Programozási nyelvek és paradigmák

Kivételkezelés

Kozsik Tamás (2020)

Kivétel

- Extrém esemény, mely kibillenti a programot a normál végrehajtásból
  - A számítógép vagy az operációs rendszer jelzése
  - Dinamikus szemantikai hiba
  - Szerződésszegés
  - A programunk is kiválthat: ritka, külön kezelt logikai eset
- Ha nem kezeljük le, terjed a hívási lánc mentén
- Program leállását eredményezheti
- Stack trace
- Robusztusság: kivételkezelés
- Kivételkezelést támogató nyelvi eszközök (LISP, PL/1, CLU...)

Kivételkezelés Eiffelben

- Eiffelben előre tervezünk, nem utólag kezelünk
- Rutin hívása előtt az előfeltételeket biztosítjuk
- Felelőségkezelés a kivételkezelés helyett

Kivételkezelés Eiffelben

- Eiffelben előre tervezünk, nem utólag kezelünk
- Rutin hívása előtt az előfeltételeket biztosítjuk
- Felelőségkezelés a kivételkezelés helyett

- Vannak esetek, amikor mégis kell kivételkezelés
- De tényleg csak kivételes esetben

Kivételkezelő rész Eiffelben

Attribute_or_routine :=	[Documentation_comment] [Precondition] [Local_declarations] Feature_body [Postcondition] [Rescue] **end**

- Rutinhoz tartozik
- *Rescue-klóz*: utasítássorozat
  - Lehet benne `retry` utasítás

Végrehajtás menete

- Sikertelen rutintörzs után `rescue`
- Ha a `rescue` során `retry`, akkor újra rutintörzs
- Ha a `rescue` során nincs `retry`, a kivétel terjed a hívóba
- Újrapróbálkozásnál a lokális változók értéke megmarad
- Első próbálkozásnál lokális változók alapértelmezett értéken

Idióma: művelet alternatív implementációval

```
my_routine
    local
        already_tried: BOOLEAN
    do
        if not already_tried then
            -- execute normal operation
        else
            -- execute Plan B
        end
    rescue
        if not already_tried then
            already_tried := True
            retry
        end
    end
```

Idióma: néhányszor újrapróbálkozás

```
attempt_transmission (message: STRING)
        -- Try to transmit message, at most 50 times.
        -- Set `successful` accordingly.
    local
        failures: INTEGER
    do
        if failures < 50 then
            transmit (message)      -- this may fail
            successful := True
        else
            successful := False
        end
    rescue
        failures := failures + 1
        retry
    end
```

Kivétel-helyes program

Minden C osztályban minden r rutinra:

- Organized panic
  - `retry` nélkül befejeződő `rescue`
  - Az utófeltételt nem tudtuk elérni
  - Az osztályinvariánst vissza kell állítani

$$\{\mathrm{True}\}\ \mathrm{rescue\_without\_retry}_r\ \{\mathrm{Inv}_C\}$$

- Újrapróbálkozás
  - `retry` hatására a törzs újraindul
  - Az előfeltételt biztosítani kell (nincs az ECMA-szabványban!)

$$\{\mathrm{True}\}\ \mathrm{rescue\_with\_retry}_r\ \{\mathrm{Inv}_C \wedge \mathrm{pre}_r\}$$

Megj: Inv_C a teljes osztályinvariáns, pre_r a teljes előfeltétele r-nek.

Idióma: objektum helyreállítása

```
attempt_transaction (arg: CONTEXT)
        -- Try transaction with arg;
        -- if impossible, reset current object
    require
        ...
    do
        ...
    ensure
        ...
    rescue
        reset (arg)
    end
```

Alapértelmezett rescue-klóz

- Ha egy rutinban nincs explicit `rescue`, akkor az `ANY`-ből megörökölt (esetleg átnevezett) `default_rescue` rutin
- Az `ANY`-ben a `default_rescue` törzse üres
- Az öröklődés során felüldefiniálható a `default_rescue`
- Nem szerepelhet benne a `retry` utasítás (mármint “úgy”)
- Az osztályunk alapértelmezett kivételkezelési stratégiája
- Lehet például ugyanaz, mint a `default_create`

Kivételkezeléshez extrák

```
class CONNECTION
inherit
    EXCEPTIONS
...
    ... raise( "Communication_failure" ) ...
...
end
```

EXCEPTIONS – kivételek értelmezése

```
class EXCEPTIONS
inherit EXCEP_CONST
...
feature
    assertion_violation: BOOLEAN
    is_signal: BOOLEAN
    is_system_exception: BOOLEAN
    is_developer_exception: BOOLEAN
    is_developer_exception_of_name (name: detachable STRING):
                                                        BOOLEAN
    developer_exception_name: detachable STRING
    tag_name: detachable STRING
    recipient_name: detachable STRING
    class_name: detachable STRING
    exception_trace: detachable STRING
    exception: INTEGER
end
```

class EXCEP_CONST – standard kivételek kódja

```
feature
    Void_call_target: INTEGER = 1
    No_more_memory: INTEGER = 2
    Precondition: INTEGER = 3
    Postcondition: INTEGER = 4
    Floating_point_exception: INTEGER = 5
    Class_invariant: INTEGER = 6
    Check_instruction: INTEGER = 7
    Routine_failure: INTEGER = 8
    Incorrect_inspect_value: INTEGER = 9
    Loop_variant: INTEGER = 10
    Loop_invariant: INTEGER = 11
    Signal_exception: INTEGER = 12
    Eiffel_runtime_panic: INTEGER = 13
    ...
end
```

EXCEPTIONS – kivételek kiváltása és alapértelmezett kezelése

```
class EXCEPTIONS
inherit EXCEP_CONST
...
feature
    ...
    exception: INTEGER
    raise (name: detachable STRING)
    meaning (except: INTEGER): detachable STRING
    catch (code: INTEGER)
    ignore (code: INTEGER)
    die (code: INTEGER)
end
```