Slides to Accompany *Programming Languages and Methodologies*

R. J. Schalkoff

Chapter 11, `ocaml` Exceptions

Exceptions in the Pervasives module

```
val raise : exn -> 'a
Raise the given exception value

val invalid_arg : string -> 'a
Raise exception Invalid_argument with the given string.

val failwith : string -> 'a
Raise exception Failure with the given string.

exception Exit
The Exit exception is not raised by any library function.
It is provided for use in your programs.
```

Example

```
(* file: expr2r1.caml
   a to the b, as per book specification
   exception example
   no side effects *)

#use "odd-even-mut-rec.caml";; (* to get function even *)

let rec expr2 a b =
if (b<0)
   then failwith "this version requires non-negative b"
   else if (b==0)
           then 1
           else if (even b)
                   then (expr2 (a * a) (b / 2))
                   else (a * (expr2 a (b - 1)));;
```

Use

```
        OCaml version 4.00.0

# #use"expr2r1.caml";;
val even : int -> bool = <fun>
val odd : int -> bool = <fun>
val expr2 : int -> int -> int = <fun>

# expr2 2 4;;
- : int = 16

# expr2 2 (-4);;
Exception: Failure "this version requires non-negative b"
```

More General Exceptions

- Exceptions are declared with the `exception` construct, and signalled with the `raise` operator.
- There are built-in exceptions (RTM)
- You may define new exceptions via:

```
exception-definition ::=
          exception constr-name  [of typexpr  { * typexpr }]  |
          exception constr-name =  constr
```

Example

```
(* file: expr2r2.caml
   a to the b, as per book specification
   2nd exception example
   no side effects *)

#use "odd-even-mut-rec.caml";; (* to get function even *)

(** define exception for negative b *)
exception BisOutOfBounds;;

let rec expr2 a b =
if (b<0)
   then raise BisOutOfBounds
   else if (b==0)
           then 1
           else if (even b)
                   then (expr2 (a * a) (b / 2))
                   else (a * (expr2 a (b - 1)));;
```

```
(** L.A.E.: BisOutOfBounds="this version requires non-negative b" *)
```

Use

```
# #use"expr2r2.caml";;
val even : int -> bool = <fun>
val odd : int -> bool = <fun>
exception BisOutOfBounds
val expr2 : int -> int -> int = <fun>

# expr2 2 4;;
- : int = 16

# expr2 2 (-4);;
Exception: BisOutOfBounds.
```