

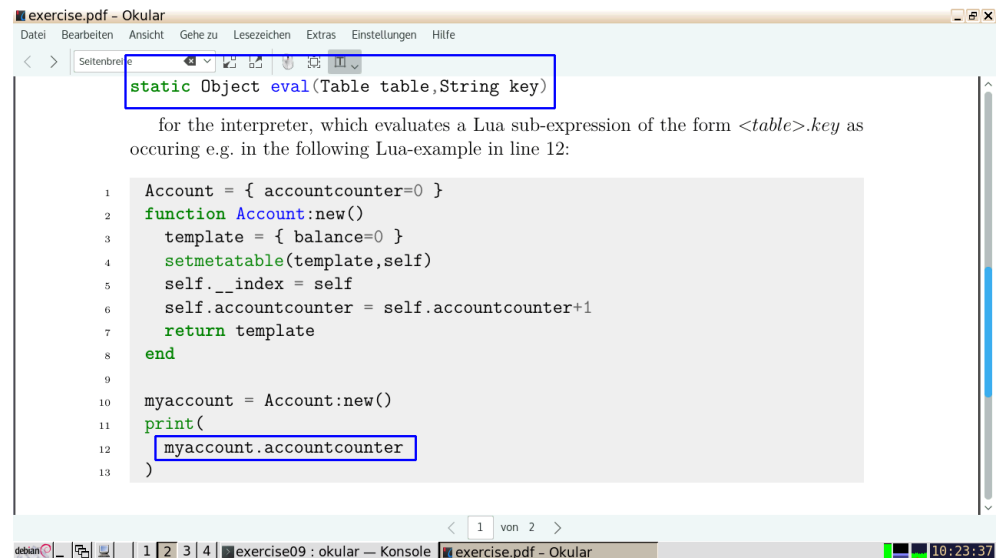
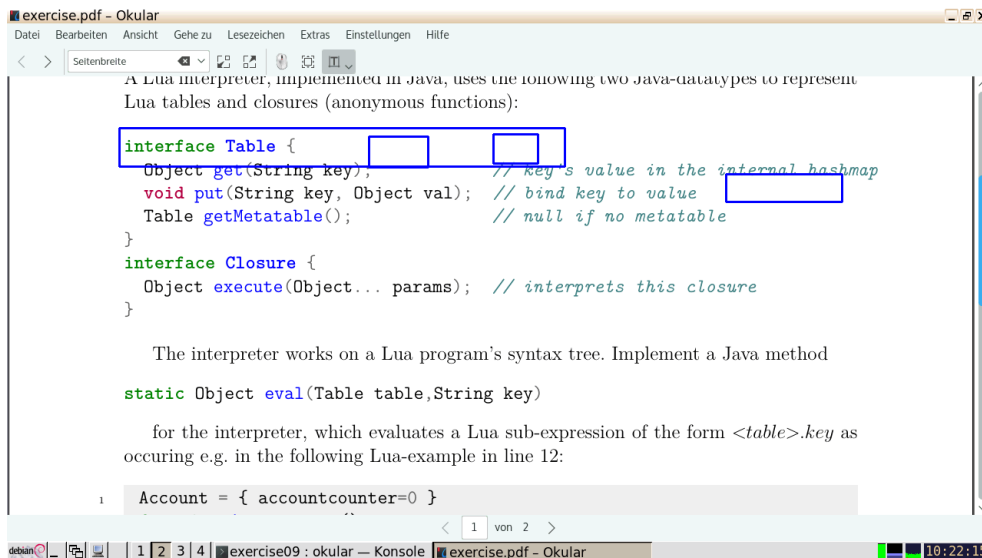
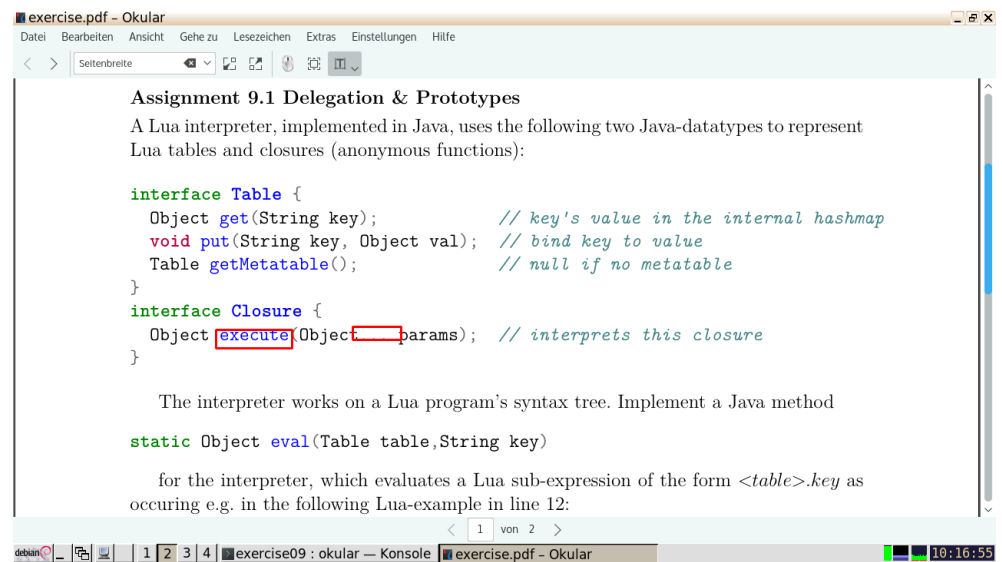
Script generated by TTT

Title: Petter: Programmiersprachen_Uebung
(03.02.2017)

Date: Fri Feb 03 10:16:55 CET 2017

Duration: 100:08 min

Pages: 68



exercise.pdf - Okular

Datei Bearbeiten Ansicht Gehe zu Lesezeichen Extras Einstellungen Hilfe

Seitenbreite

The interpreter works on a Lua program's syntax tree. Implement a Java method

```
static Object eval(Table table,String key)
```

for the interpreter, which evaluates a Lua sub-expression of the form <table>.key as occurring e.g. in the following Lua-example in line 12:

```
1 Account = { accountcounter=0 }
2 function Account:new()
3   template = { balance=0 }
4   setmetatable(template,self)
5   self._index = self
6   self.accountcounter = self.accountcounter+1
7   return template
8 end
9
10 myaccount = Account:new()
11 print(
12   myaccount.accountcounter
```

1 von 2

exercise09 : okular — Konsole

Datei Bearbeiten Ansicht Lesezeichen Einstellungen Hilfe

```
petter@michaels-t420s:/home/petter$ cd lehre/proglang/
.git/ slides/ slides2012/ slides2013/ slides2014/ slides2015/ uebungen/
petter@michaels-t420s:/home/petter$ cd lehre/proglang/uebungen/
2014/ concurrent/ inheritance/ klausur2014/ locks/ muendlich/ template/ traits/
2015/ concur_summary/ klausur2012/ klausur2015/ mes/ primes/ test/
2016/ dispatching/ klausur2013/ klausur2016/ mixins/ prototypes/ tm/
petter@michaels-t420s:/home/petter$ cd lehre/proglang/uebungen/2016/exercise09
exercise.aex exercise.fdb latexmk exercise.log exercise.pdf exercise.tex exercise.w18 _minted-exercise
petter@michaels-t420s:/home/petter/lehre/proglang/uebungen/2016/exercise09$ ls
exercise.aux exercise.fls exercise.out exercise.pyg exercise.tex.deps Makefile svg
petter@michaels-t420s:/home/petter/lehre/proglang/uebungen/2016/exercise09$ okular exercise.pdf
X Error: BadDrawable (invalid Pixmap or Window parameter) 9
Major opcode: 62 (X_CopyArea)
Resource id: 0x0
Connecting to deprecated signal QDBusConnectionInterface::serviceOwnerChanged(QString,QString,QString)
QDBusConnection: session D-Bus connection created before QCoreApplication. Application may misbehave.
QDBusConnection: session D-Bus connection created before QCoreApplication. Application may misbehave.
Xlib: extension "XInputExtension" missing on display ":1".
X Error: BadDrawable (invalid Pixmap or Window parameter) 9
Major opcode: 62 (X_CopyArea)
Resource id: 0x0
kbldsyncoc4 running...
X Error: BadDrawable (invalid Pixmap or Window parameter) 9
Major opcode: 62 (X_CopyArea)
Resource id: 0x0
QDBusConnection: name 'org.ally.atspi.Registry' had owner '' but we thought it was ':1.2'
Bus::open: Can not get ibus-daemon's address
IBusInputContext::createInputContext: no connection to ibus-daemon
```

exercise09 : okular

emacs@michaels-t420s

```
Stream.class Synchron.class
Streams.java
Streams.java-
```

1 2 3 4 tutorial : bash — Konsole emacs@michaels-t420s 10:25:20

emacs@michaels-t420s

```
1 public class Evaluate {
2
3 }
```

```
Stream.class Synchron.class
Streams.java
Streams.java-
```

1 2 3 4 tutorial : bash — Konsole emacs@michaels-t420s 10:25:57

```
emacs@michaels-t420s
1 public class Evaluate {
2   static Object eval()
3 }
```

Stream.class Synchron.class
Streams.java
Streams.java-

debian 1 2 3 4 tutorial : bash — Konsole emacs@michaels-t420s 10:26:13

```
emacs@michaels-t420s
1 public class Evaluate {
2   static Object eval(Table table, String key){
3
4   }
5 }
6
7
8 // table.key -> table["key"]
```

Stream.class Synchron.class
Streams.java
Streams.java-

debian 1 2 3 4 tutorial : bash — Konsole emacs@michaels-t420s 10:27:50

```
emacs@michaels-t420s
1 public class Evaluate {
2   static Object eval(Table table, String key){
3     Object ret = table.get(key);
4
5     return ret;
6   }
7 }
8
9
10 // table.key -> table["key"]
```

Stream.class Synchron.class
Streams.java
Streams.java-

debian 1 2 3 4 tutorial : bash — Konsole emacs@michaels-t420s 10:29:14

```
emacs@michaels-t420s
1 public class Evaluate {
2   static Object eval(Table table, String key){
3     Object ret = table.get(key);
4     Table meta = table.getmetatable();
5     return ret;
6   }
7 }
8
9
10 // table.key -> table["key"]
```

Stream.class Synchron.class
Streams.java
Streams.java-

debian 1 2 3 4 tutorial : bash — Konsole emacs@michaels-t420s 10:29:32

exercise.pdf - Okular

Assignment 9.1 Delegation & Prototypes

A Lua interpreter, implemented in Java, uses the following two Java-datypes to represent Lua tables and closures (anonymous functions):

```
interface Table {
    Object get(String key);           // key's value in the internal hashmap
    void put(String key, Object val); // bind key to value
    Table getMetatable();           // null if no metatable
}

interface Closure {
    Object execute(Object... params); // interprets this closure
}
```

The interpreter works on a Lua program's syntax tree. Implement a Java method

```
static Object eval(Table table, String key)
```

for the interpreter, which evaluates a Lua sub-expression of the form `<table>.key` as occurring e.g. in the following Lua-example in line 12:

1 von 2

emacs@michaels-t420s

```
1 public class Evaluate {
2     static Object eval(Table table, String key){
3         Object ret = table.get(key);
4         if (ret != null) return ret;
5         Table meta = table.getMetatable();
6
7         return ret;
8     }
9 }
10
11
12 // table.key -> table["key"]
```

Stream.class Synch.class
Streams.java
Streams.java-

1 2 3 4 tutorial : bash — Konsole emacs@michaels-t420s 10:30:09

exercise.pdf - Okular

Assignment 9.1 Delegation & Prototypes

A Lua interpreter, implemented in Java, uses the following two Java-datypes to represent Lua tables and closures (anonymous functions):

```
interface Table {
    Object get(String key);           // key's value in the internal hashmap
    void put(String key, Object val); // bind key to value
    Table getMetatable();           // null if no metatable
}

interface Closure {
    Object execute(Object... params); // interprets this closure
}
```

The interpreter works on a Lua program's syntax tree. Implement a Java method

```
static Object eval(Table table, String key)
```

for the interpreter, which evaluates a Lua sub-expression of the form `<table>.key` as occurring e.g. in the following Lua-example in line 12:

1 von 2

emacs@michaels-t420s

```
1 public class Evaluate {
2     static Object eval(Table table, String key){
3         Object ret = table.get(key);
4         if (ret != null) return ret;
5         Table meta = table.getMetatable();
6         if (meta == null) return null; // return nil
7         Table delegationTarget = meta.get("__index");
8         if (delegationTarget == null) return null;
9         ret = eval(delegationTarget, key);
10        return ret;
11    }
12 }
13
14
15 // table.key -> table["key"]
```

Stream.class Synch.class
Streams.java
Streams.java-

lay ":1".

1 2 3 4 tutorial : bash — Konsole emacs@michaels-t420s 10:37:07

```
emacs@michaels-t420s
1 public class Evaluate {
2     static Object eval(Table table, String key){
3         Object ret = table.get(key);
4         if (ret != null) return ret;
5         Table meta = table.getMetatable();
6         if (meta == null) return null; // return nil
7         Object index = meta.get("__index");
8         if (index == null) return null;
9         if (index instanceof Table ) {
10            Table delegationTarget = (Table)index;
11            ret = eval(delegationTarget,key);
12        }
13        if (index instanceof Closure) {
14            Closure delegationFunction = (Closure)index;
15            eval(_
16
17            return ret;
18        }
19    }
20 }
21
22 // table.key -> table["key"]
```

```
Stream.class Synchron.class
Streams.java
Streams.java-
lay ":1".
```

exercise.pdf - Okular

Assignment 9.1 Delegation & Prototypes

A Lua interpreter, implemented in Java, uses the following two Java-datatypes to represent Lua tables and closures (anonymous functions):

```
interface Table {
    Object get(String key);           // key's value in the internal hashmap
    void put(String key, Object val); // bind key to value
    Table getMetatable();           // null if no metatable
}

interface Closure {
    Object execute(Object... params); // interprets this closure
}
```

The interpreter works on a Lua program's syntax tree. Implement a Java method

```
static Object eval(Table table, String key)
```

for the interpreter, which evaluates a Lua sub-expression of the form `<table>.key` as occurring e.g. in the following Lua-example in line 12:

```
emacs@michaels-t420s
1 public class Evaluate {
2     static Object eval(Table table, String key){
3         Object ret = table.get(key);
4         if (ret != null) return ret;
5         Table meta = table.getMetatable();
6         if (meta == null) return null; // return nil
7         Object index = meta.get("__index");
8         if (index == null) return null;
9         if (index instanceof Table ) {
10            Table delegationTarget = (Table)index;
11            ret = eval(delegationTarget,key);
12        }
13        if (index instanceof Closure) {
14            Closure delegationFunction = (Closure)index;
15            delegationFunction.exec(_
16
17            return ret;
18        }
19    }
20 }
21
22 // table.key -> table["key"]
```

```
Stream.class Synchron.class
Streams.java
Streams.java-
lay ":1".
```

exercise.pdf - Okular

Assignment 9.1 Delegation & Prototypes

A Lua interpreter, implemented in Java, uses the following two Java-datatypes to represent Lua tables and closures (anonymous functions):

```
interface Table {
    Object get(String key);           // key's value in the internal hashmap
    void put(String key, Object val); // bind key to value
    Table getMetatable();           // null if no metatable
}

interface Closure {
    Object execute(Object... params); // interprets this closure
}
```

The interpreter works on a Lua program's syntax tree. Implement a Java method

```
static Object eval(Table table, String key)
```

for the interpreter, which evaluates a Lua sub-expression of the form `<table>.key` as occurring e.g. in the following Lua-example in line 12:

exercise.pdf - Okular

Assignment 9.2 Traits in Lua

Trait composition $+$ is defined as a symmetric join \sqcup between two maps c_1, c_2 :

$$(c_1 + c_2)(n) = b_1 \sqcup b_2 = \begin{cases} b_2 & \text{if } b_1 = \perp \vee n \notin \text{pre}(c_1) \\ b_1 & \text{if } b_2 = \perp \vee n \notin \text{pre}(c_2) \\ b_2 & \text{if } b_1 = b_2 \\ \top & \text{otherwise} \end{cases} \quad \text{with } b_i = c_i(n)$$

The following Lua function dispatches lookups for key k from map **receiver** to the two maps m_1, m_2 in an ordered fashion with priority on m_1 :

```
function asymmetricDispatch (receiver, k)
  local v = receiver.m1[k]
  if not v then return receiver.m2[k] end
  return v
end
```

- provide a Lua implementation of the function `symmetricDispatch`, which implements dispatching of key k based on the symmetric join \sqcup .

10:49:45

exercise.pdf - Okular

$$(c_1 + c_2)(n) = b_1 \sqcup b_2 = \begin{cases} b_2 & \text{if } b_1 = \perp \vee n \notin \text{pre}(c_1) \\ b_1 & \text{if } b_2 = \perp \vee n \notin \text{pre}(c_2) \\ b_2 & \text{if } b_1 = b_2 \\ \top & \text{otherwise} \end{cases} \quad \text{with } b_i = c_i(n)$$

The following Lua function dispatches lookups for key k from map **receiver** to the two maps m_1, m_2 in an ordered fashion with priority on m_1 :

```
function asymmetricDispatch (receiver, k)
  local v = receiver.m1[k]
  if not v then return receiver.m2[k] end
  return v
end
```

- provide a Lua implementation of the function `symmetricDispatch`, which implements dispatching of key k based on the symmetric join \sqcup .
- use this function to implement a function `composeTraits`, which takes a pair of

10:52:24

tutorial : bash — Konsole

```
petter@michaels-t420s:~/Lehre$ cd tutorial/
petter@michaels-t420s:~/Lehre/tutorial$ ls
bumper      bumper.c-      bumper-semaphore.c  dekker.c  DQueue.c-  FileStream.class  NetworkStream.class  Stream.class  Synch.class
bumper#     bumper-fetchandadd.c  bumper-semaphore.c-  dekker.c-  FileStream.class  sheet04.pdf      Streams.java
bumper.c    bumper-semaphore     dekker               DQueue.c   mjc-1_3_2.jar  sheet06.pdf      Streams.java-
petter@michaels-t420s:~/Lehre/tutorial$ emacs eval.lua
libGL error: No matching fbConfigs or visuals found
libGL error: failed to load driver: swrast
petter@michaels-t420s:~/Lehre/tutorial$ emacs Evaluate.java &
[1] 3055
petter@michaels-t420s:~/Lehre/tutorial$ Xlib: extension "XInputExtension" missing on display ":1".
Xlib: extension "XInputExtension" missing on display ":1".
[]
```

10:53:07

emacs@michaels-t420s

```
Stream.class  Synch.class
Streams.java
Streams.java-

lay ":1".

lay ":1".
```

10:53:21

emacs@michaels-t420s

```
function
end
```

Stream.class Synchron.class
Streams.java
Streams.java-

lay ":1".

lay ":1".

debian 1 2 3 4 tutorial : bash — Konsole emacs@michaels-t420s 10:53:43

exercise.pdf - Okular

Datei Bearbeiten Ansicht Gehe zu Lesezeichen Extras Einstellungen Hilfe

Seitenbreite

$$(c_1 + c_2)(n) = b_1 \sqcup b_2 = \begin{cases} b_2 & \text{if } b_1 = \perp \vee n \notin \text{pre}(c_1) \\ b_1 & \text{if } b_2 = \perp \vee n \notin \text{pre}(c_2) \\ b_2 & \text{if } b_1 = b_2 \\ \top & \text{otherwise} \end{cases} \quad \text{with } b_i = c_i(n)$$

The following Lua function dispatches lookups for key k from map receiver to the two maps m_1, m_2 in an ordered fashion with priority on m_1 :

```
function asymmetricDispatch (receiver, k)
  local v = receiver.m1[k]
  if not v then return receiver.m2[k] end
  return v
end
```

1. provide a Lua implementation of the function `symmetricDispatch`, which implements dispatching of key k based on the symmetric join \sqcup .
2. use this function to implement a function `composeTraits`, which takes a pair of

2 von 2 >

debian 1 2 3 4 exercise09 : okular — Konsole exercise.pdf - Okular 10:54:23

emacs@michaels-t420s

```
function symmetricDispatch(receiver, key)
end
```

Stream.class Synchron.class
Streams.java
Streams.java-

lay ":1".

lay ":1".

debian 1 2 3 4 tutorial : bash — Konsole emacs@michaels-t420s 10:54:46

exercise.pdf - Okular

Datei Bearbeiten Ansicht Gehe zu Lesezeichen Extras Einstellungen Hilfe

Seitenbreite

Trait composition $+$ is defined as a symmetric join \sqcup between two maps c_1, c_2 :

$$(c_1 + c_2)(n) = b_1 \sqcup b_2 = \begin{cases} b_2 & \text{if } b_1 = \perp \vee n \notin \text{pre}(c_1) \\ b_1 & \text{if } b_2 = \perp \vee n \notin \text{pre}(c_2) \\ b_2 & \text{if } b_1 = b_2 \\ \top & \text{otherwise} \end{cases} \quad \text{with } b_i = c_i(n)$$

The following Lua function dispatches lookups for key k from map receiver to the two maps m_1, m_2 in an ordered fashion with priority on m_1 :

```
function asymmetricDispatch (receiver, k)
  local v = receiver.m1[k]
  if not v then return receiver.m2[k] end
  return v
end
```

1. provide a Lua implementation of the function `symmetricDispatch`, which implements dispatching of key k based on the symmetric join \sqcup .

2 von 2 >

debian 1 2 3 4 exercise09 : okular — Konsole exercise.pdf - Okular 10:55:54

```
emacs@michaels-t420s
function symmetricDispatch(receiver, key)
  local v1 = receiver.m1[key]
  local v2 = receiver.m2[key]
  if not v1 t
end

Stream.class  Synch.class
Streams.java
Streams.java-

lay ":1".

lay ":1".
```

```
emacs@michaels-t420s
function symmetricDispatch(receiver, key)
  local v1 = receiver.m1[key]
  local v2 = receiver.m2[key]
  if not v1 then return v2 end
end

Stream.class  Synch.class
Streams.java
Streams.java-

lay ":1".

lay ":1".
```

```
emacs@michaels-t420s
function symmetricDispatch(receiver, key)
  local v1 = receiver.m1[key]
  local v2 = receiver.m2[key]
  if not v1 then return v2 end
  if not v2 then return v1 end
end

Stream.class  Synch.class
Streams.java
Streams.java-

lay ":1".

lay ":1".
```

```
emacs@michaels-t420s
function symmetricDispatch(receiver, key)
  local v1 = receiver.m1[key]
  local v2 = receiver.m2[key]
  if not v1 then return v2 end
  if not v2 then return v1 end
  if v1 == v2 then return v1 end
end

Stream.class  Synch.class
Streams.java
Streams.java-

lay ":1".

lay ":1".
```


exercise.pdf - Okular

Trait composition $+$ is defined as a symmetric join \sqcup between two maps c_1, c_2 :

$$(c_1 + c_2)(n) = b_1 \sqcup b_2 = \begin{cases} b_2 & \text{if } b_1 = \perp \vee n \notin \text{pre}(c_1) \\ b_1 & \text{if } b_2 = \perp \vee n \notin \text{pre}(c_2) \\ b_2 & \text{if } b_1 = b_2 \\ \top & \text{otherwise} \end{cases} \quad \text{with } b_i = c_i(n)$$

The following Lua function dispatches lookups for key k from map `receiver` to the two maps `m1, m2` in an ordered fashion with priority on `m1`:

```
function asymmetricDispatch (receiver, k)
  local v = receiver.m1[k]
  if not v then return receiver.m2[k] end
  return v
end
```

1. provide a Lua implementation of the function `symmetricDispatch`, which implements dispatching of key k based on the symmetric join \sqcup .

emacs@michaels-t420s

```
top = {}
function symmetricDispatch(receiver, key)
  local v1 = receiver.m1[key]
  local v2 = receiver.m2[key]
  if not v1 then return v2 end
  if not v2 then return v1 end
  if v1 == v2 then return v1 end
  return top
end
```

Stream.class Synch.class
Streams.java
Streams.java-

lay ":1".

lay ":1".

emacs@michaels-t420s

```
top = {}
bot = {}

function symmetricDispatch(receiver, key)
  local v1 = receiver.m1[key]
  local v2 = receiver.m2[key]
  if not v1 then return v2 end
  if not v2 then return v1 end
  if v1 == v2 then return v1 end
  return top
end
```

Stream.class Synch.class
Streams.java
Streams.java-

lay ":1".

lay ":1".

emacs@michaels-t420s

```
top = {}

function symmetricDispatch(receiver, key)
  local v1 = receiver.m1[key]
  local v2 = receiver.m2[key]
  if not v1 then return v2 end
  if not v2 then return v1 end
  if v1 == v2 then return v1 end
  return top
end
```

Stream.class Synch.class
Streams.java
Streams.java-

lay ":1".

lay ":1".

exercise.pdf - Okular

Trait composition $+$ is defined as a symmetric join \sqcup between two maps c_1, c_2 :

$$(c_1 + c_2)(n) = b_1 \sqcup b_2 = \begin{cases} b_2 & \text{if } b_1 = \perp \vee n \notin \text{pre}(c_1) \\ b_1 & \text{if } b_2 = \perp \vee n \notin \text{pre}(c_2) \\ b_2 & \text{if } b_1 = b_2 \\ \top & \text{otherwise} \end{cases} \quad \text{with } b_i = c_i(n)$$

The following Lua function dispatches lookups for key k from map **receiver** to the two maps m_1, m_2 in an ordered fashion with priority on m_1 :

```
function asymmetricDispatch (receiver, k)
  local v = receiver.m1[k]
  if not v then return receiver.m2[k] end
  return v
end
```

- provide a Lua implementation of the function `symmetricDispatch`, which implements dispatching of key k based on the symmetric join \sqcup .

2 von 2 >

exercise.pdf - Okular

$$(c_1 + c_2)(n) = b_1 \sqcup b_2 = \begin{cases} b_2 & \text{if } b_1 = \perp \vee n \notin \text{pre}(c_1) \\ b_1 & \text{if } b_2 = \perp \vee n \notin \text{pre}(c_2) \\ b_2 & \text{if } b_1 = b_2 \\ \top & \text{otherwise} \end{cases} \quad \text{with } b_i = c_i(n)$$

The following Lua function dispatches lookups for key k from map **receiver** to the two maps m_1, m_2 in an ordered fashion with priority on m_1 :

```
function asymmetricDispatch (receiver, k)
  local v = receiver.m1[k]
  if not v then return receiver.m2[k] end
  return v
end
```

- provide a Lua implementation of the function `symmetricDispatch`, which implements dispatching of key k based on the symmetric join \sqcup .
- use this function to implement a function `composeTraits`, which takes a pair of trait maps as input and creates an object-like map as output, that delegates its

2 von 2 >

exercise.pdf - Okular

```
function asymmetricDispatch (receiver, k)
  local v = receiver.m1[k]
  if not v then return receiver.m2[k] end
  return v
end
```

- provide a Lua implementation of the function `symmetricDispatch`, which implements dispatching of key k based on the symmetric join \sqcup .
- use this function to implement a function `composeTraits`, which takes a pair of trait maps as input and creates an object-like map as output, that delegates its lookups to the traits in symmetric join fashion.

Assignment 9.3 Prototype Based Design

Plan and implement the datastructures to represent symbolical arithmetical expressions, composed of the operators $+$, $-$, $*$, $/$, constants and variables in Lua. Don't forget to include

2 von 2 >

emacs@michaels-t420s

```
top = {}

function symmetricDispatch(receiver, key)
  local v1 = receiver.m1[key]
  local v2 = receiver.m2[key]
  if not v1 then return v2 end
  if not v2 then return v1 end
  if v1 == v2 then return v1 end
  return top
end
```

```
Stream.class   Synch.class
Streams.java
Streams.java-
```

```
lay ":1".
lay ":1".
```

1 2 3 4 tutorial: bash - Konsole emacs@michaels-t420s 11:06:12

exercise.pdf - Okular

Seitenbreite

otherwise

The following Lua function dispatches lookups for key `k` from map `receiver` to the two maps `m1, m2` in an ordered fashion with priority on `m1`:

```
function asymmetricDispatch (receiver, k)
  local v = receiver.m1[k]
  if not v then return receiver.m2[k] end
  return v
end
```

1. provide a Lua implementation of the function `symmetricDispatch`, which implements dispatching of key `k` based on the symmetric join \sqcup .
2. use this function to implement a function `composeTraits`, which takes a pair of trait maps as input and creates an object-like map as output, that delegates its lookups to the traits in symmetric join fashion.

Arbeitsbereich: 2

1 2 3 4 exercise09 : okular — Konsole exercise.pdf - Okular 11:15:09

exercise.pdf - Okular

Seitenbreite

Assignment 9.3 Prototype Based Design

Plan and implement the datastructures to represent symbolical arithmetical expressions, composed of the operators `+`, `-`, `*`, `/`, constants and variables in Lua. Don't forget to include nice ways to specify and evaluate them!

Assignment 9.4 Optional at home: Multiple Inheritance

In the lecture, we established a function `createClass()`, to establish a Dual-Inheritance-Relation for an arbitrary new table. Extend this function, to obtain true n-ary multiple-inheritance.

- go to the Lua documentation and research vararg-parameters
- reimplement the `createClass`-function as vararg

1 2 3 4 exercise09 : okular — Konsole exercise.pdf - Okular 11:17:54

emacs@michaels-t420s

```
top = {}

function symmetricDispatch(receiver, key)
  local v1 = receiver.m1[key]
  local v2 = receiver.m2[key]
  if not v1 then return v2 end
  if not v2 then return v1 end
  if v1 == v2 then return v1 end
  return top
end

function composeTraits ( trait1, trait2 )
  o = { m1=trait1, m2=trait2 }
  setmetatable(o, {
    __index=symmetricDispatch
  })
  return o
end
```

```
Stream.class  Synchron.class
Streams.java
Streams.java-

lay ":1".

lay ":1".
```

1 2 3 4 tutorial : bash — Konsole emacs@michaels-t420s 11:20:24

emacs@michaels-t420s

```
U: --- expressions.lua All (1,0) (Fundamental company)
(New file)
```

1 2 3 4 tutorial : bash — Konsole emacs@michaels-t420s 11:20:35

```
emacs@michaels-t420s
Expr = {}
Expr.__index=Expr

exp = const(5) + const(42)*var("x")

print(exp:eval({x=4711}))
print(exp:eval({x=42}))

U:**- expressions.lua All (2,0) (Fundamental company)
```

```
emacs@michaels-t420s
Expr = {}
Expr.__index=Expr

exp = const(5) + const(42)*var("x")

print(exp:eval({x=4711}))
print(exp:eval({x=42}))

U:**- expressions.lua All (3,0) (Fundamental company)
```

debian | 1 2 3 4 | tutorial : bash — Konsole | emacs@michaels-t420s | 11:29:33

debian | 1 2 3 4 | tutorial : bash — Konsole | emacs@michaels-t420s | 11:29:46

```
emacs@michaels-t420s
Expr = {}
Expr.__index=Expr

function Expr:new(o){
    setmetatable(o,self)
}

exp = const(5) + const(42)*var("x")

print(exp:eval({x=4711}))
print(exp:eval({x=42}))

U:**- expressions.lua All (6,10) (Fundamental company)
```

```
emacs@michaels-t420s
Expr = {}
Expr.__index=Expr

function Expr:new(o)
    setmetatable(o,self)
    return o
end

exp = const(5) + const(42)*var("x")

print(exp:eval({x=4711}))
print(exp:eval({x=42}))

U:**- expressions.lua All (9,0) (Fundamental company)
```

debian | 1 2 3 4 | tutorial : bash — Konsole | emacs@michaels-t420s | 11:30:46

debian | 1 2 3 4 | tutorial : bash — Konsole | emacs@michaels-t420s | 11:31:56

```
emacs@microhals-t420s
Expr = {}
Expr.__index=Expr

function Expr:new(o)
    setmetatable(o,self)
    return o
end

function Expr.__add(a,b)
end

exp = const(5) + const(42)*var("x")

print(exp:eval({x=4711}))
print(exp:eval({x=42}))

U:**- expressions.lua All (10,0) (Fundamental company)
```

```
emacs@microhals-t420s
Expr = {}
Expr.__index=Expr

function Expr:new(o)
    setmetatable(o,self)
    return o
end

function Expr.__add(a,b)
    return {}
end

exp = const(5) + const(42)*var("x")

print(exp:eval({x=4711}))
print(exp:eval({x=42}))

U:**- expressions.lua All (11,9) (Fundamental company)
```

debiun | 1 2 3 4 | tutorial : bash — Konsole | emacs@microhals-t420s | 11:32:19

debiun | 1 2 3 4 | tutorial : bash — Konsole | emacs@microhals-t420s | 11:32:46

```
emacs@microhals-t420s
Expr = {}
Expr.__index=Expr

function Expr:new(o)
    setmetatable(o,self)
    return o
end

function Expr.__add(a,b)
    return Expr:new({left=a,right=b,operand=function (a,b) return a+b end})
end

exp = const(5) + const(42)*var("x")

print(exp:eval({x=4711}))
print(exp:eval({x=42}))

U:**- expressions.lua All (13,0) (Fundamental company)
```

```
emacs@microhals-t420s
Expr = {}
Expr.__index=Expr

function Expr:new(o)
    setmetatable(o,self)
    return o
end

function Expr.__add(a,b)
    return Expr:new({left=a,right=b,operator=function (a,b) return a+b end})
end

function Expr.__mul(a,b)
    return Expr:new({left=a,right=b,operator=function (a,b) return a*b end})
end

exp = const(5) + const(42)*var("x")

print(exp:eval({x=4711}))
print(exp:eval({x=42}))

U:**- expressions.lua All (14,0) (Fundamental company)
Mark set
```

debiun | 1 2 3 4 | tutorial : bash — Konsole | emacs@microhals-t420s | 11:35:22

debiun | 1 2 3 4 | tutorial : bash — Konsole | emacs@microhals-t420s | 11:36:48

```
emacs@microhals-t420s
Expr = {}
Expr.__index=Expr

function Expr:new(o)
    setmetatable(o,self)
    return o
end

function Expr.__add(a,b)
    return Expr:new({left=a,right=b,operator=function (a,b) return a+b end})
end
function Expr.__mul(a,b)
    return Expr:new({left=a,right=b,operator=function (a,b) return a*b end})
end

exp = const(5) + const(42)*var("x")

print(exp:eval({x=4711}))
print(exp:eval({x=42}))

U:**- expressions.lua All (9,0) (Fundamental company)
```

```
emacs@microhals-t420s
Expr = {}
Expr.__index=Expr

function Expr:new(o)
    setmetatable(o,self)
    return o
end

function const(i)
    _
end

function Expr.__add(a,b)
    return Expr:new({left=a,right=b,operator=function (a,b) return a+b end})
end
function Expr.__mul(a,b)
    return Expr:new({left=a,right=b,operator=function (a,b) return a*b end})
end

exp = const(5) + const(42)*var("x")

print(exp:eval({x=4711}))
print(exp:eval({x=42}))

U:**- expressions.lua All (10,0) (Fundamental company)
```

```
emacs@microhals-t420s
Expr = {}
Expr.__index=Expr

function Expr:new(o)
    setmetatable(o,self)
    return o
end

function const(i)
    return Expr:new({ const=i })
end

function Expr.__add(a,b)
    return Expr:new({left=a,right=b,operator=function (a,b) return a+b end})
end
function Expr.__mul(a,b)
    return Expr:new({left=a,right=b,operator=function (a,b) return a*b end})
end

exp = const(5) + const(42)*var("x")

print(exp:eval({x=4711}))
print(exp:eval({x=42}))

U:**- expressions.lua All (13,0) (Fundamental company)
```

```
emacs@microhals-t420s
Expr = {}
Expr.__index=Expr

function Expr:new(o)
    setmetatable(o,self)
    return o
end

function const(i)
    return Expr:new({ const=i })
end

function var(s)
    _
end

function Expr.__add(a,b)
    return Expr:new({left=a,right=b,operator=function (a,b) return a+b end})
end
function Expr.__mul(a,b)
    return Expr:new({left=a,right=b,operator=function (a,b) return a*b end})
end

exp = const(5) + const(42)*var("x")

print(exp:eval({x=4711}))
print(exp:eval({x=42}))

U:**- expressions.lua All (14,0) (Fundamental company)
```

```
emacs@microhals-t420s
end

function Expr.__add(a,b)
    return Expr:new({left=a,right=b,operator=function (a,b) return a+b end})
end
function Expr.__mul(a,b)
    return Expr:new({left=a,right=b,operator=function (a,b) return a*b end})
end

exp = const(5) + const(42)*var("x")

print(exp:eval({x=4711}))
print(exp:eval({x=42}))

U:**- expressions.lua Bot (27,15) (Fundamental company)
```

```
emacs@microhals-t420s
end
function Expr.__add(a,b)
    return Expr:new({left=a,right=b,operator=function (a,b) return a+b end})
end
function Expr.__mul(a,b)
    return Expr:new({left=a,right=b,operator=function (a,b) return a*b end})
end

exp = const(5) + const(42)*var("x")

print(exp:eval({x=4711}))
print(exp:eval({x=42}))

U:**- expressions.lua Bot (24,0) (Fundamental company)
```

```
emacs@microhals-t420s
end
function Expr.__add(a,b)
    return Expr:new({left=a,right=b,operator=function (a,b) return a+b end})
end
function Expr.__mul(a,b)
    return Expr:new({left=a,right=b,operator=function (a,b) return a*b end})
end

function Expr:eval(env)
end

exp = const(5) + const(42)*var("x")

print(exp:eval({x=4711}))
print(exp:eval({x=42}))

U:**- expressions.lua Bot (25,0) (Fundamental company)
```

```
emacs@microhals-t420s
end
function Expr.__add(a,b)
    return Expr:new({left=a,right=b,operator=function (a,b) return a+b end})
end
function Expr.__mul(a,b)
    return Expr:new({left=a,right=b,operator=function (a,b) return a*b end})
end

function Expr:eval(env)
end

exp = const(5) + const(42)*var("x")

print(exp:eval({x=4711}))
print(exp:eval({x=42}))

U:**- expressions.lua Bot (25,0) (Fundamental company)
```

```
emacs@michaels-t420s
Expr = {}
Expr.__index=Expr

function Expr:new(o)
    setmetatable(o,self)
    return o
end

function const(i)
    return Expr:new({ const=i })
end

function var(s)
    return Expr:new({ varname=s})
end

function Expr.__add(a,b)
    return Expr:new({left=a,right=b,operator=function (a,b) return a+b end})
end

function Expr.__mul(a,b)
    return Expr:new({left=a,right=b,operator=function (a,b) return a*b end})
end

function Expr:eval(env)
    if self.const
end

U:**- expressions.lua Top (12,0) (Fundamental company)
```

```
emacs@michaels-t420s
end

function Expr.__add(a,b)
    return Expr:new({left=a,right=b,operator=function (a,b) return a+b end})
end

function Expr.__mul(a,b)
    return Expr:new({left=a,right=b,operator=function (a,b) return a*b end})
end

function Expr:eval(env)
    if self.const then return self.const end
    if self.varname then return env[
end_

exp = const(5) + const(42)*var("x")

print(exp:eval({x=4711}))
print(exp:eval({x=42}))

U:**- expressions.lua Bot (28,0) (Fundamental company)
```

debain | 1 2 3 4 | tutorial: bash — Konsole | emacs@michaels-t420s | 11:44:00

debain | 1 2 3 4 | tutorial: bash — Konsole | emacs@michaels-t420s | 11:44:56

```
emacs@michaels-t420s
end

function Expr.__add(a,b)
    return Expr:new({left=a,right=b,operator=function (a,b) return a+b end})
end

function Expr.__mul(a,b)
    return Expr:new({left=a,right=b,operator=function (a,b) return a*b end})
end

function Expr:eval(env)
    if self.const then return self.const end
    if self.varname then return env[self.varname] end
end

exp = const(5) + const(42)*var("x")

print(exp:eval({x=4711}))
print(exp:eval({x=42}))

U:**- expressions.lua Bot (27,2) (Fundamental company)
```

```
emacs@michaels-t420s
Expr = {}
Expr.__index=Expr

function Expr:new(o)
    setmetatable(o,self)
    return o
end

function const(i)
    return Expr:new({ const=i })
end

function var(s)
    return Expr:new({ varname=s})
end

function Expr.__add(a,b)
    return Expr:new({left=a,right=b,operator=function (a,b) return a+b end})
end

function Expr.__mul(a,b)
    return Expr:new({left=a,right=b,operator=function (a,b) return a*b end})
end

function Expr:eval(env)
    if self.const then return self.const end
    if self.varname then return env[self.varname] end
    return self.operator(self.left:eval(env),self.right:eval(env))
end

U:**- expressions.lua Top (11,3) (Fundamental company)
```

debain | 1 2 3 4 | tutorial: bash — Konsole | emacs@michaels-t420s | 11:45:14

debain | 1 2 3 4 | tutorial: bash — Konsole | emacs@michaels-t420s | 11:48:48


```
emacs@microhals-t420s
end

function Expr.__add(a,b)
    return Expr:new({left=a,right=b,operator=function (a,b) return a+b end})
end
function Expr.__mul(a,b)
    return Expr:new({left=a,right=b,operator=function (a,b) return a*b end})
end

function Expr:eval(env)
    if self.const then return self.const end
    if self.varname then return env[self.varname] end
    return self.operator(self.left:eval(env),self.right:eval(env))
end

exp = const(5) + const(42)*var("x")

print(exp:eval({x=4711}))
print(exp:eval({x=42}))

U:**- expressions.lua Bot (33,23) (Fundamental company)
End of buffer
```

```
emacs@microhals-t420s
Expr = {}
Expr.__index=Expr

function Expr:new(o)
    setmetatable(o,self)
    return o
end

function const(i)
    return Expr:new({ const=i })
end

function var(s)
    return Expr:new({ varname=s})
end

function Expr.__add(a,b)
    return Expr:new({left=a,right=b,operator=function (a,b) return a+b end})
end
function Expr.__mul(a,b)
    return Expr:new({left=a,right=b,operator=function (a,b) return a*b end})
end

function Expr:eval(env)
    if self.const then return self.const end
    if self.varname then return env[self.varname] end
    return self.operator(self.left:eval(env),self.right:eval(env))
end

U:**- expressions.lua Top (12,0) (Fundamental company)
```

```
emacs@microhals-t420s
end

function Expr.__add(a,b)
    return Expr:new({left=a,right=b,operator=function (a,b) return a+b end})
end
function Expr.__mul(a,b)
    return Expr:new({left=a,right=b,operator=function (a,b) return a*b end})
end

function Expr:eval(env)
    if self.const then return self.const end
    if self.varname then return env[self.varname] end
    return self.operator(self.left:eval(env),self.right:eval(env))
end

exp = const(5) + const(42)*var("x")

print(exp:eval({x=4711}))
print(exp:eval({x=42}))

U:**- expressions.lua Bot (33,23) (Fundamental company)
End of buffer
```

```
emacs@microhals-t420s
Expr = {}
Expr.__index=Expr

function Expr:new(o)
    setmetatable(o,self)
    return o
end

function const(i)
    return Expr:new({ const=i })
end

function var(s)
    return Expr:new({ varname=s})
end

function Expr.__add(a,b)
    return Expr:new({left=a,right=b,operator=function (a,b) return a+b end})
end
function Expr.__mul(a,b)
    return Expr:new({left=a,right=b,operator=function (a,b) return a*b end})
end

function Expr:eval(env)
    if self.const then return self.const end
    if self.varname then return env[self.varname] end
    return self.operator(self.left:eval(env),self.right:eval(env))
end

U:**- expressions.lua Top (11,3) (Fundamental company)
```

```
emacs@michaels-t420s
end

function Expr.__add(a,b)
    return Expr:new({left=a,right=b,operator=function (a,b) return a+b end})
end
function Expr.__mul(a,b)
    return Expr:new({left=a,right=b,operator=function (a,b) return a*b end})
end

function Expr:eval(env)
    if self.const then return self.const end
    if self.varname then return env[self.varname] end
    return self.operator(self.left:eval(env),self.right:eval(env))
end

exp = const(5) + const(42)*var("x")

print(exp:eval({x=4711}))
print(exp:eval({x=42}))

U:**- expressions.lua Bot (33,23) (Fundamental company)
End of buffer
```