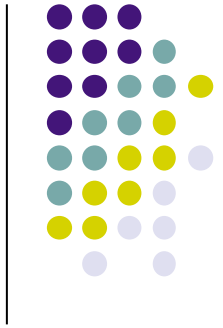
Example using exceptions

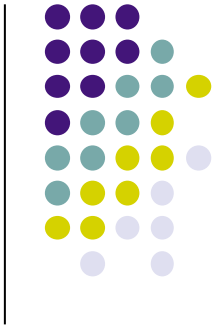


```
fun {Eval E}
   if {IsNumber E} then E
   else
      case E
      of plus(X Y) then {Eval X}+{Eval Y}
      [] times(X Y) then {Eval X}*{Eval Y}
      else raise badExpression(E) end
      end
   end
end

try
   {Browse {Eval plus(23 times(5 5))}}
   {Browse {Eval plus(23 minus(4 3))}}
catch X then {Browse X} end
```

- The error handling code does not clutter up the program

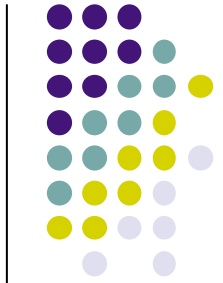
If we did not have exceptions…



```
fun {Eval E}
    if {IsNumber E} then E
    else
        case E
        of plus(X Y) then R={Eval X} in
            case R of badExpression(RE) then badExpression(RE)
            else R2={Eval Y} in
                case R2 of badExpression(RE) then badExpression(RE)
                else R+R2
                end
            end
        [] times(X Y) then
            % … Same code as plus
        else badExpression(E)
        end
    end
end
```

- Much more code!
  - In this example, 22 lines instead of 10 (more than double)
- The code is much more complicated because of all the **case** statements handling badExpression

The “finally” clause



- The **try** has an additional **finally** clause, for an operation that must always be executed (in both the correct and error cases):

```
FH={OpenFile “foobar”}
try
      {ProcessFile FH}
catch X then
      {Show “*** Exception during execution ***”}
finally {CloseFile FH} end % Always close the file
```