.

/* ***************************  Main Block  *************************

hQuery:QUERY-OPEN () .

FIND Salesrep WHERE Salesrep.SalesRep = "HXX" NO-LOCK .

CATCH err AS Progress.Lang.Error:
    /* Ignore ** Salesrep record not on file. (138) */
    IF err:NumMessages = 0 OR err:GetMessageNum (1) <> 138 THEN
       UNDO, THROW err .
END CATCH.
```

Re-THROW of Progress.Lang.SysError

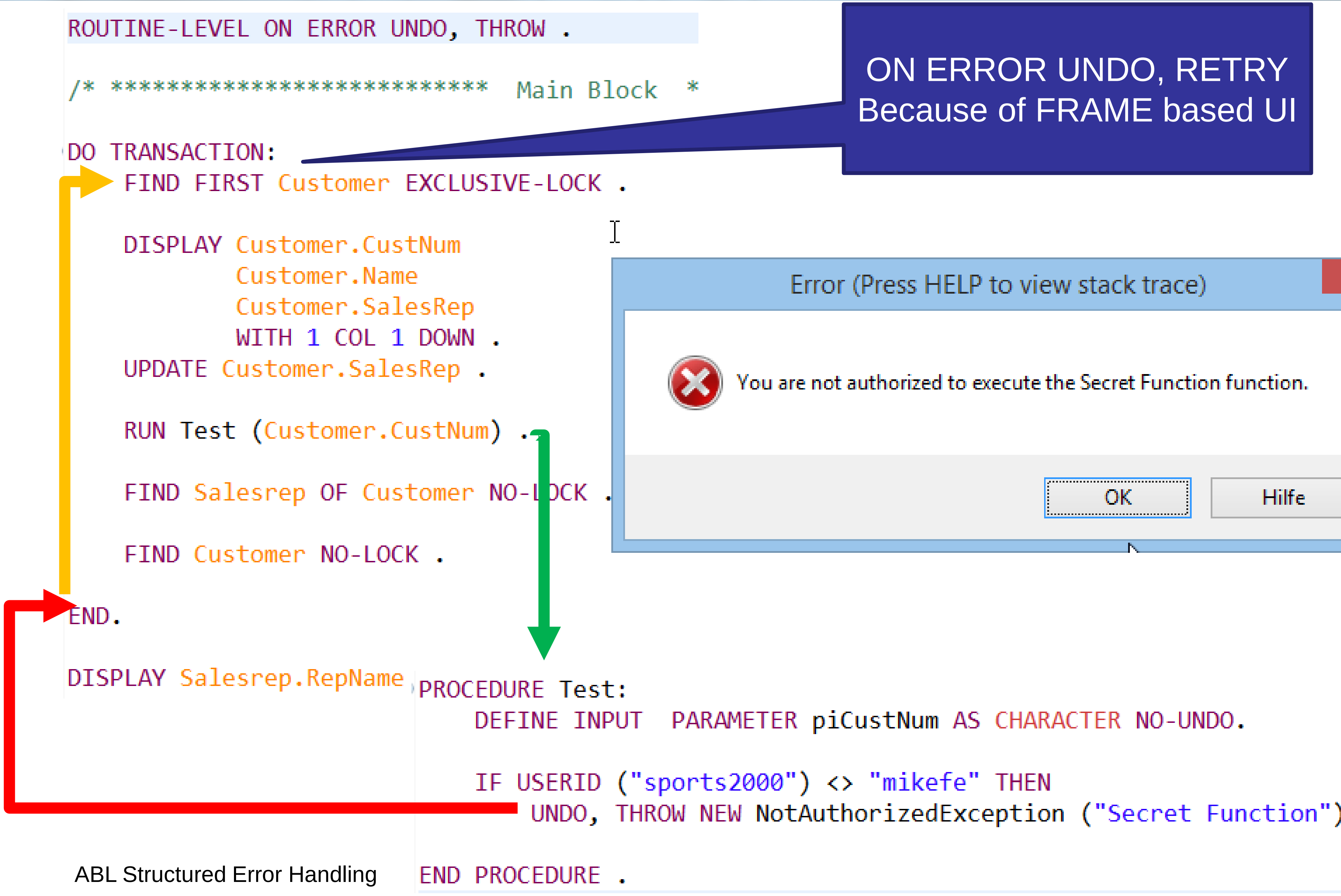
- Runtime error often the root cause for application raised error
- Application raised error may be adding context
- Unexpected SysError should be re-thrown
- May require to “track” possible runtime error numbers – no (complete) list of errors expected per ABL statement or Widget method availalbe

Agenda

- Traditional/Classic Error Handling recap
- Structured Error Handling
- Combining Structured and Classic Err. Handling
- STOP Conditions
- Error Handling best practices

Combining Structured and Classic Err. Hndl

- THROW and classic error handling fully compatible
- AppError or any of the runtime error handled based on branch action of receiving block:
  - RETRY
  - NEXT
  - LEAVE
  - RETURN



```
ROUTINE-LEVEL ON ERROR UNDO, THROW .

/* ****************************  Main Block  *

DO TRANSACTION:
    FIND FIRST Customer EXCLUSIVE-LOCK .

    DISPLAY Customer.CustNum
            Customer.Name
            Customer.SalesRep
            WITH 1 COL 1 DOWN .
    UPDATE Customer.SalesRep .

    RUN Test (Customer.CustNum) .

    FIND Salesrep OF Customer NO-LOCK .

    FIND Customer NO-LOCK .

END.

DISPLAY Salesrep.RepName
```

ON ERROR UNDO, RETRY
Because of FRAME based UI

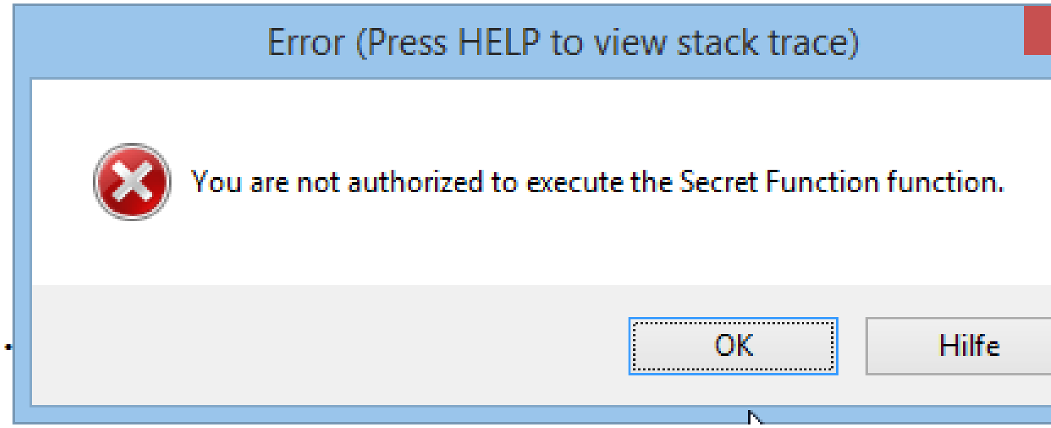
```
PROCEDURE Test:
    DEFINE INPUT  PARAMETER piCustNum AS CHARACTER NO-UNDO.

    IF USERID ("sports2000") <> "mikefe" THEN
        UNDO, THROW NEW NotAuthorizedException ("Secret Function")

END PROCEDURE .
```

Error (Press HELP to view stack trace)

You are not authorized to execute the Secret Function function.

```
ROUTINE-LEVEL ON ERROR UNDO, THROW .

/* ***************************  Main Block  *

DO TRANSACTION:
    FIND FIRST Customer EXCLUSIVE-LOCK .

    DISPLAY Customer.CustNum
            Customer.Name
            Customer.SalesRep
            WITH 1 COL 1 DOWN .
    UPDATE Customer.SalesRep .

    RUN Test (Customer.CustNum) .

    FIND Salesrep OF Customer NO-LOCK .

    FIND Customer NO-LOCK .

END.

DISPLAY Salesrep.RepName
```

ON ERROR UNDO, RETRY
Because of FRAME based UI


Error (Press HELP to view stack trace)

You are not authorized to execute the Secret Function function.

```
PROCEDURE Test:
    DEFINE INPUT  PARAMETER piCustNum AS CHARACTER NO-UNDO.

    IF USERID ("sports2000") <> "mikefe" THEN
        UNDO, THROW NEW NotAuthorizedException ("Secret Function")

END PROCEDURE .
```

Combining Structured and Classic Err. Hndl

- All errors available through ERROR-STATUS system handle
- AppError with ReturnValue same impact as RETURN ERROR <return-value>

Recommendation

- Do not add ROUTINE-LEVEL or BLOCK-LEVEL Error Handling to existing code without testing!
- Users may be used to executing reports with a few „*Item record not on file*“ messages
- Adding ROUTINE-LEVEL or BLOCK-LEVEL error handling will severely impact the program flow.
- Users may not be able to receive the report they need to get their job done

Agenda

- Traditional/Classic Error Handling recap
- Structured Error Handling
- Combining Structured and Classic Err. Handling
- STOP Conditions
- Error Handling best practices

STOP Conditions

- Critical system error
- RUN non-existing .p file
- .r code CRC does not match schema
- trigger r. code not matching the CRC stored in DB schema
- DB connection missing/lost
- …
- STOP-AFTER (unfortunately)

STOP Condition handling

- UNDO blocks until it reaches an ON STOP Block
- AVM may decide to skip ON STOP and proceed with reverting …
- Eventually may restart client startup procedure
- FINALLY Blocks are NOT executed!

Convert STOP into AppError

- STOP can be handled by RETURN ERROR
- FINALLY will be executed

```
DO ON STOP UNDO, RETURN ERROR NEW StopConditionException
    (SUBSTITUTE ("A stop condition has been raised while waiting for the backend.~nBackend component: &
                 pcEntityName),
     0):

    RUN VALUE(THIS-OBJECT:ServiceInterfacePath + "/proSIfetchDataset.p":U) ON hAppServer
                                (INPUT pcEntityName,
                                 OUTPUT DATASET-HANDLE phDataset,
                                 INPUT-OUTPUT DATASET-HANDLE hContextDataset BY-REFERENCE) .
END.

FINALLY:
    {Consultingwerk/OERA/delete-dataset.i phDataset deleteerr}

    THIS-OBJECT:OnCollectContextFromServer (Consultingwerk.EventArgs:Empty) .

    IF VALID-HANDLE (hContextDataset) THEN
        DELETE OBJECT hContextDataset NO-ERROR .
END FINALLY.
```

Agenda

- Traditional/Classic Error Handling recap
- Structured Error Handling
- Combining Structured and Classic Err. Handling
- STOP Conditions
- Error Handling best practices

Consultingwerk
software architecture and development

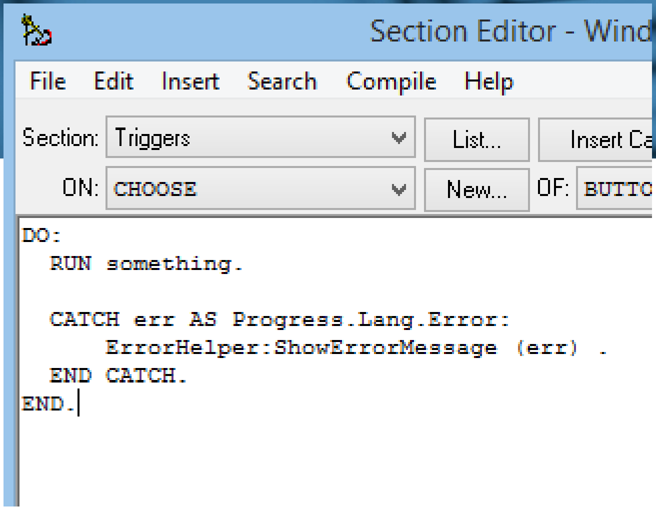
Errors should be exceptional

- Errors are called “Exception” in .NET and Java

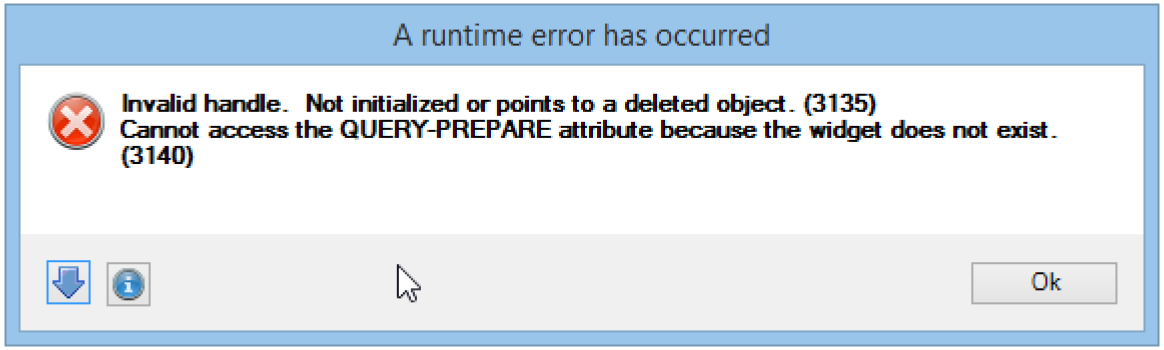
- Don‘t use errors as alternative RETURN-VALUE
- Errors should not happen
- Error handling is an expensive operation
- Transaction will be undone

CATCH in every Event handlers

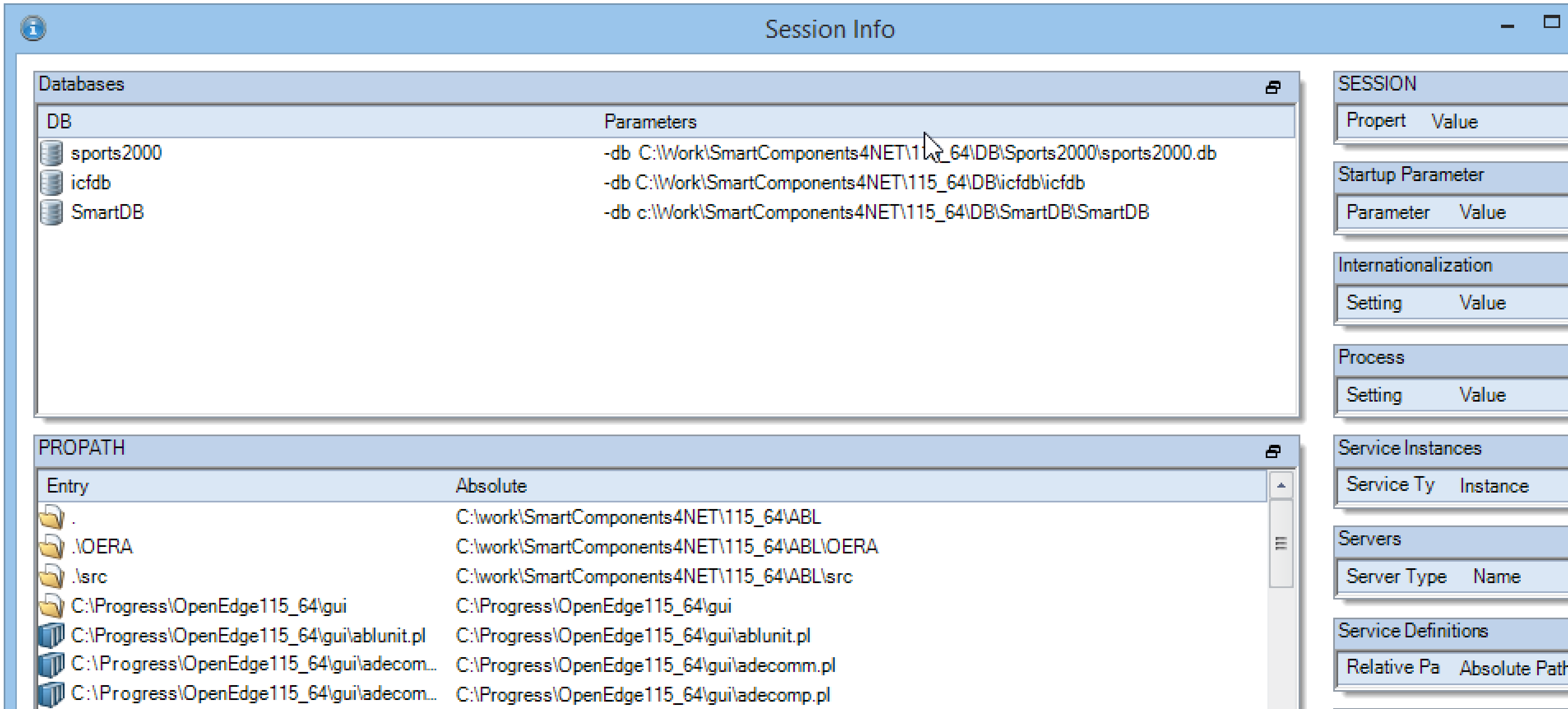
- When using structured error handling, every UI event handler should have a CATCH
  - Classic ABL GUI
  - TTY
  - GUI for .NET
- If you don't show messages by then the AVM will do
- Only way to create nice and feature-rich error dialog



```
DO:
  RUN something.

  CATCH err AS Progress.Lang.Error:
      ErrorHelper:ShowErrorMessage (err) .
  END CATCH.
END.
```



Invalid handle. Not initialized or points to a deleted object. (3135)
Cannot access the QUERY-PREPARE attribute because the widget does not exist. (3140)



DB	Parameters
sports2000	-db C:\Work\SmartComponents4NET\115_64\DB\Sports2000\sports2000.db
icfdb	-db C:\Work\SmartComponents4NET\115_64\DB\icfdb\icfdb
SmartDB	-db c:\Work\SmartComponents4NET\115_64\DB\SmartDB\SmartDB

Entry	Absolute
.	C:\work\SmartComponents4NET\115_64\ABL
.\OERA	C:\work\SmartComponents4NET\115_64\ABL\OERA
.\src	C:\work\SmartComponents4NET\115_64\ABL\src
C:\Progress\OpenEdge115_64\gui	C:\Progress\OpenEdge115_64\gui
C:\Progress\OpenEdge115_64\gui\ablunit.pl	C:\Progress\OpenEdge115_64\gui\ablunit.pl
C:\Progress\OpenEdge115_64\gui\adecom...	C:\Progress\OpenEdge115_64\gui\adecomm.pl
C:\Progress\OpenEdge115_64\gui\adecom...	C:\Progress\OpenEdge115_64\gui\adecomp.pl

CATCH in worker methods

- Non UI code, batch routines, code that is calculating values etc.
- Worker methods on the other end, should not show error messages, all errors should be thrown
- Worker methods should only CATCH to handle automatically
  - open query if not open, then re-run
  - delete dynamic query widget, when QUERY-PREPARE-FAILED

Avoid NO-ERROR

- This cannot be handled with a CATCH block
- hQuery:QUERY-PREPARE ("…") NO-ERROR
- Only way to handle this is ERROR-STATUS:ERROR or explicit checks
  - IF AVAILABLE Customer
- ERROR-STATUS:ERROR valid until next statement is executed with NO-ERROR

Consultingwerk
software architecture and development

InnerException Pattern

- Known from .NET
- Useful to „upgrade“ certain runtime errors to application errors (similar to RE-THROW)
- Keep original error information, such as Stack-Trace, ReturnValue, error messages, custom properties
- When error happens, it happens, … all I can do in some locations is to add context data, to simplify debugging
- See Consultingwerk.Exceptions.Exception and DataAccess class

```
CATCH err AS Progress.Lang.Error :
    /* Mike Fechner, Consultingwerk Ltd. 27.05.2014
       SCL-258 */
    ASSIGN THIS-OBJECT:QueryPurpose = QueryPurposeEnum:CatchBlock .

    DO ON ERROR UNDO, THROW:
        cDefault = THIS-OBJECT:SourceDefaultQuery (hBuffer:NAME) .

        CATCH innererr AS Progress.Lang.Error:
            /* ignore errors from here */
        END CATCH.
    END.

    ASSIGN THIS-OBJECT:QueryPurpose = QueryPurposeEnum:Invalid .

    ASSIGN oException = NEW FetchDataException (err,
                                                err:GetMessage (1),
                                                err:GetMessageNum (1),
                                                cQuery,
                                                cDefault) .

    DO iMessage = 2 TO err:NumMessages:
        oException:AddMessage (err:GetMessage(iMessage), 0) .
    END.

    oException:AddMessage (SUBSTITUTE ("Client-Query: &1"{&TRAN}, cQuery), 0) .
    oException:AddMessage (SUBSTITUTE ("Default-Query: &1"{&TRAN}, cDefault), 0) .

    UNDO, THROW oException .
END CATCH.
```

Code Review

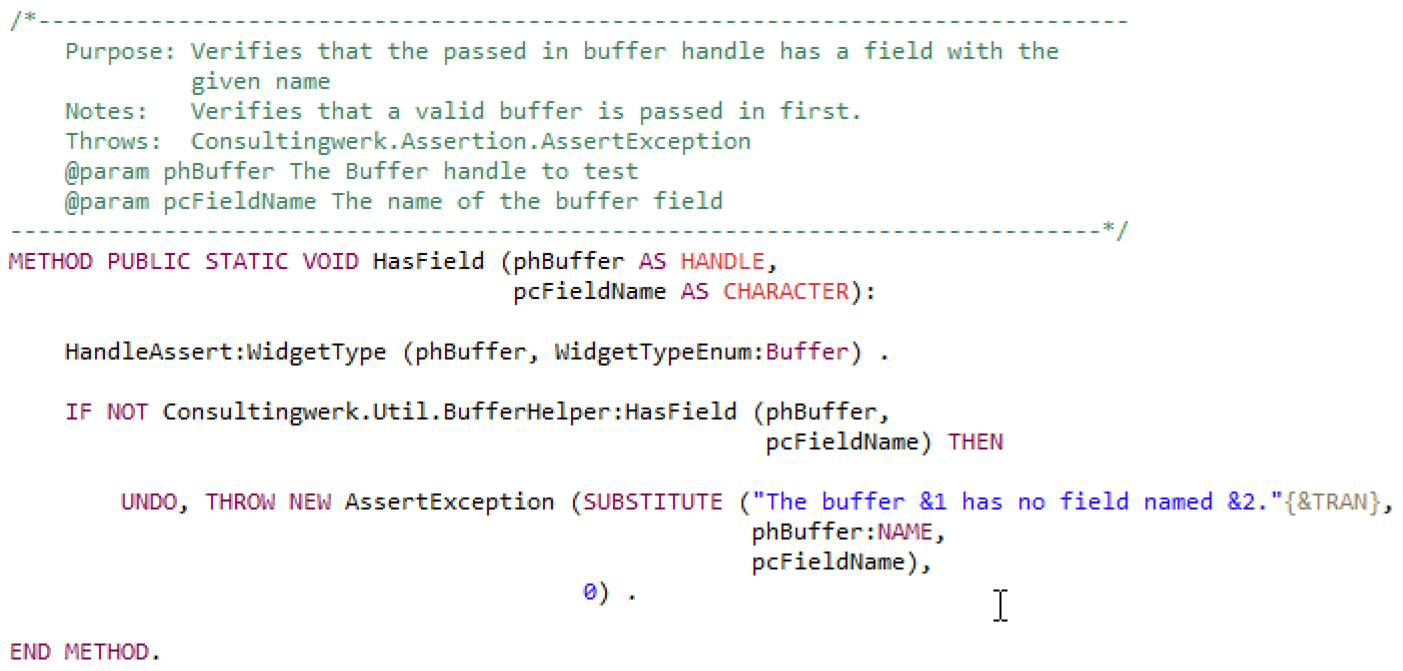
- Consultingwerk.Exceptions.Exception
  - InnerException Implementation
  - SessionInfo from AppServer to Client

Assertions

- Concept for validating method parameters
- Verify a value
- Throw an error when expected condition is not met
- Static methods
- Simple way to provide consistent errors for common issues

Example from BufferAssert

- Consultingwerk.Assertion.BufferAssert:HasField (hBuffer, “CustNum”) .



```
/*------------------------------------------------------------------------------
    Purpose: Verifies that the passed in buffer handle has a field with the
             given name
    Notes:   Verifies that a valid buffer is passed in first.
    Throws:  Consultingwerk.Assertion.AssertException
    @param phBuffer The Buffer handle to test
    @param pcFieldName The name of the buffer field
------------------------------------------------------------------------------*/
METHOD PUBLIC STATIC VOID HasField (phBuffer AS HANDLE,
                                    pcFieldName AS CHARACTER):

    HandleAssert:WidgetType (phBuffer, WidgetTypeEnum:Buffer) .

    IF NOT Consultingwerk.Util.BufferHelper:HasField (phBuffer,
                                                      pcFieldName) THEN

        UNDO, THROW NEW AssertException (SUBSTITUTE ("The buffer &1 has no field named &2."{&TRAN},
                                                     phBuffer:NAME,
                                                     pcFieldName),
                                         0) .

END METHOD.
```

Questions?

Don‘t miss my other presentations

- Monday 11.00: **Telerik .NET for Infragistics Users**
- Monday 16.45: DIY: **Lists, Enumerators, Enumerations, Serialization**
- Tuesday 11.00: **Modernization – the SmartComponent Library**
- Tuesday 14.15: **Structured Error Handling**
- Wednesday 11.00: **Telerik Kendo UI with WebSpeed**

PUG
CHALLENGE
EXCHANGE
AMERICAS