```
// public struct. visible everywhere, even externally (if exported)
pub struct MyStruct {
  // define the properties of the struct
  // all variables with no 'pri' before them are publicly accessible,
  // with 'pub' properties externally visible
  properties {
    pub int a
    pri dint a_dyn
    str name
  }

  // public constructor. visible everywhere, even externally
  pub fn new(int a, str name) -> Self {
    set a_dyn = a as dint
    return Self { a: a, a_dyn: a_dyn, name: name }
  }

  // exposed function. visible only within the package
  fn sqrt(*self) -> Option<int> {
    return self.try_sqrt().ok()
  }

  // private function; not public. only to be used by the struct 'internally'
  pri fn try_sqrt(*self) -> Result<int, null> {
    return self.a.try_sqrt().into()
  }
}

// Extend MyStruct with Into<int> allowing you to use
// `MyStruct as int`
extend *MyStruct with Into<int> {
  fn into(self) -> int {
    return self.a
  }
}

enum MyEnum {
  variants { }
}

pri mod constants {
  const float PI = 3.14159_26535
}

use constants::PI

fn main() {
  set fib_10 = fibonacci(10)
  for fib in fib_10.iter() { print(fib) }
}

/// Fibonacci function
/// Returns the first `reps` fibonacci numbers starting [0, 1]
/// @param(reps) Number of fibonacci numbers to return
fn fibonacci(uint reps) -> [int] {
  match reps {
    0 => return []
    1 => return [0]
    2 => return [0, 1]
  }
  set out = [0, 1]
  set low = 0
  set high = 1
```

```
  reps = (reps - 2).max(0)
  repeat reps {
    set new = low + high
    out.push(new)
    low = high
    high = new
  }
  return out
}

extension MyExtension<T> {
  fn into(*self) -> T
}
```