Object-Oriented Software Construction
Bertrand Meyer

Lecture 15: Exception handling

Exception handling

- The need for exceptions arises when the contract is broken.
- Two concepts:
  - Failure: a routine, or other operation, is unable to fulfill its contract.
  - Exception: an undesirable event occurs during the execution of a routine — as a result of the failure of some operation called by the routine.

Causes of exceptions

- Assertion violation
- Void call (x.f with no object attached to x)
- Operating system signal (arithmetic overflow, no more memory, interrupt ...)
- Program-triggered

The original strategy

\[
\begin{align*}
  r & \text{ is require} \\
  \ldots & \\
  \text{do} \\
  & \text{op}_1 \\
  & \text{op}_2 \\
  \ldots & \\
  & \text{op}_i \\
  \ldots & \\
  & \text{op}_n \\
  \text{ensure} & \\
  \ldots & \\
  \text{end}
\end{align*}
\]

Fails, triggering an exception in \( r \) (\( r \) is recipient of exception).
Handling exceptions properly

- Safe exception handling principle:
  - There are only two acceptable ways to react for the recipient of an exception:
    - Concede failure, and trigger an exception in the caller (Organized Panic).
    - Try again, using a different strategy (or repeating the same strategy) (Retrying).

How not to do it

(From an Ada textbook)

\[
\text{sqrt (x: REAL) return REAL is}
\begin{align*}
&\text{begin} \\
&\text{if } x < 0.0 \text{ then} \\
&\quad \text{raise Negative;} \\
&\text{else} \\
&\quad \text{normal_square_root_computation;} \\
&\text{end}
\end{align*}
\]

\[
\text{exception}
\begin{align*}
&\text{when Negative =>} \\
&\quad \text{put ("Negative argument");} \\
&\quad \text{return;} \\
&\text{when others => ...}
\end{align*}
\]

Exception mechanism

- Two constructs:
  - A routine may contain a rescue clause.
  - A rescue clause may contain a retry instruction.

- A rescue clause that does not execute a retry leads to failure of the routine (this is the organized panic case).
Transmitting over an unreliable line (1)

Max_attempts: INTEGER is 100

attempt_transmission (message: STRING) is
  -- Transmit message in at most
  -- Max_attempts attempts.
local
  failures: INTEGER
do
  unsafe_transmit (message)
rescue
  failures := failures + 1
  if failures < Max_attempts then
    retry
  end
end

Transmitting over an unreliable line (2)

Max_attempts: INTEGER is 100
failed: BOOLEAN

attempt_transmission (message: STRING) is
  -- Try to transmit message;
  -- if impossible in at most Max_attempts
  -- attempts, set failed to true.
local
  failures: INTEGER
do
  if failures < Max_attempts then
    unsafe_transmit (message)
  else
    failed := True
  end
rescue
  failures := failures + 1
  retry
end

If no exception clause (1)

- Absence of a rescue clause is equivalent, in first approximation, to an empty rescue clause:
  
  \[
  f(\ldots) \text{ do} \quad \text{end} \quad \ldots
  \]

  is an abbreviation for
  
  \[
  f(\ldots) \text{ do} \quad \text{rescue} \quad \text{end} \quad \text{Nothing here}
  \]

  (This is a provisional rule; see next.)

The correctness of a class

- (1-n) For every exported routine \( r \):
  
  \{INV and Pre\} do \{Post and INV\}

- (1-m) For every creation procedure \( cp \):
  
  \{Pre\} do\( \# \) \{Post\} and INV
Exception correctness: A quiz

- For the normal body:
  \{INV and Pre.\} do. \{Post and INV\}

- For the exception clause:
  \{ ??? \} rescue. \{ ??? \}

If no exception clause (2)

- Absence of a rescue clause is equivalent to a default rescue clause:
  \[
  \begin{align*}
  f (...) & \text{ is} \\
  & \text{do} \\
  & \text{end} \\
  \end{align*}
  \]
  is an abbreviation for
  \[
  \begin{align*}
  f (...) & \text{ do} \\
  & \text{rescue} \\
  & \text{end} \\
  \text{default\_rescue} \\
  \end{align*}
  \]

- The task of \textit{default\_rescue} is to restore the invariant.

Quiz answers

- For the normal body:
  \{INV and Pre.\} do. \{Post and INV\}

- For the exception clause:
  \{True\} rescue. \{INV\}

For finer-grain exception handling

- Use class \textit{EXCEPTIONS} from the Kernel Library.

- Some features:
  - \textit{exception} (code of last exception that was triggered).
  - \textit{assertion\_violation}, etc.
  - \textit{raise} ("\textit{exception\_name}")
End of lecture 15