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.
The original strategy

\[
\begin{align*}
 r \ (\ldots) \ & \text{is} \\
 \text{require} \\
 \quad \ldots \\
 \text{do} \\
 \quad op_1 \\
 \quad op_2 \\
 \quad \ldots \\
 \quad op_i \quad \text{Fails, triggering an exception in } r \\
 \quad \ldots \\
 \text{ensure} \\
 \quad \ldots \\
 \text{end}
\end{align*}
\]

(\text{\(r\) 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 …)
- Program-triggered
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)

```ada
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)

\[ \text{Max\_attempts: } \text{INTEGER is } 100 \]

\[ \text{attempt\_transmission (message: STRING) is} \]
\[ \quad \text{-- Transmit message in at most} \]
\[ \quad \text{-- Max\_attempts attempts.} \]
\[ \text{local} \]
\[ \quad \text{failures: INTEGER} \]
\[ \text{do} \]
\[ \quad \text{unsafe\_transmit (message)} \]
\[ \text{rescue} \]
\[ \quad \text{failures := failures + 1} \]
\[ \quad \text{if failures < Max\_attempts then} \]
\[ \quad \quad \text{retry} \]
\[ \quad \text{end} \]
\[ \text{end} \]
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:

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

  is an abbreviation for

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

- (This is a provisional rule; see next.)
(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{ rescue}_r \{ ??? \}
Quiz answers

- For the normal body:

\{INV \text{ and } \text{Pre}_r\} \text{ do}_r \{\text{Post}_r \text{ and } \text{INV}\}

- For the exception clause:

\{\text{True}\} \text{ rescue}_r \{\text{INV}\}
If no exception clause (2)

- Absence of a rescue clause is equivalent to a default rescue clause:

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

  is an abbreviation for

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

- The task of \text{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 15