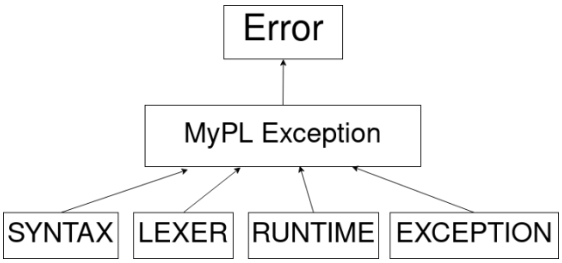
Zac Foteff . OPL Final Project: Exception Handling Specification

I have chosen to implement exception handling through the use of try-catch blocks as my MyPL extension project. Try-Catch blocks allow programmers to control the flow of execution of their programs and allows them to manage possible exception cases that can occur in MyPL. UML diagram number 1 shows the basic structure of exceptions in MyPL. There are 3 existing error cases that need to be considered, and a blanket exception class that needs to be implemented.



UML Diagram 1

The construct will be implemented into each different component of the MyPL infrastructure. Three new tokens, 'TRY' , 'THROW', and 'CATCH' would need to be added to token.h. The lexer will have to be altered in order to check for the three new reserved words – 'try', 'throw',and 'catch' – and create tokens for each. These tokens will be added to the Abstract Syntax Tree (AST) as statements.

The TryStmt nodes will contain the capability to store a list of the statements contained in the body of the statement, a nullptr RaiseStmt and a Token containing the id of the MyPL exception to handle. The parser will then check that those values exist in the given environment and will construct new TryStmt nodes in the AST. The RaiseStmt nodes will contain an expression that evaluates to an int, double, or a Boolean. The typechecker will check for these new grammar rules:

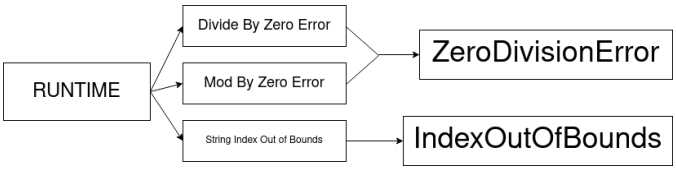
$$\frac{\Gamma \vdash e : t \in \{int, double, bool\}}{\Gamma, raise\ e \vdash e : expr}$$

$$\Gamma, try \ldots catch\ e\ end \vdash e : exception$$

$$\Gamma, try \ldots raise\ e\ \ catch\ e\ end \vdash e : expr$$

The interpreter will implement the try statements by evaluating all statements in the try block until an exception is raised. Then the environment is popped, removing that environment from the scope, then continuing normal execution with the exception having been resolved. Additionally, the generic exception will be implemented in the implementer similarly to the MyPLReturnException, allowing for the user to catch any unforeseen exceptions that occur when running Try-Catch block

The Try-Catch blocks will check primarily for errors that occur during runtime, such as divide by zero errors, as well as index out of bounds errors. UML diagram two has a preliminary list of runtime errors that can be caught in a MyPL Try-Catch block. These will be raised by interpreter in the case of events like divide by zero errors, and the programmer will be able to raise a value to be caught by the catch block.



UML Diagram 2

Here are some example test cases for the Try/Catch block:

Zac Foteff . OPL Final Project: Exception Handling Specification

```
fun bool parity(x:int)
try
   if (x % 2) == 0 then
     raise true
   end
  catch (true) then
   print("Even!")
   return true
  end

  print("Odd!")
  return false
end


fun nil divs (x:double, y:double)
  try
   if (x / y) != 0 then
     x -= 1
     y += 1
   end
  catch (ZeroDivisionError)
    print("Divide by zero!")
  end
end


fun char get_str (in:string, idx:int)
  try
   var char_at_idx = in.get(idx)
  catch (IndexOutOfBounds)
   return ''
  end
end

get_str("hello", 10) --> Error --> ' '
get_str("hello", 2) --> 'l'
```