Signatures and Structures

- **Signature** in SML ≈ **Interface** in Java
- **Structure** in SML ≈ **Class** in Java

Signature

- defines an **interface**, like a Java interface
- unlike Java interfaces, may define **0**, **1** or **more** types
- unlike Java interfaces, **hides** everything outside the signature

Example

```
signature RAT = sig
    type rat
    exception DivisionByZero
    val makeRat : int * int -> rat
    val plus : rat * rat -> rat
    val minus : rat * rat -> rat
    val times : rat * rat -> rat
    val inverse : rat -> rat
    val toString : rat -> string
end
```

Structure

- defines an **implementation**, like a Java class
- unlike Java classes, may define **0**, **1** or **more** types
- unlike Java classes, **everything** not hiden with a **signature** is visible

Example

```
structure Rat : RAT = struct

type rat = int * int
exception DivisionByZero

fun gcd(0, m) = m
  | gcd(n, m) = gcd(m mod n, n)
fun makeRat (_, 0) = raise DivisionByZero
  | makeRat (x, y) = if y < 0 then makeRat (~x, ~y)
                     else let val g = gcd(y,x) in
                              (x div g, y div g)
                          end
fun plus ((x, y), (z, t)) = makeRat (x*t+z*y, y*t)
(* remaining functions ... *)
end
```

Remarks

- Without implementing a signature, **everything** defined in a structure is visible:

Example

```
structure Rat = struct (* same as before *) end
```

function `Rat.gcd` is visible outside the structure

-
  ```
  sig
    type 'a bTree
    val left : 'a bTree -> 'a bTree
    ...
  end
  ```

  makes the type `'a bTree` available through the functions of the structure but the constructors (Node, EmptyTree) are **hidden**

-
  ```
  sig
    datatype 'a bTree = EmptyTree | Node of 'a * 'a bTree
    val left : 'a bTree -> 'a bTree
    ...
  end
  ```

  makes the constructors **available** outside the structure

- Using **eqtype** instead of **type** defines a type with **equality** (`''a type`)
- Using `:>` instead of `:` makes defined types **unique** to the structure
- `sig ... end` represents an **anonymous signature**:

Example

```
structure Rat :
  sig
    type rat
    exception DivisionByZero
    val makeRat : int * int -> rat
    ...
  end
= struct (* implementation here *) end
```

- `struct ... end` represents an **anonymous structure**:

Example

```
structure Calc = Calculator (struct ... end)
```

Examples of Signatures and Structures

```
- signature S = sig
= type t
= val f : t -> t
= val x : t
= end;
signature S =
  sig
    type t
    val f : t -> t
    val x : t
  end
- structure S1 : S = struct
= type t = int
= val x = 0
= fun f x = x+1
= end;
structure S1 : S
- S1.f;
val it = fn : S1.t -> S1.t
- S1.f 5;
val it = 6 : S1.t
```

```
- structure S2 :> S = struct
= type t = int
= val x = 0
= fun f x = x+1
= end;
structure S2 : S
- S2.f;
val it = fn : S2.t -> S2.t
- S2.f 5;
stdIn:25.1-25.7 Error: operator and operand don't agree [literal]
  operator domain: S2.t
  operand:         int
  in expression:
    S2.f 5
- S1.x = S1.x;
val it = true : bool
- S2.x = S2.x;
stdIn:26.1-26.12 Error: operator and operand don't agree [equality type required]
  operator domain: ''Z * ''Z
  operand:         S2.t * S2.t
  in expression:
    S2.x = S2.x
```

```
- signature S' = sig
= eqtype t
= val x : t
= val f : t -> t
= end;
signature S' =
  sig
    eqtype t
    val x : t
    val f : t -> t
  end
- structure S2' :> S' = struct
= type t = int
= val x = 0
= fun f x = x+1
= end;
structure S2' : S'
- S2'.x = S2'.x;
val it = true : bool
- S2'.x = 4;
stdIn:44.1-44.10 Error: operator and operand don't agree [literal]
  operator domain: S2'.t * S2'.t
  operand:         S2'.t * int
  in expression:
    S2'.x = 4
```

```
- structure S3 : S = struct
= datatype t = TRUE | FALSE
= val x = FALSE
= fun f TRUE = FALSE | f FALSE = TRUE
= end;
structure S3 : S
- S3.x;
val it = FALSE : S3.t
- local
= open S3
= in
= val a = f x
= val b = f a
= end;
val a = TRUE : S3.t
val b = FALSE : S3.t
- local
= open S3
= in
= fun g TRUE = FALSE | g FALSE = TRUE
= end;
stdIn:59.14-59.19 Error: unbound variable or constructor: FALSE
stdIn:59.32-59.36 Error: unbound variable or constructor: TRUE
```

```
- structure S4 :> S = struct
= datatype t = TRUE | FALSE
= val x = FALSE
= fun f TRUE = FALSE | f FALSE = TRUE
= end;
structure S4 : S
- S4.x;
val it = - : S4.t
- local
= open S4
= in
= val a = f x
= val b = f a
= end;
val a = - : S4.t
val b = - : S4.t
- S4.f b;
val it = - : S4.t
```