Object-Oriented Software Construction

Bertrand Meyer
Lecture 24: 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.
The original strategy

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

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

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

```
sqrt (x: REAL) return REAL is
  begin
    if x < 0.0 then
      raise Negative;
    else
      normal_square_root_computation;
    end
  exception
    when Negative =>
      put ("Negative argument");
      return;
    when others => ...
  end; -- sqrt
```
The call chain
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)

\[\text{Max_attempts: INTEGER is 100}\]

\[\text{attempt_transmission (message: STRING) is}\]

\[\text{-- Transmit message in at most}\]

\[\text{-- Max_attempts attempts.}\]

\[\text{local}\]

\[\text{failures: INTEGER}\]

\[\text{do}\]

\[\text{unsafe_transmit (message)}\]

\[\text{rescue}\]

\[\text{failures := failures + 1}\]

\[\text{if failures < Max_attempts then}\]

\[\text{retry}\]

\[\text{end}\]

\[\text{end}\]
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:

\[
\text{f (...) is} \\
\text{do} \\
\text{...} \\
\text{end}
\]

is an abbreviation for

\[
\text{f (...) is} \\
\text{do} \\
\text{...} \\
\text{rescue} \\
\text{-- Nothing here} \\
\text{end}
\]

- (This is a provisional rule; see next.)
The correctness of a class

- (1-n) For every exported routine $r$:
  \[
  \{\text{INV and Pre}_r\} \text{ do}_r \{\text{Post}_r \text{ and INV}\}
  \]

- (1-m) For every creation procedure $cp$:
  \[
  \{\text{Pre}_cp\} \text{ do}_{cp} \{\text{Post}_{cp} \text{ and INV}\}
  \]
Exception correctness: A quiz

- For the normal body:
  \[ \{ \text{INV and Pre}_r \} \text{ do}_r \{ \text{Post}_r \text{ and INV} \} \]

- For the exception clause:
  \[ \{ \text{ ??? } \} \text{ rescue}_r \{ \text{ ??? } \} \]
Quiz answers

- For the normal body:
  \{	ext{INV and Pre}_r\} \text{ do}_r \{\text{Post}_r \text{ and INV}\}

- For the exception clause:
  \{\text{True}\} \text{ rescue}_r \{\text{INV}\}
Absence of a rescue clause is equivalent to a default rescue clause:

\[
\begin{align*}
  f (\ldots) & \textbf{is} \\
  & \textbf{do} \\
  & \textbf{...} \\
  & \textbf{end}
\end{align*}
\]

is an abbreviation for

\[
\begin{align*}
  f (\ldots) & \textbf{is} \\
  & \textbf{do} \\
  & \textbf{...} \\
  & \textbf{rescue} \\
  & \textbf{default_rescue} \\
  & \textbf{end}
\end{align*}
\]

The task of \textit{default_rescue} is to restore the invariant.
For finer-grain exception handling

- Use class *EXCEPTIONS* from the Kernel Library.

- Some features:
  - *exception* (code of last exception that was triggered).
  - *assertion_violation*, etc.
  - *raise* ("exception_name")
End of lecture 24