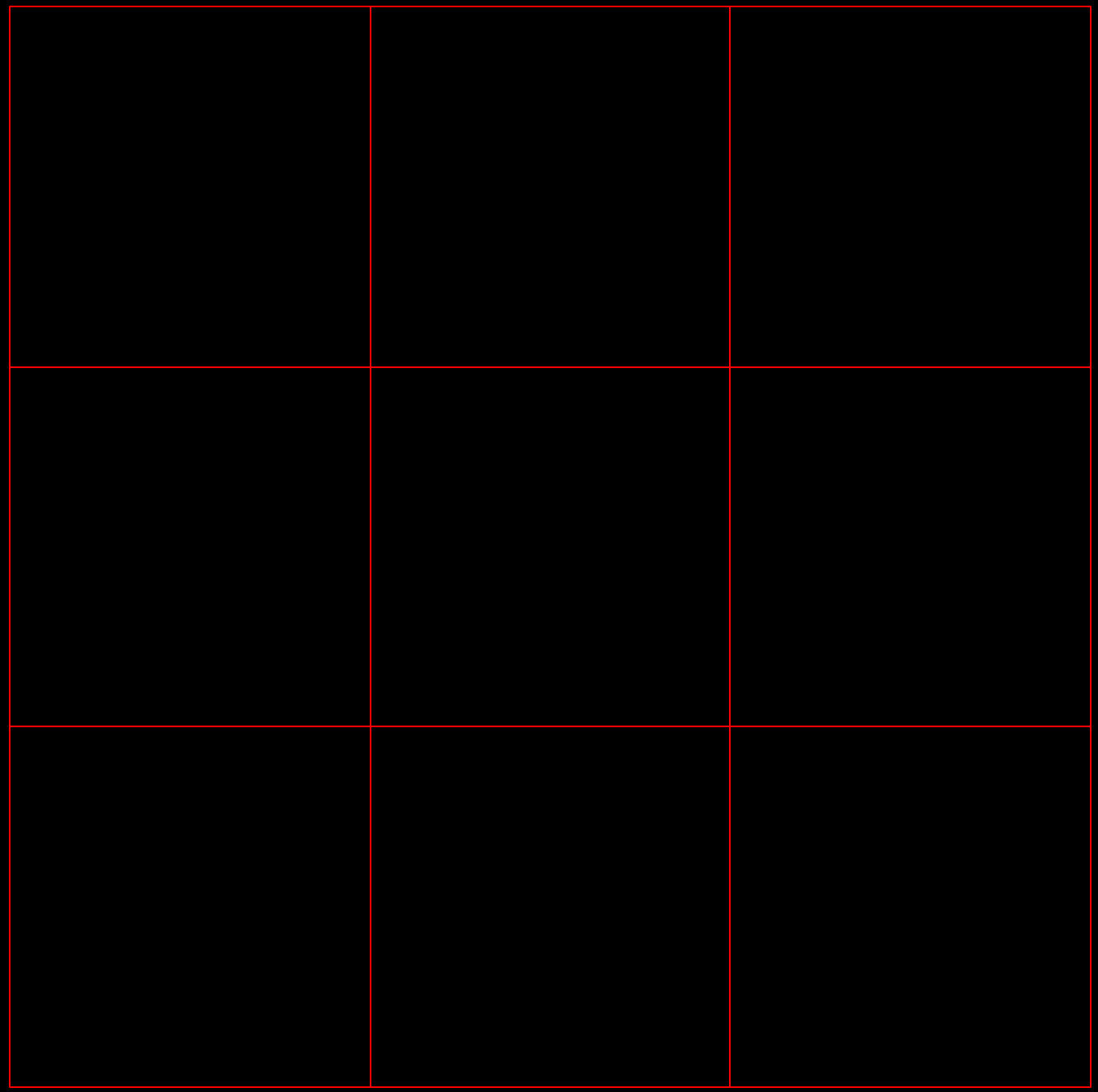
Easier encapsulation with static types

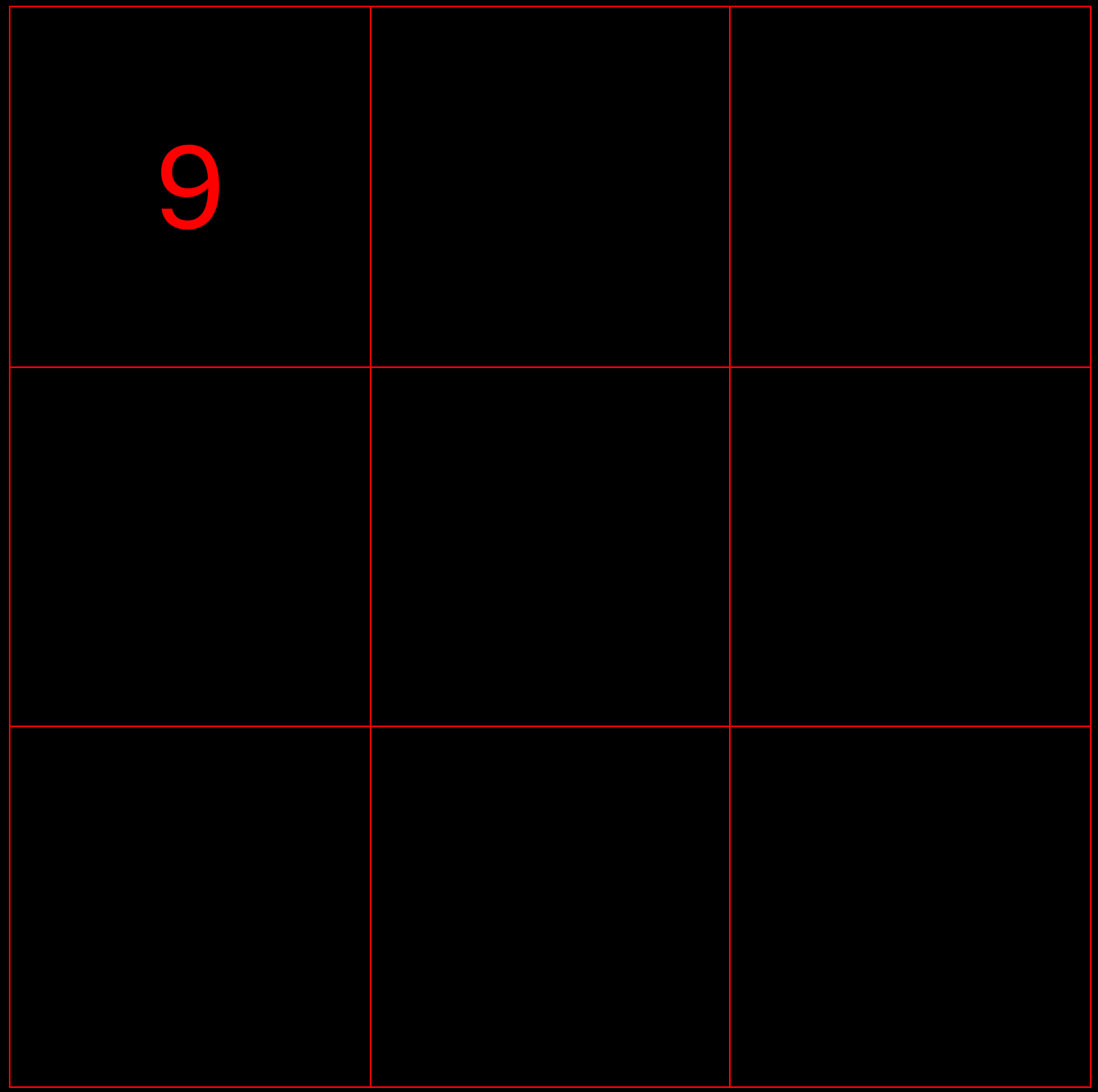
Mark Seemann

https://blog.ploeh.dk



https://blog.ploeh.dk

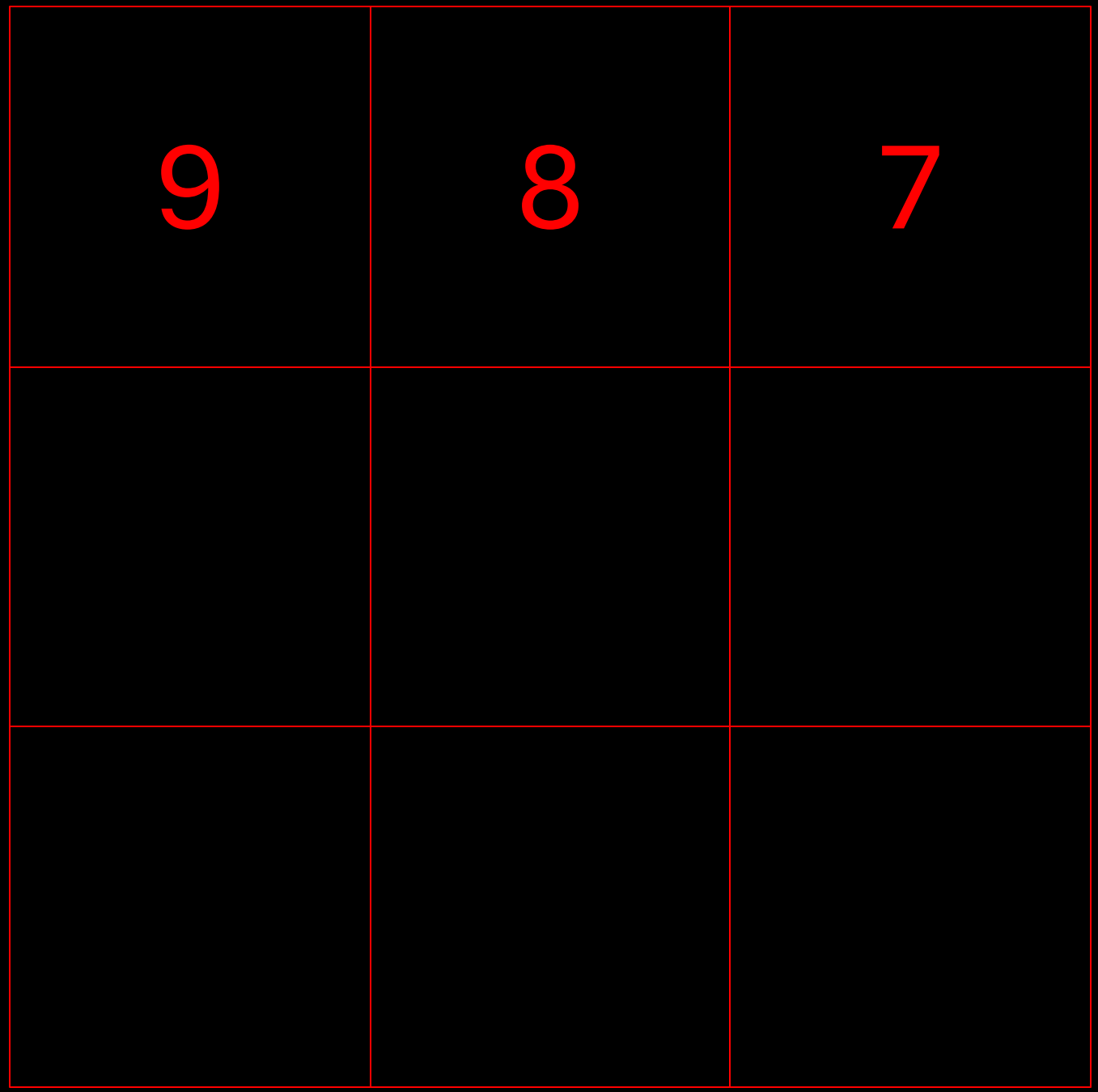



9



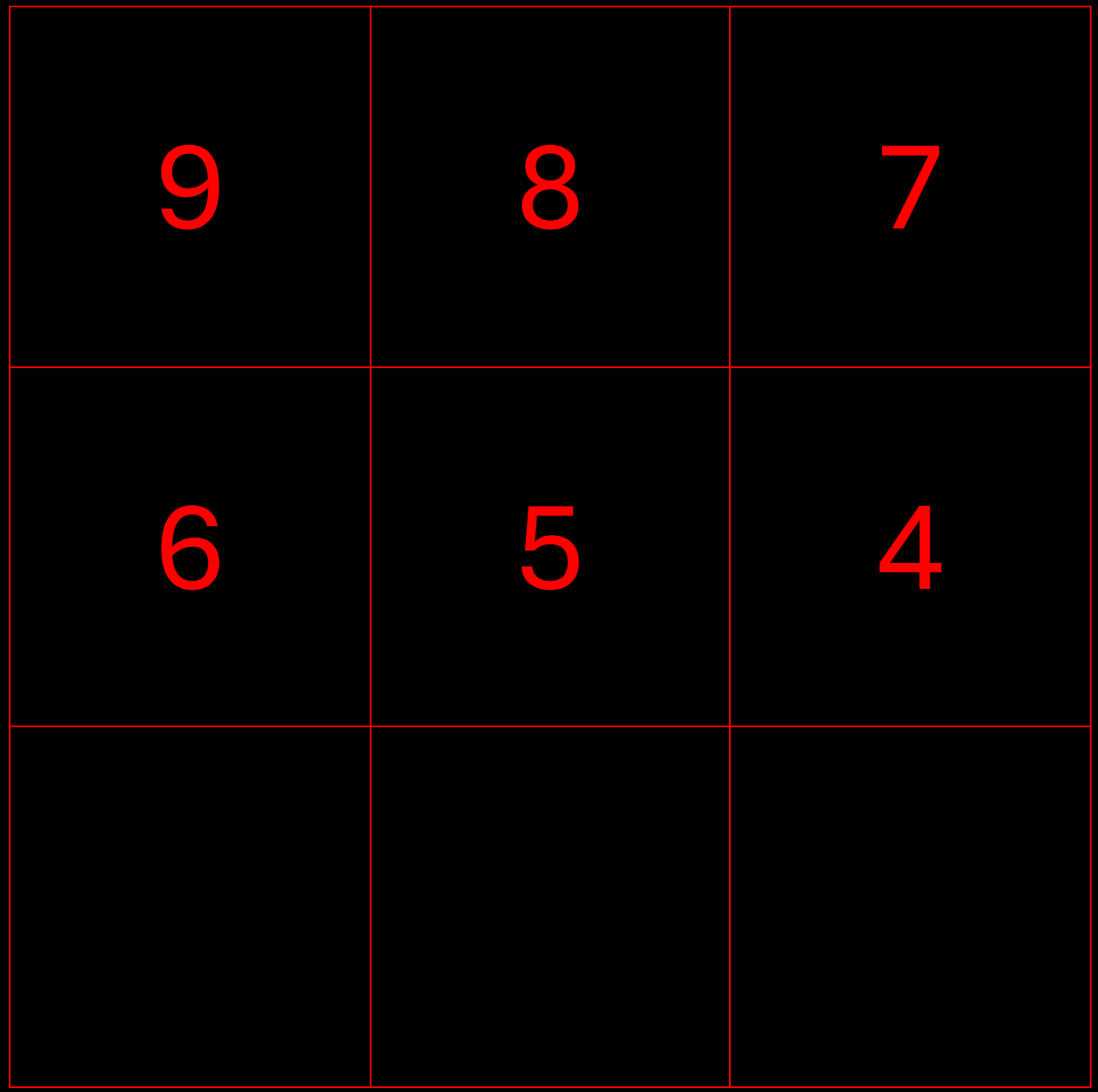
https://blog.ploeh.dk

9	8	


9
8
7

9	8	7
6		

9	8	7
6	5	


9
8
7
6
5
4

9	8	7
6	5	4
3		

9	8	7
6	5	4
3	2	


9
8
7
6
5
4
3
2
1

9·8·7·6·5·4·3·2·1

$$9 \cdot 8 \cdot 7 \cdot 6 \cdot 5 \cdot 4 \cdot 3 \cdot 2 \cdot 1$$

$$=$$

$$9!$$

$$9 \cdot 8 \cdot 7 \cdot 6 \cdot 5 \cdot 4 \cdot 3 \cdot 2 \cdot 1$$

$$=$$

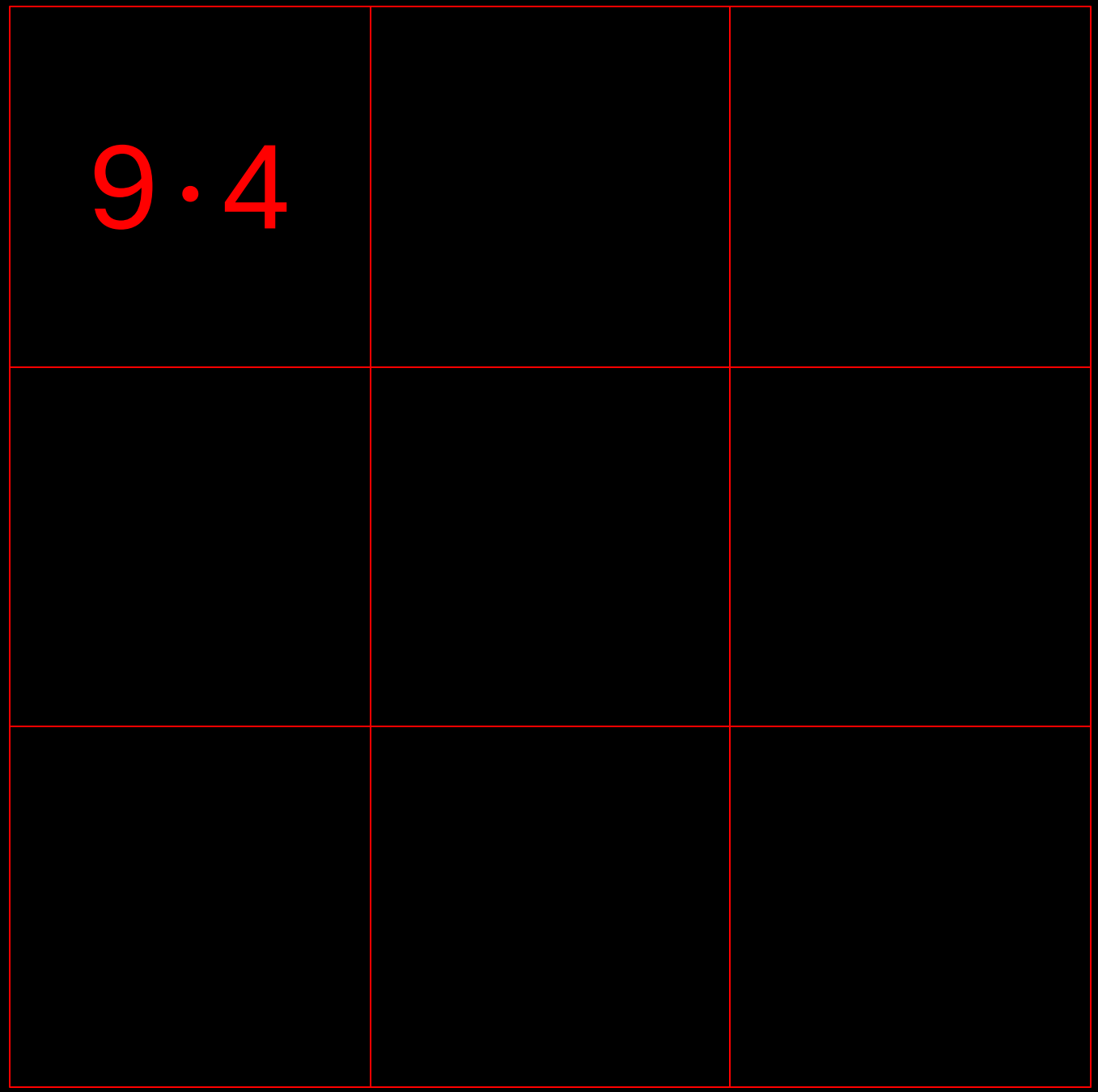
$$9!$$

$$=$$

$$362{,}880$$



https://blog.ploeh.dk


9 · 4

$9 \cdot 4$	$8 \cdot 4$	

$9 \cdot 4$	$8 \cdot 4$	$7 \cdot 4$

$9 \cdot 4$	$8 \cdot 4$	$7 \cdot 4$
$6 \cdot 4$		

$9 \cdot 4$	$8 \cdot 4$	$7 \cdot 4$
$6 \cdot 4$	$5 \cdot 4$	

$9 \cdot 4$	$8 \cdot 4$	$7 \cdot 4$
$6 \cdot 4$	$5 \cdot 4$	$4 \cdot 4$

https://blog.ploeh.dk

$9 \cdot 4$	$8 \cdot 4$	$7 \cdot 4$
$6 \cdot 4$	$5 \cdot 4$	$4 \cdot 4$
$3 \cdot 4$		

https://blog.ploeh.dk

$9 \cdot 4$	$8 \cdot 4$	$7 \cdot 4$
$6 \cdot 4$	$5 \cdot 4$	$4 \cdot 4$
$3 \cdot 4$	$2 \cdot 4$	

$9 \cdot 4$	$8 \cdot 4$	$7 \cdot 4$
$6 \cdot 4$	$5 \cdot 4$	$4 \cdot 4$
$3 \cdot 4$	$2 \cdot 4$	$1 \cdot 4$

https://blog.ploeh.dk

9·4·8·4·7·4·6·4·5·4·4·4·3·4·2·4·1·4

4 · 4 · 4 · 4 · 4 · 4 · 4 · 4 · 4 · 9 · 8 · 7 · 6 · 5 · 4 · 3 · 2 · 1

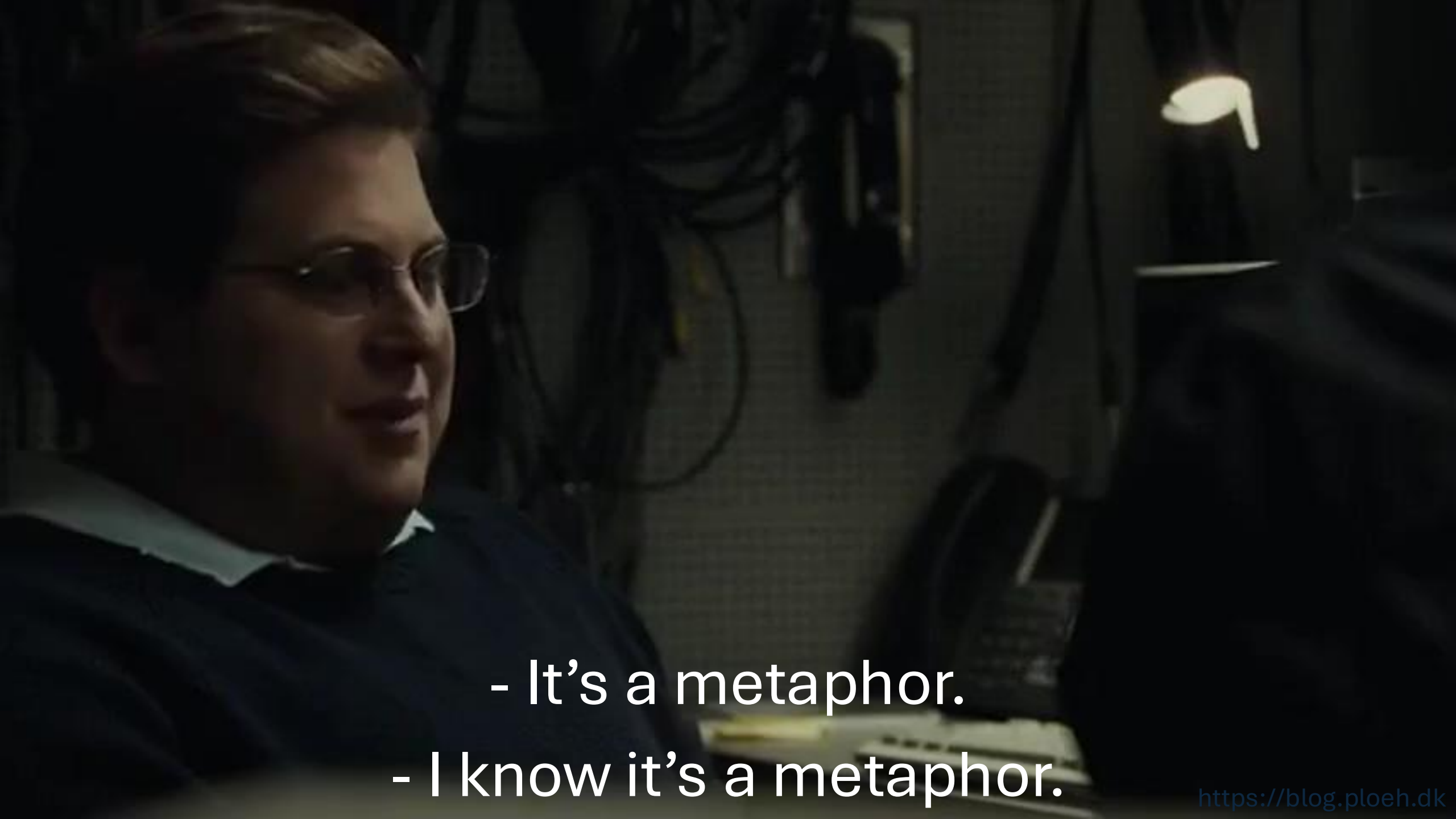
$4^{9}9!$

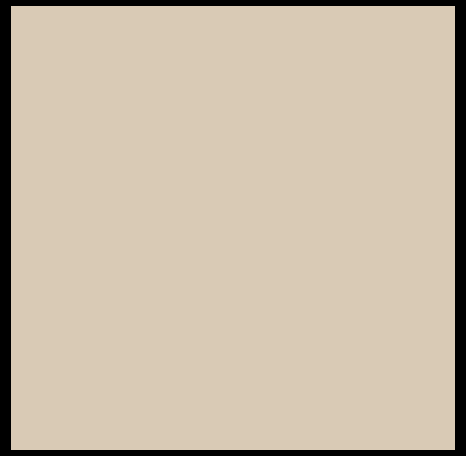
$$4^{9}9!$$

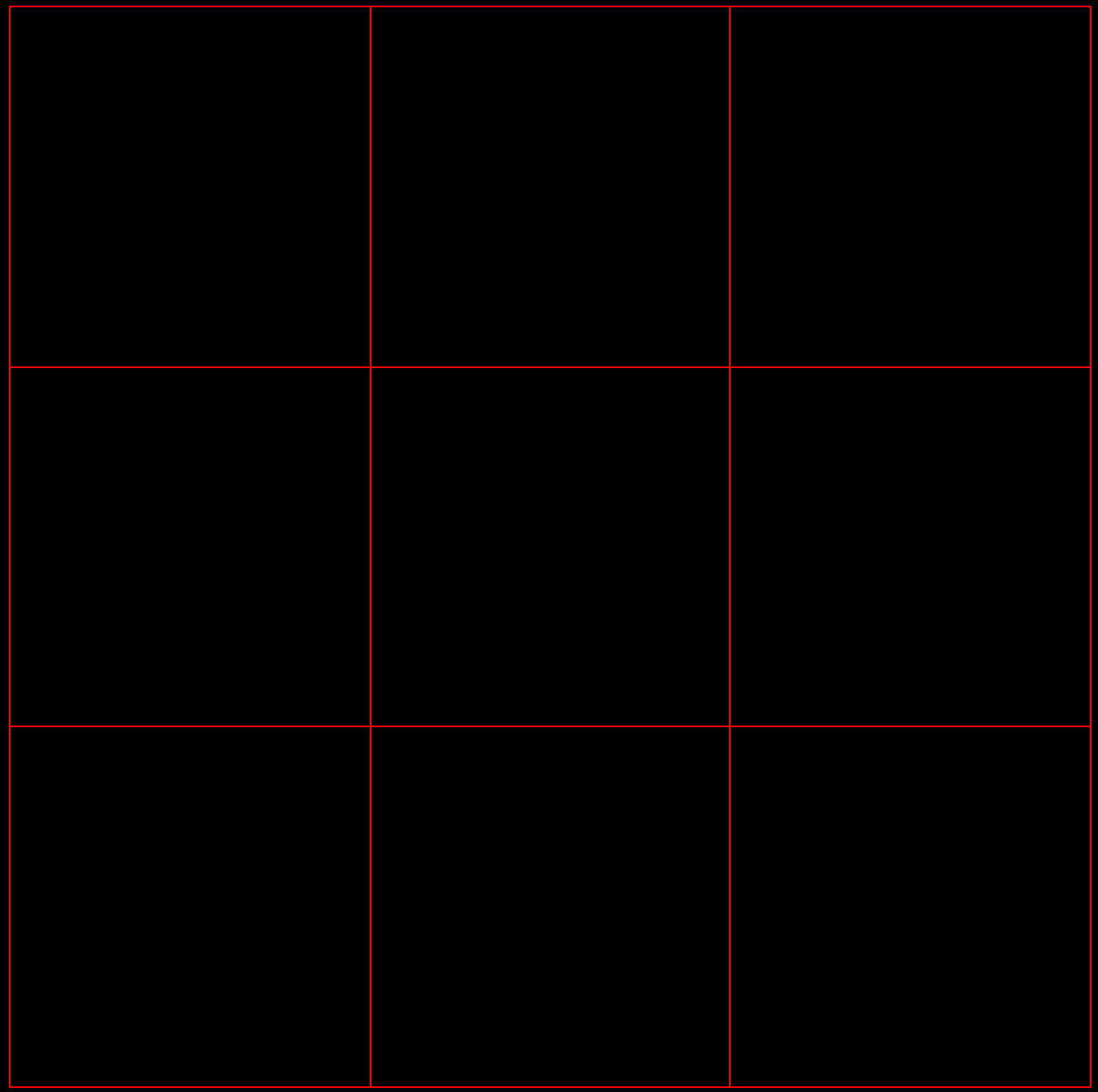
$$=$$

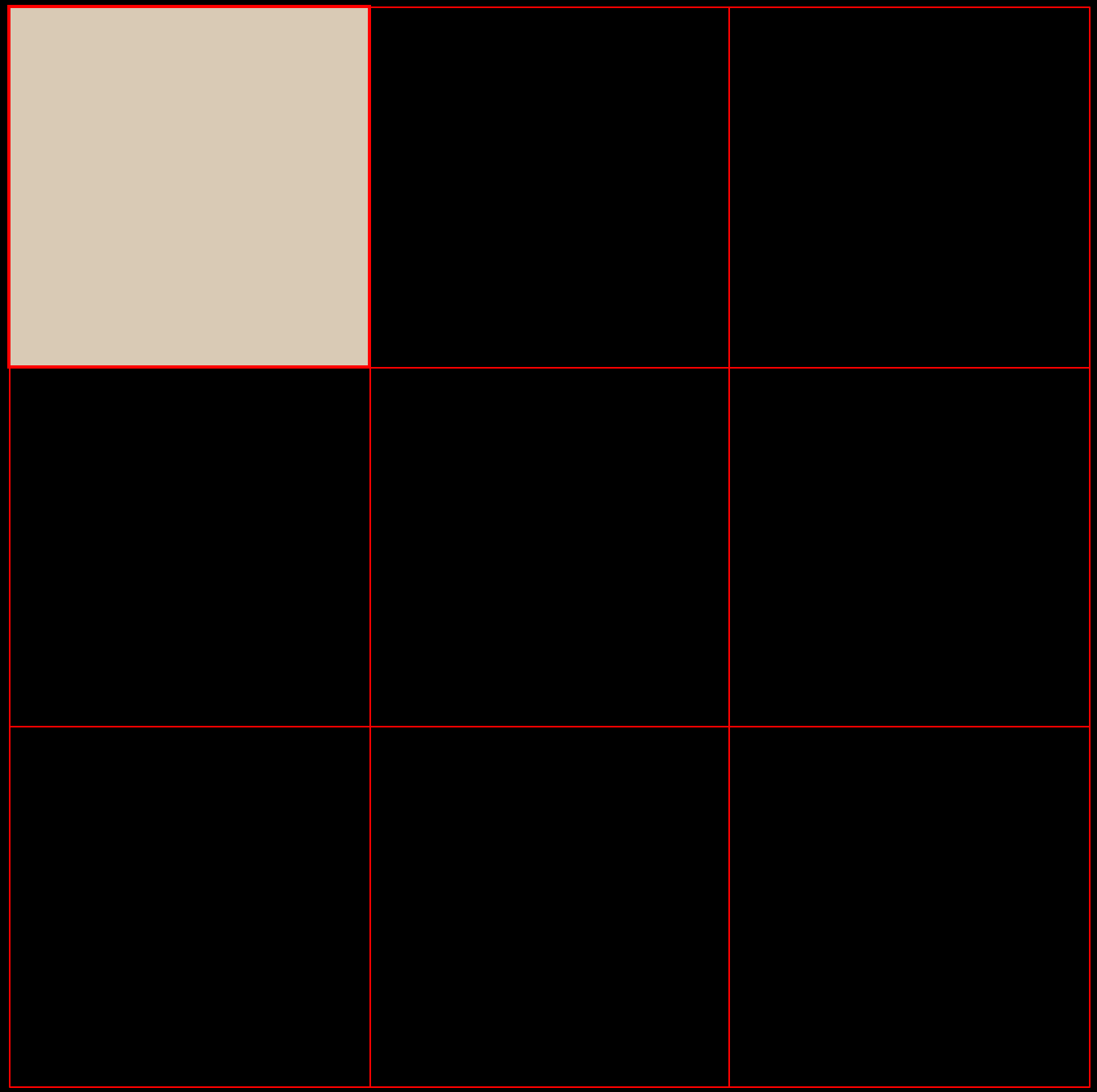
$$95,126,814,720$$

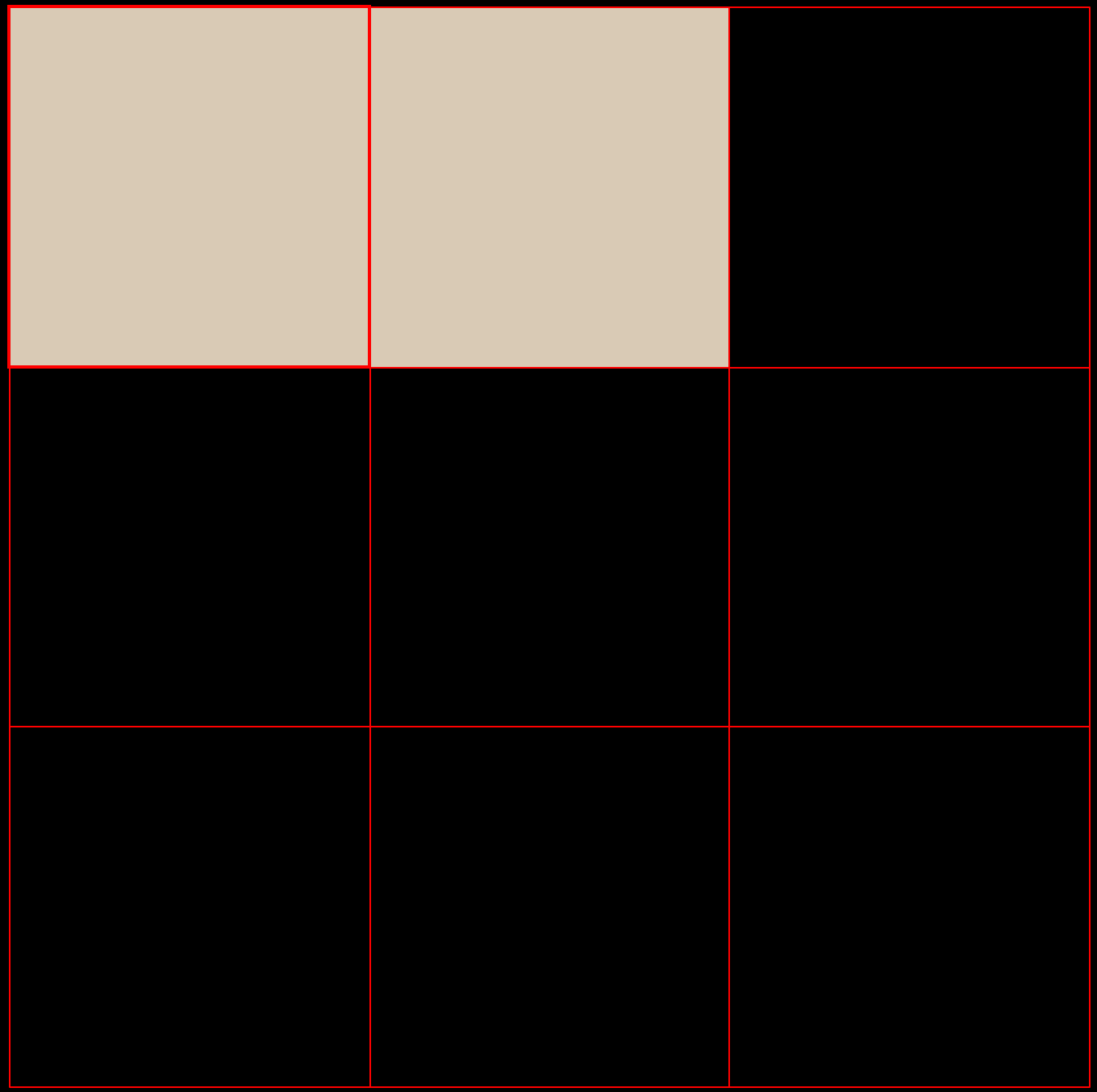
From now on,
let’s assume that we know the solution
We just need to find the right tiles
and compose them

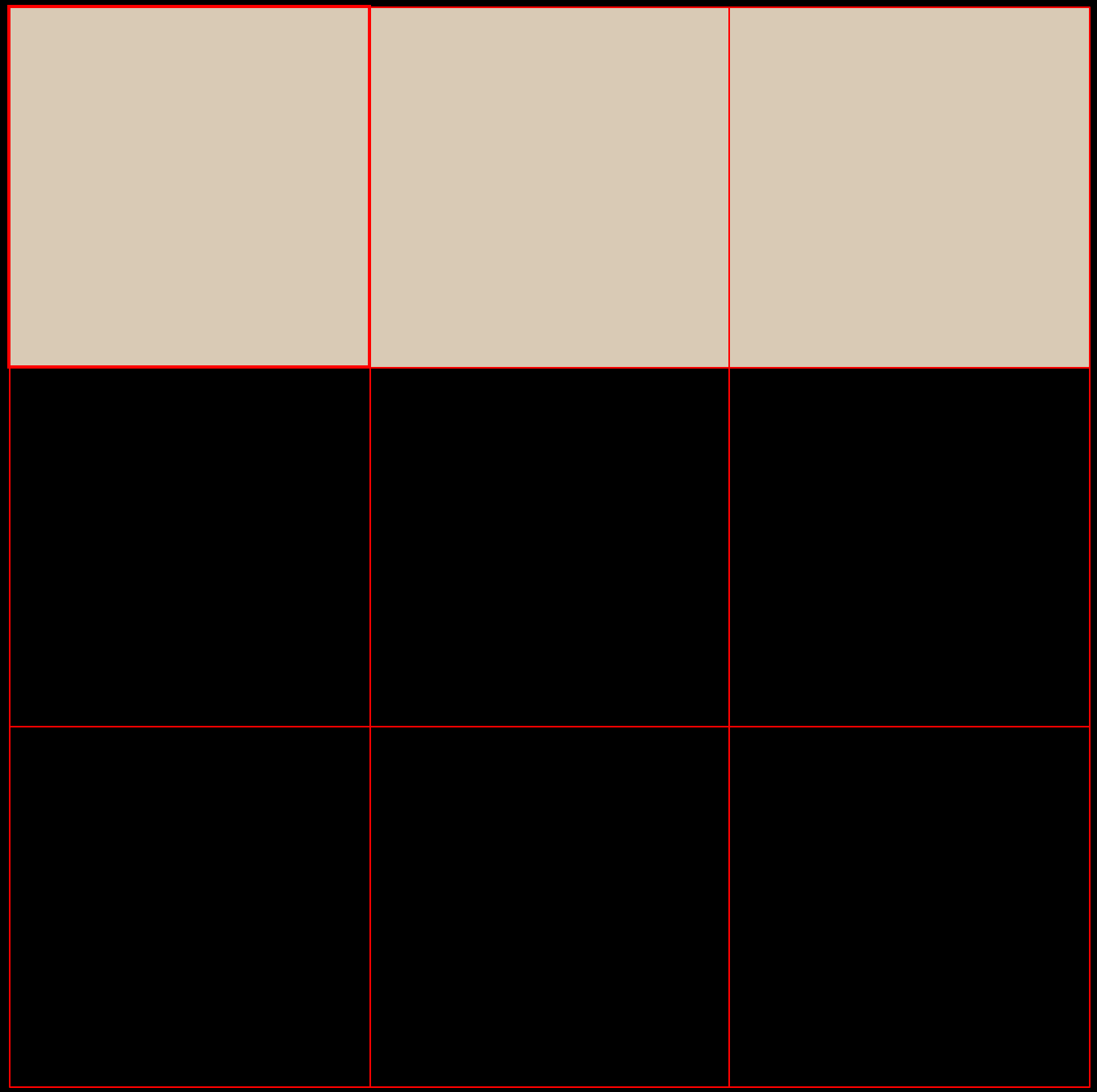

- It’s a metaphor.
- I know it’s a metaphor.

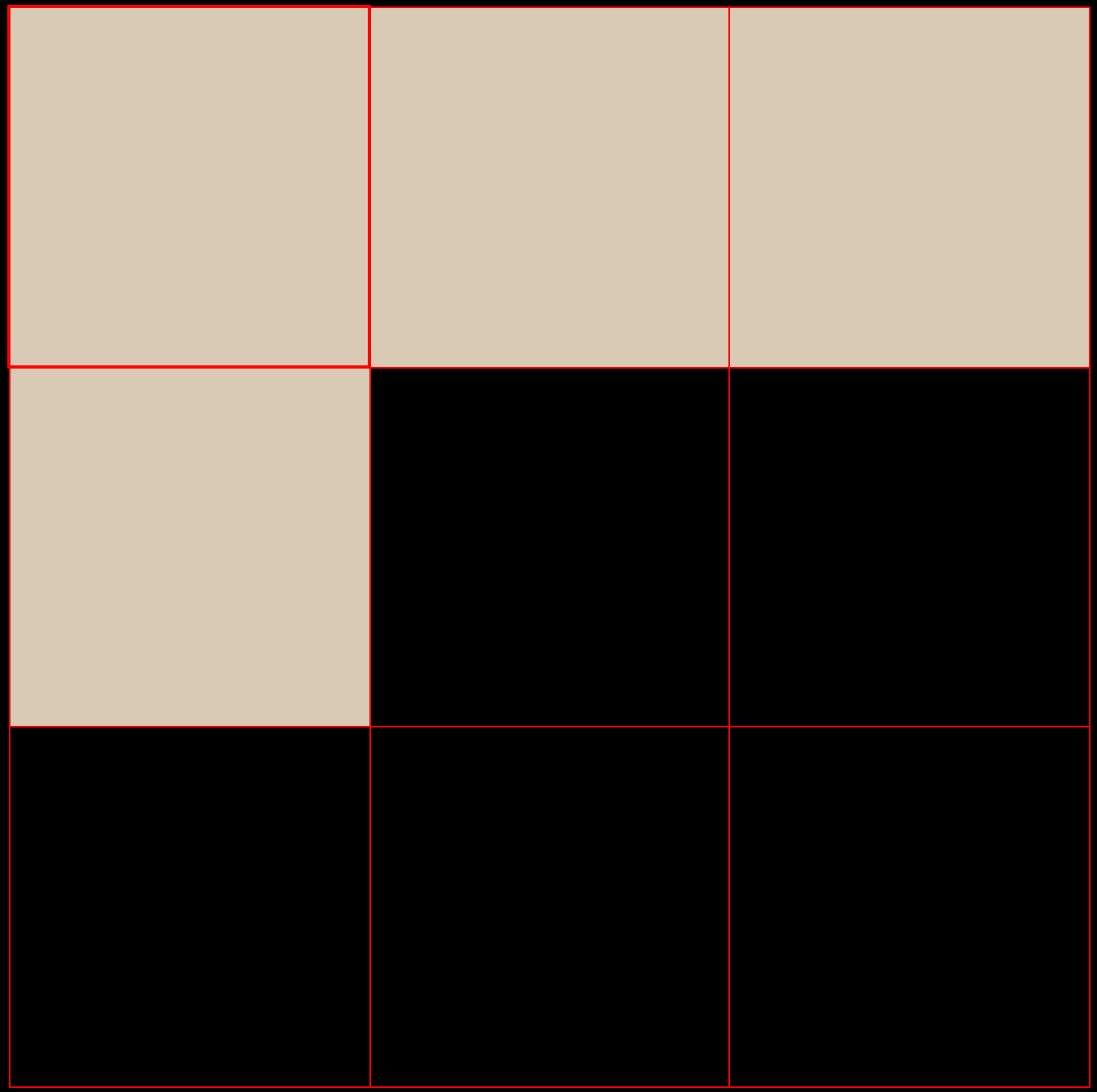


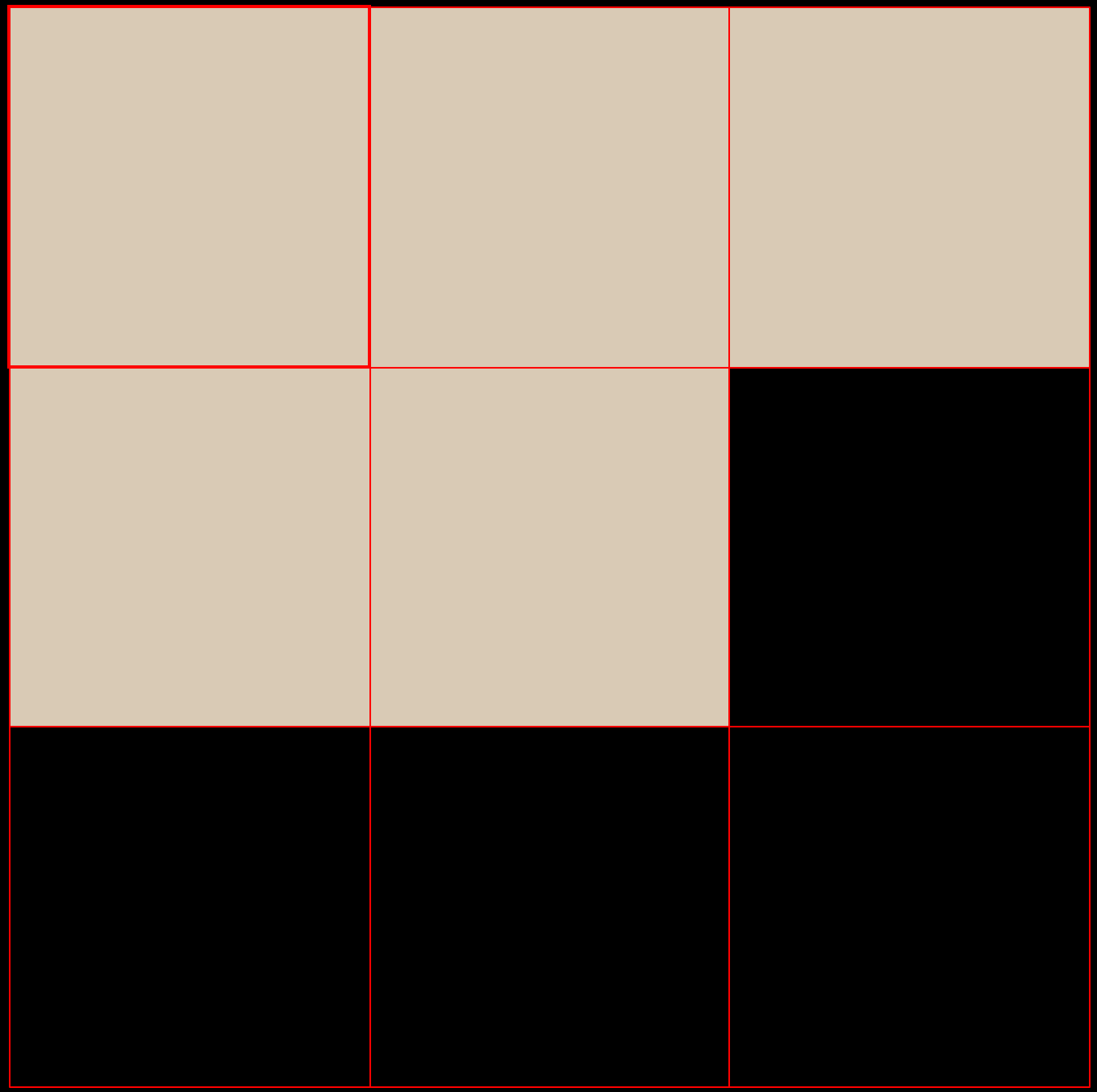


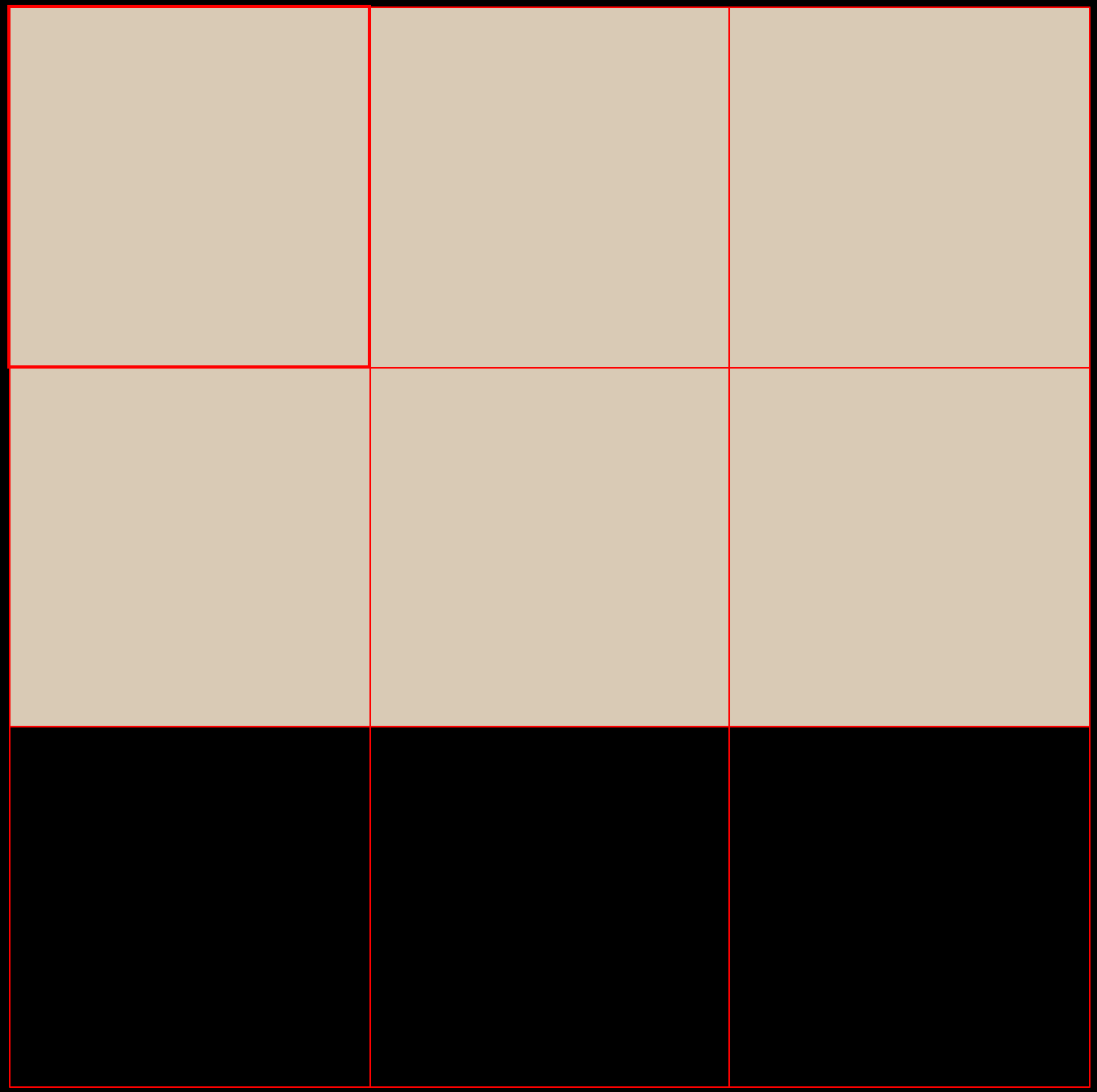


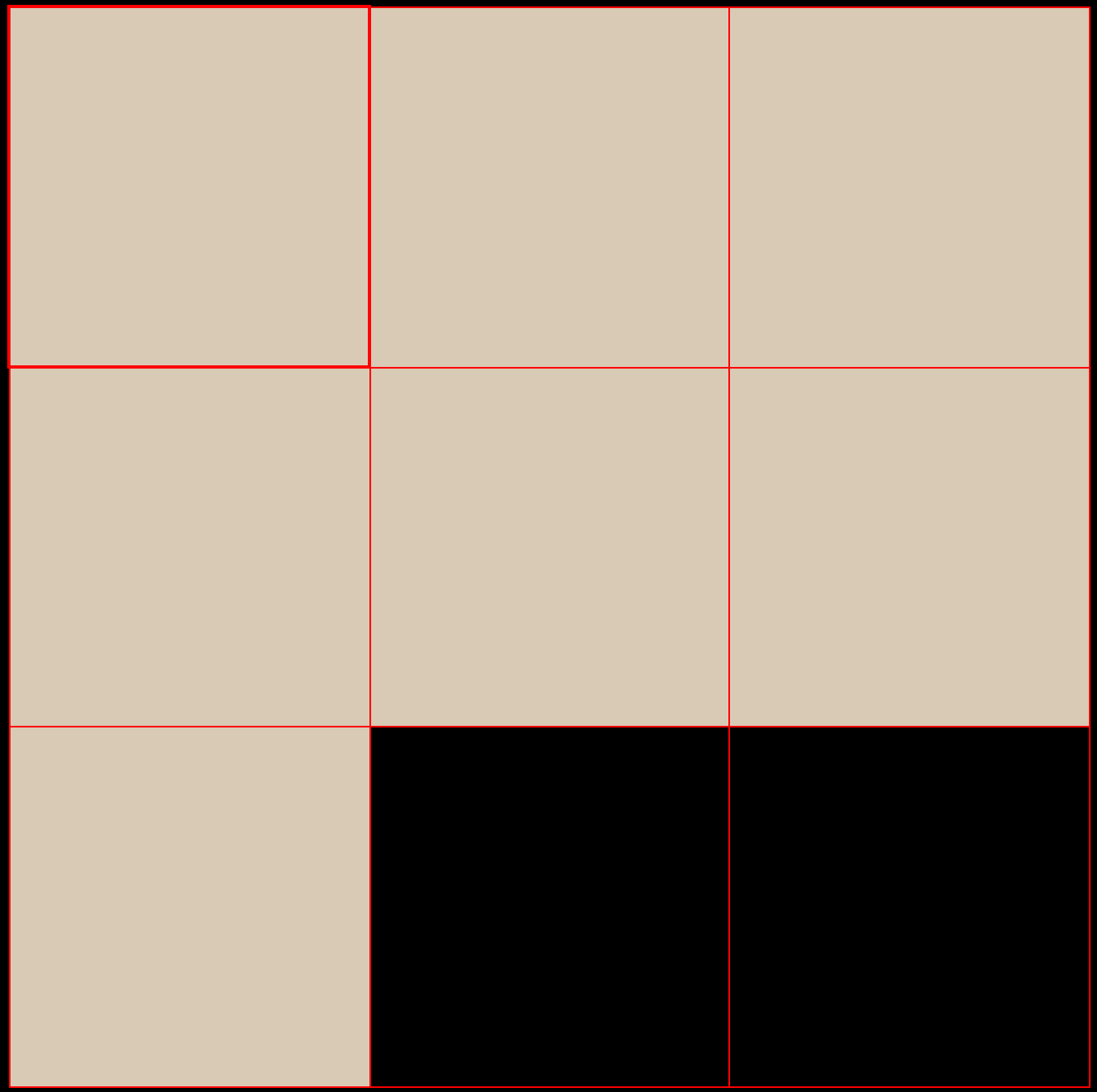




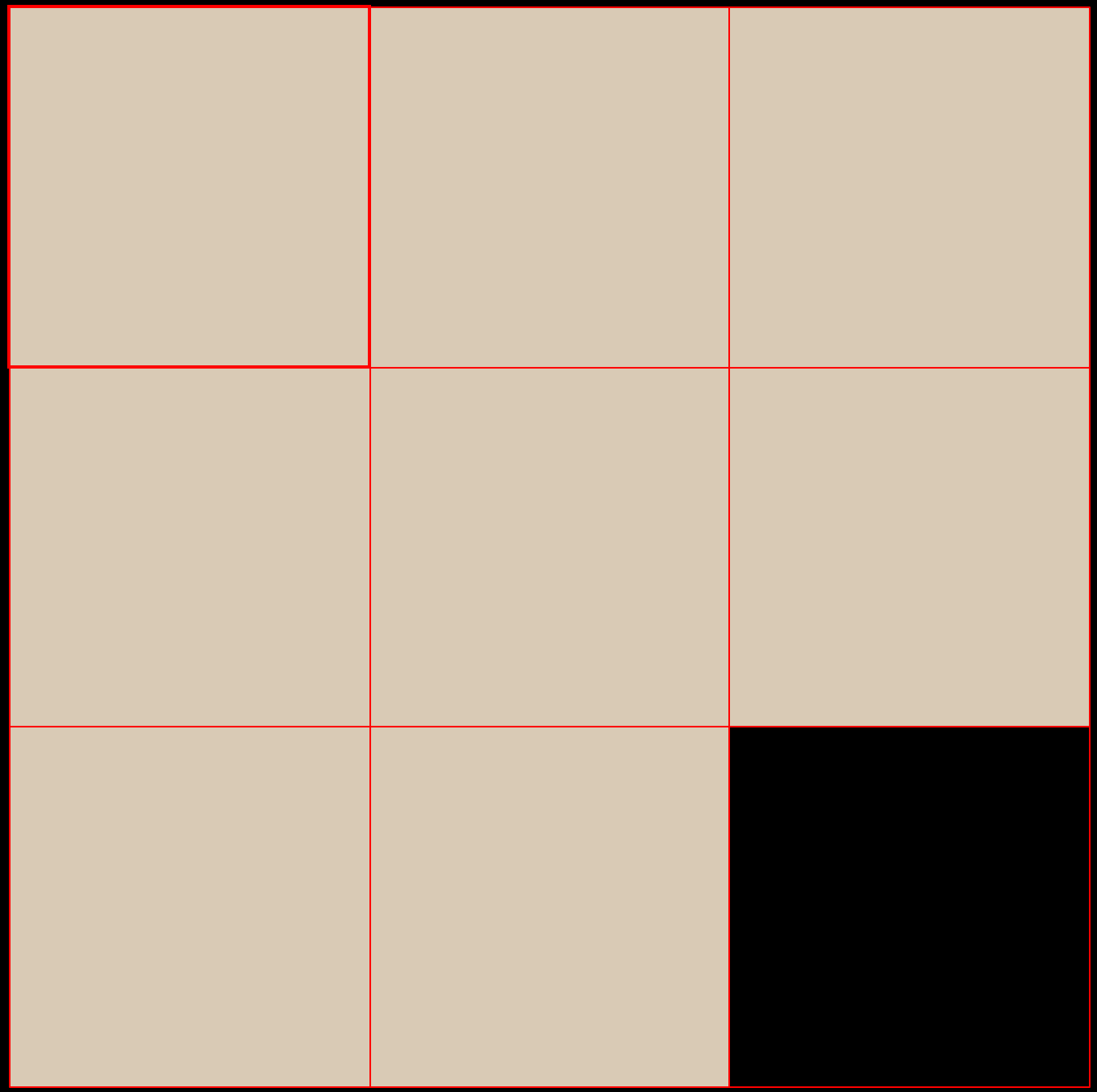


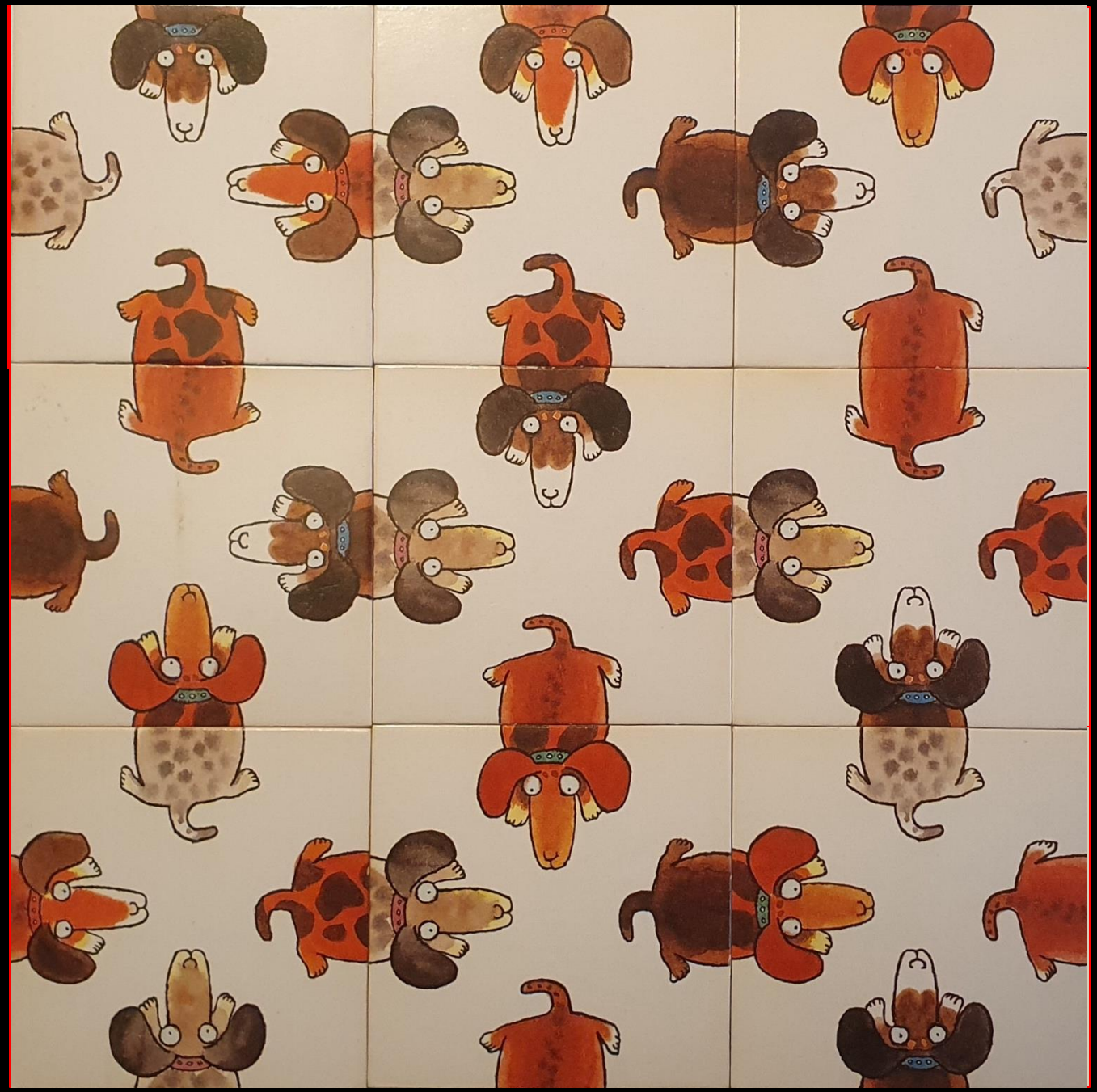




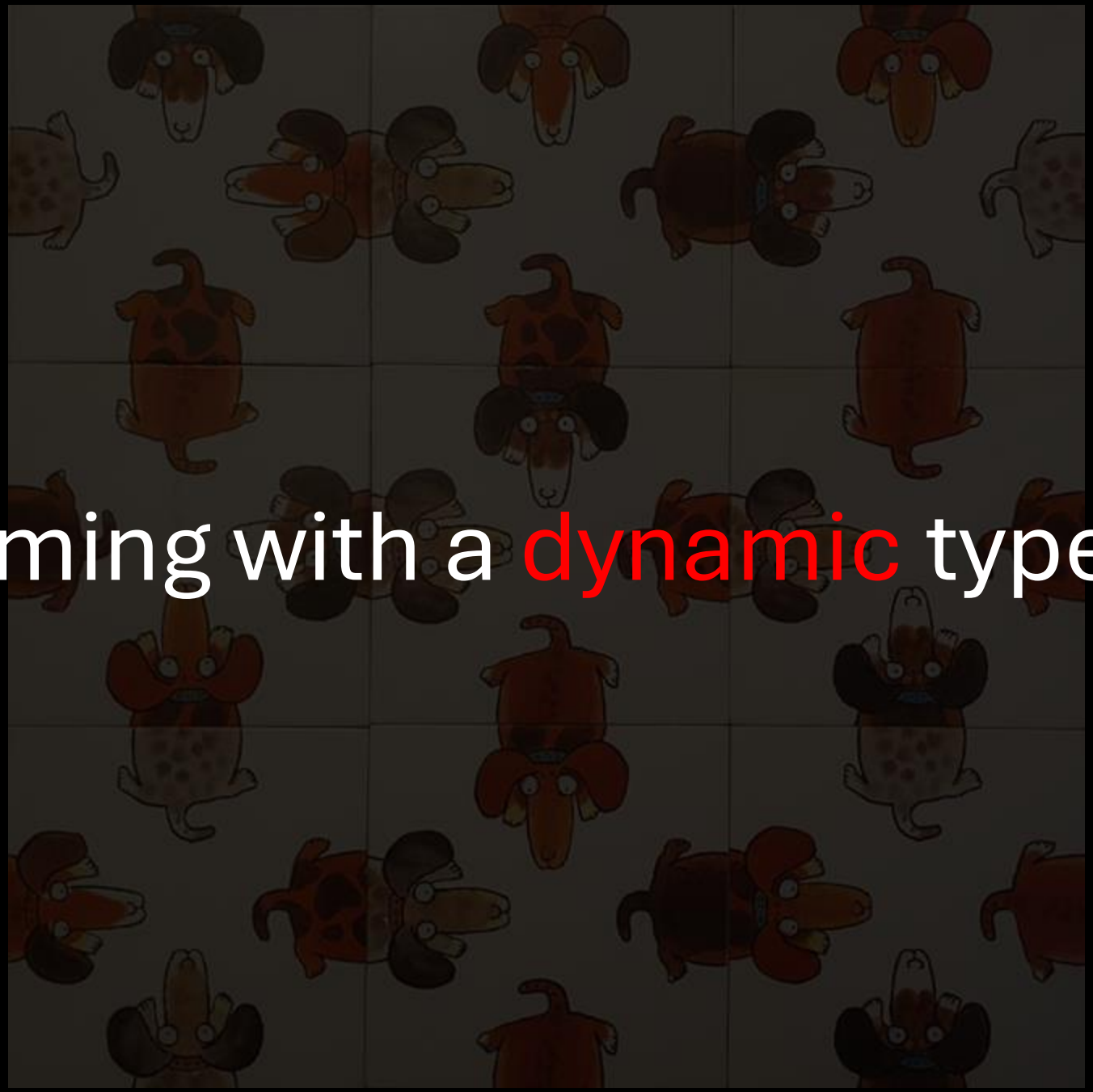


https://blog.ploeh.dk





https://blog.ploeh.dk



Programming with a dynamic type system

```
class RandomColorDimmer {
  constructor(inner, dimFactor = 1) {
    this.inner = inner;
    this.dimFactor = dimFactor;
  }

  getPetalColor(level) {
    const color = this.inner.getPe
    return d3.color(color).darker(
  }

  getCenterColor(level) {
    const color = this.inner.getCe
    return d3.color(color).darker()
  }
}
```

```
darker(k?: number): RGBColor | HSLColor
```

A color space dependent number to determine, how much darker the returned color should be.

Returns a darker copy of this color. If k is specified, it controls how much darker the returned color should be. If k is not specified, it defaults to 1.

```
class RandomColorDimmer {
  constructor(inner, dimFactor = 1) {
    this.inner = inner;
    this.dimFactor = dimFactor;
  }

  getPetalColor(level) {
    const color = this.inner.getPetalColor(level);
    return d3.color(color).darker(Math.random() * this.dimFactor);
  }

  getCenterColor(level) {
    const color = this.inner.getCenterColor(level);
    return d3.color(color).darker(Math.random() * this.)
  }
}
```

getCenterColor method

getPetalColor method

```
class RandomColorDimmer {
  constructor(inner, dimFactor = 1) {
    this.inner = inner;
    this.dimFactor = dimFactor;
  }

  getPetalColor(level) {
    const color = this.inner.getPetalColor(level);
    return d3.color(color).darker(Math.random() * this.dimFactor);
  }

  getCenterColor(level) {
    const color = this.inner.
    return d3.color(color).darker(Math.random() * this.dimFactor);
  }
}
```

```
class RandomColorDimmer {
  constructor(inner, dimFactor = 1) {
    this.inner = inner;
    this.dimFactor = dimFactor;
  }

  getPetalColor(level) {
    const color = this.inner.getPetalColor(level);
    return d3.color(color).darker(Math.random() * this.dimFactor);
  }

  getCenterColor(level) {
    const color = this.inner.g
    return d3.color(color).da                    imFactor);
  }
}
```

getCenterColor local

getPetalColor local

```
class RandomColorDimmer {
  constructor(inner, dimFactor = 1) {
    this.inner = inner;
    this.dimFactor = dimFactor;
  }

  getPetalColor(level) {
    const color = this.inner.getPetalColor(level);
    return d3.color(color).darker(Math.random() * this.dimFactor);
  }

  getCenterColor(level) {
    const color = this.inner.getCenterColor(
    return d3.color(color).darker(Math.random() * this.dimFactor);
  }
}
```

```
arr = load_array('input.12.1.txt')
```

```
(variable) arr: NDArray[Any]
arr = load_array('input.12.1.txt')
```

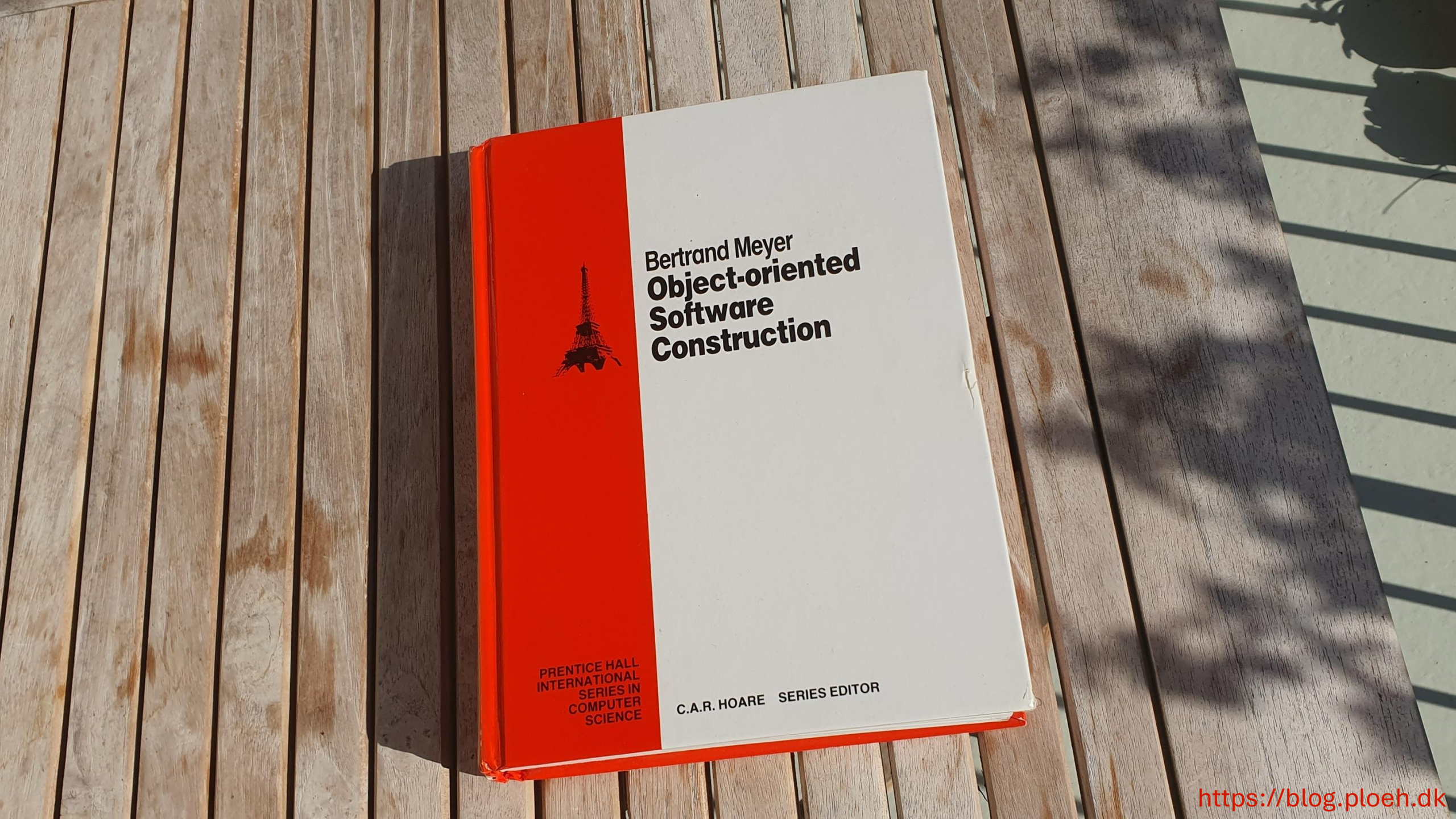
```
arr = load_array('i
g = build_graph(arr)
```

```
(arr: Any) -> Any
```

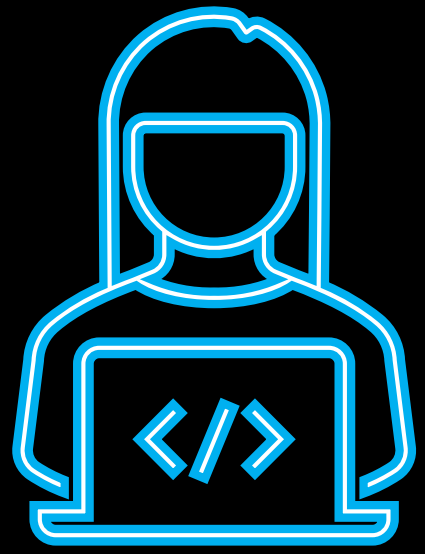
```
=> (doc range)
 -------------------------
 clojure.core/range
 ([] [end] [start end] [start end step])
 Returns a lazy seq of nums from start (inclusive) to
end (exclusive), by step, where start defaults to 0,
step to 1, and end to infinity.
 nil
```

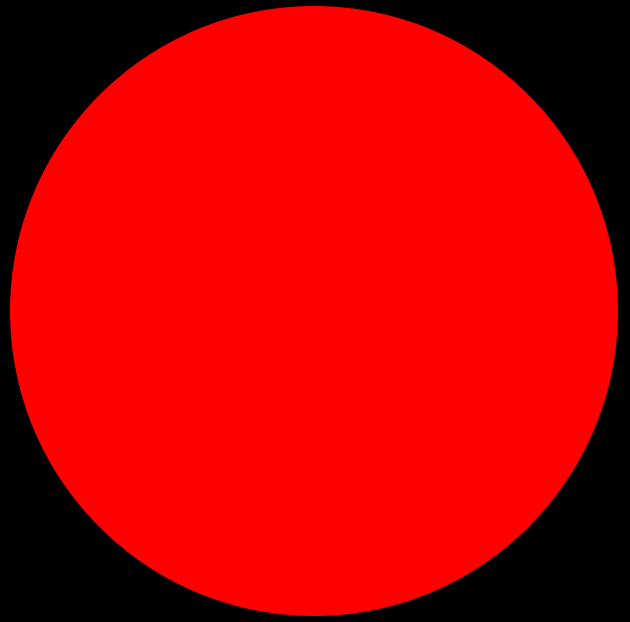
```
=> (doc foo)
-------------------------
user/foo
nil
```

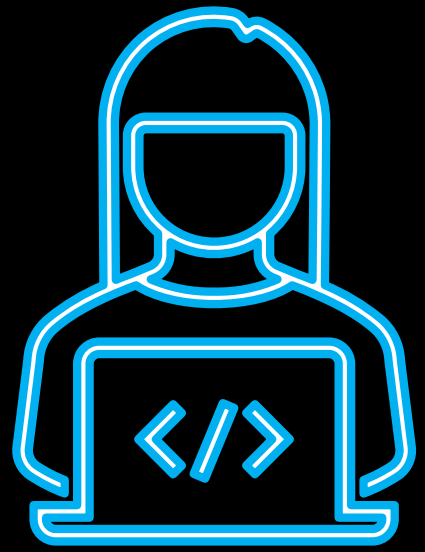
Encapsulation

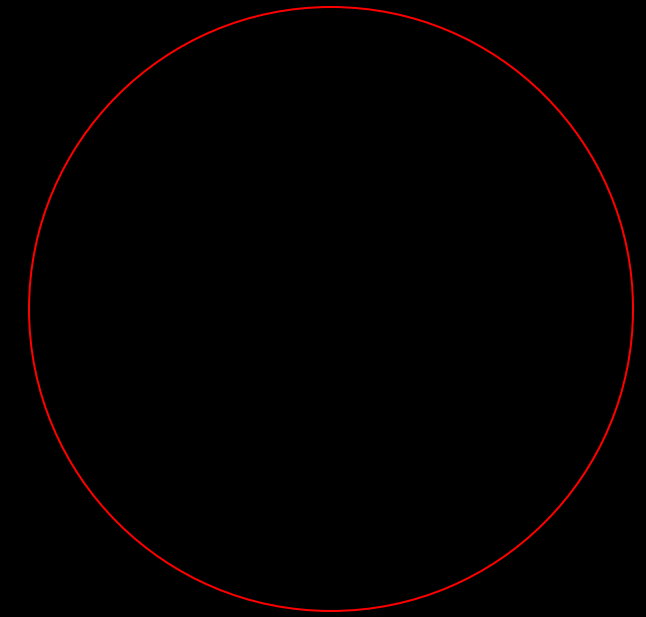


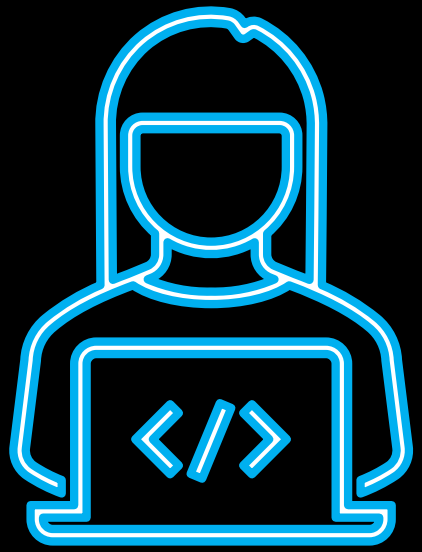
https://blog.ploeh.dk

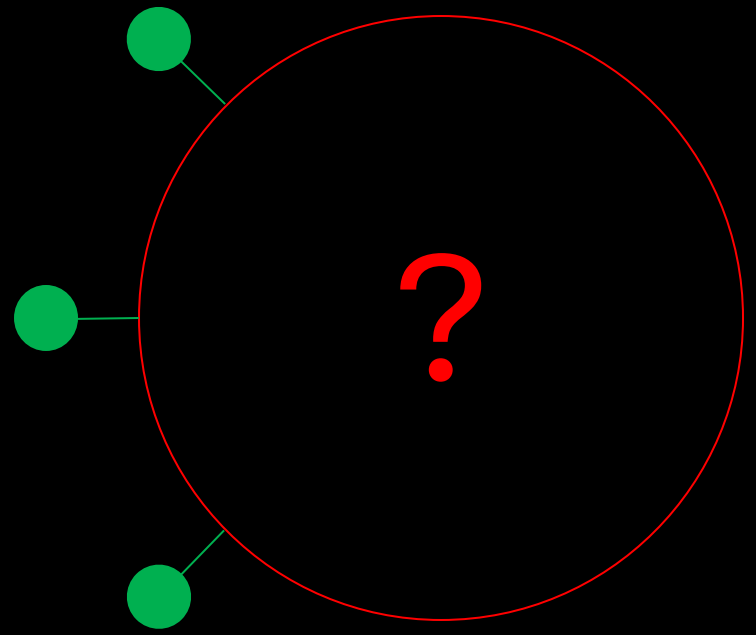




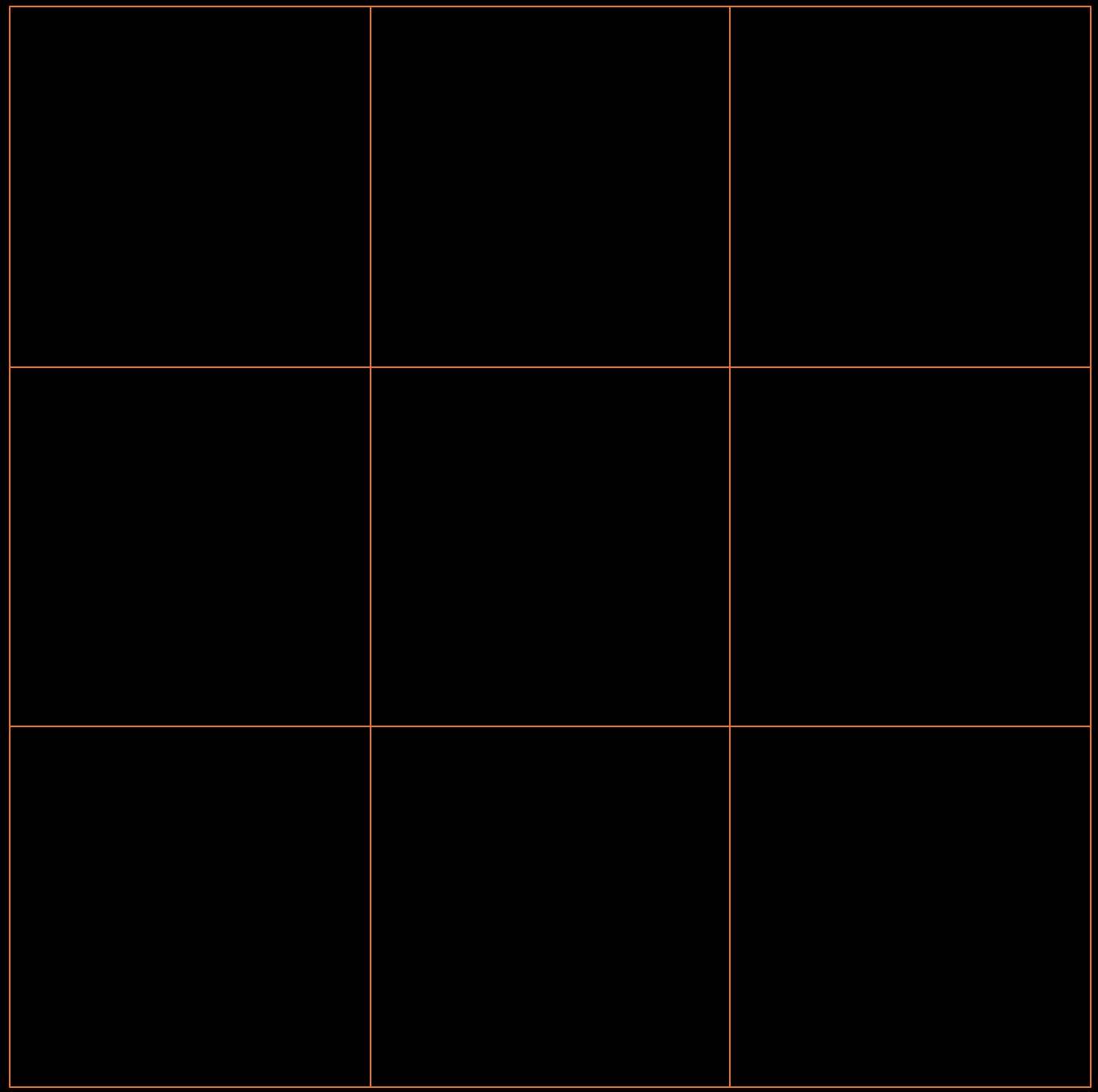








A fairer representation of composition with dynamic types



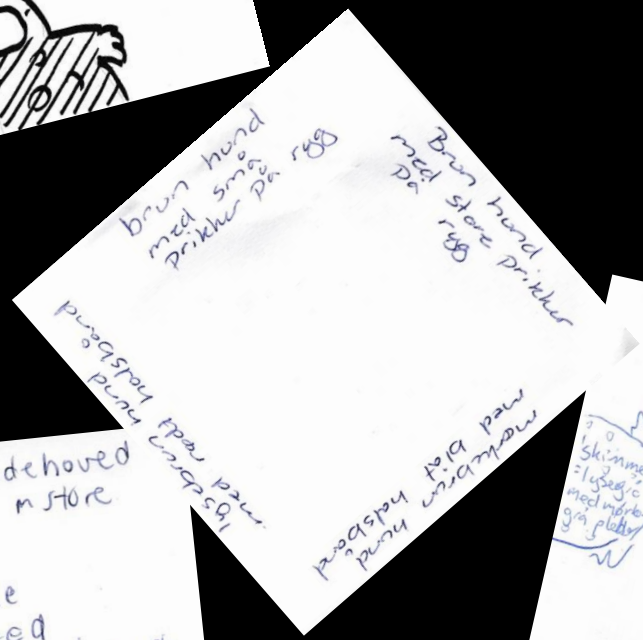


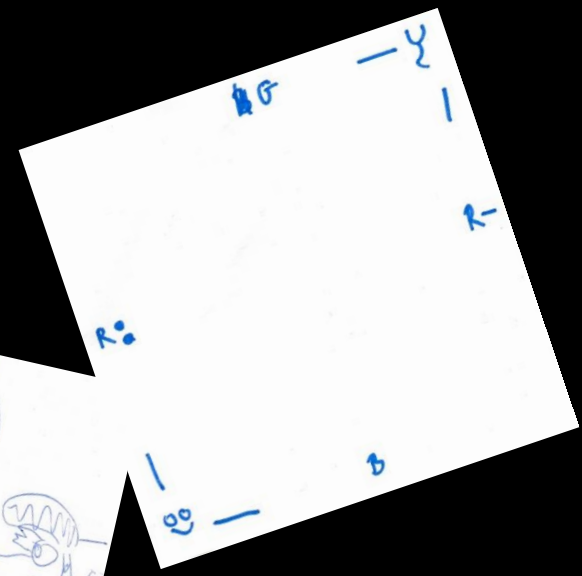


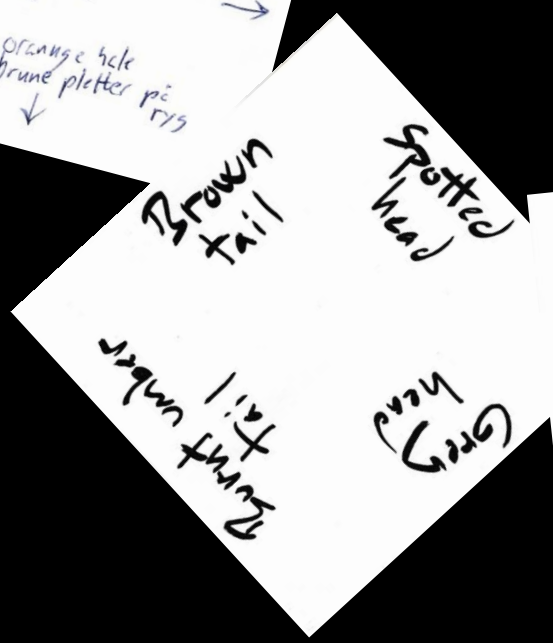


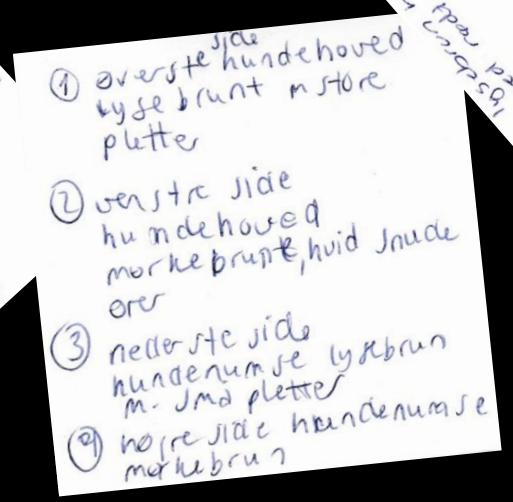


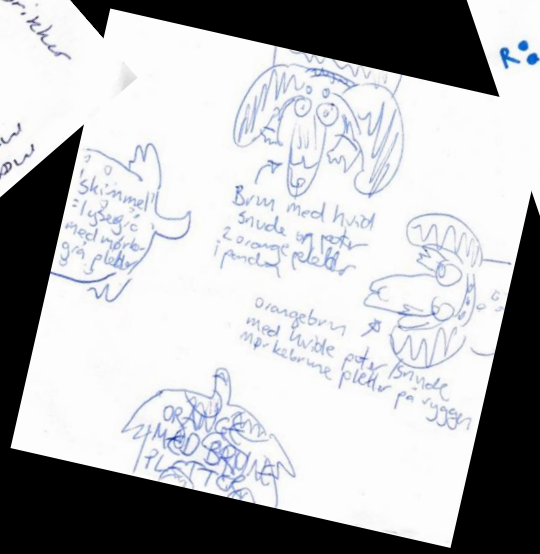




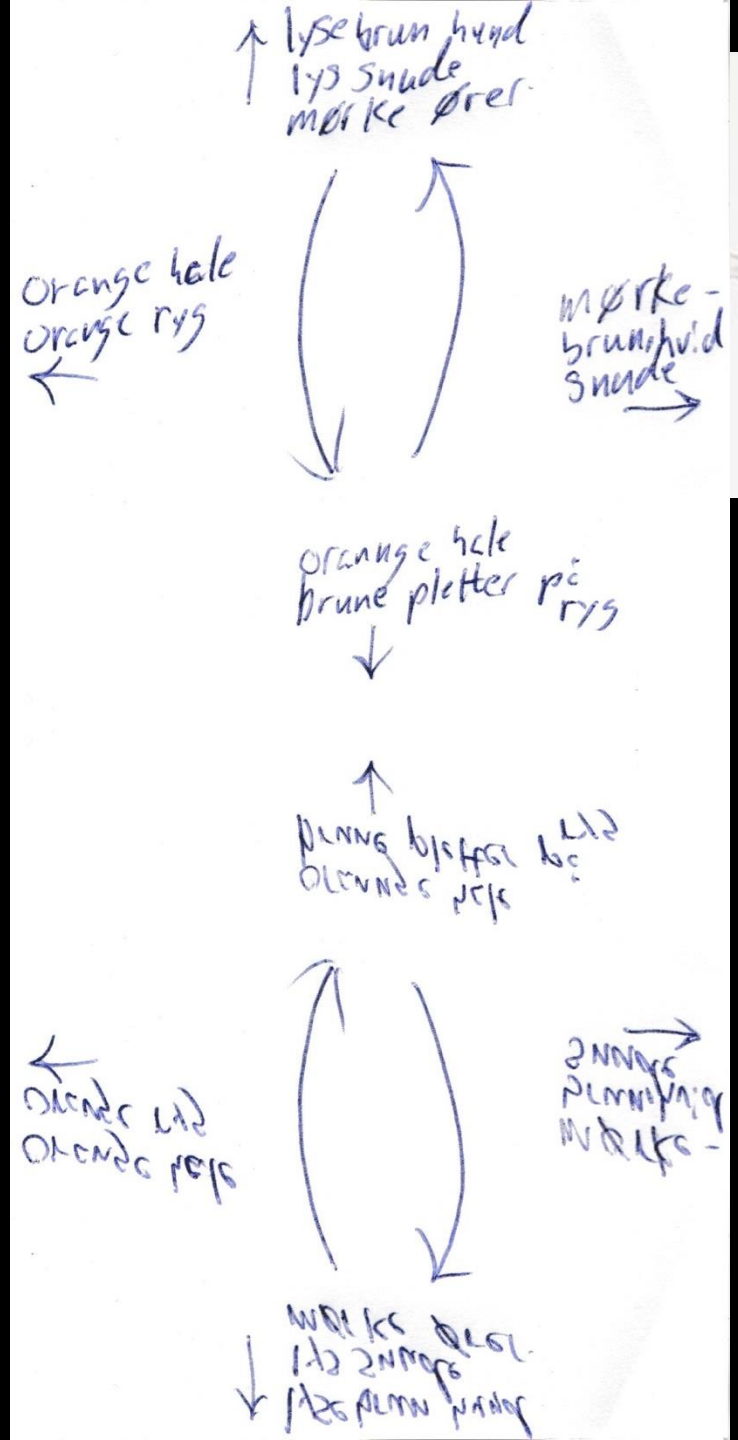




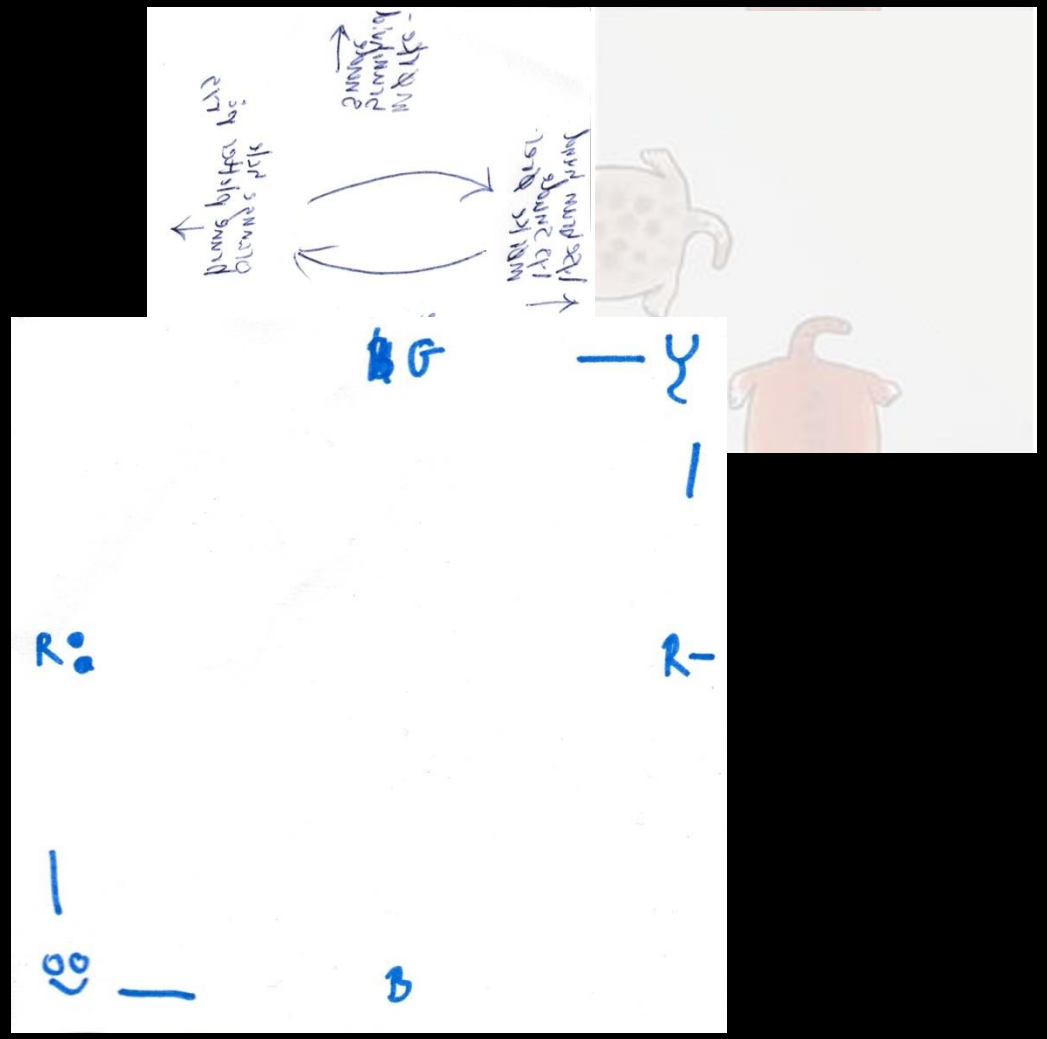




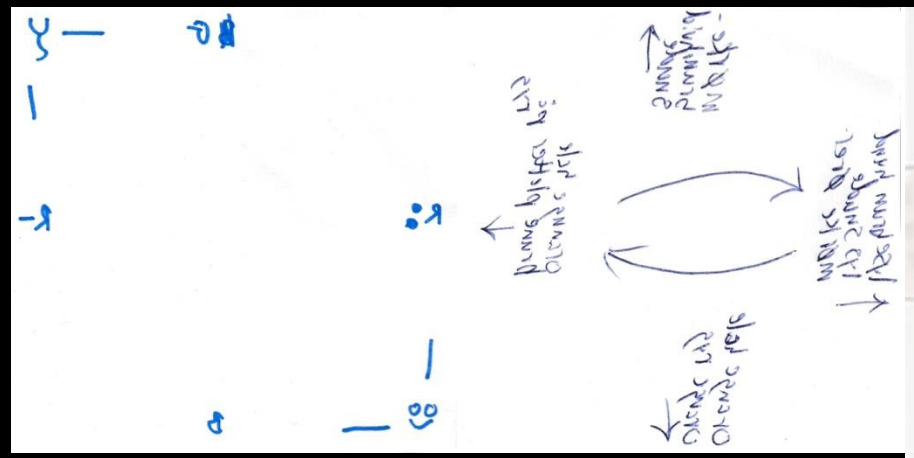
https://blog.ploeh.dk

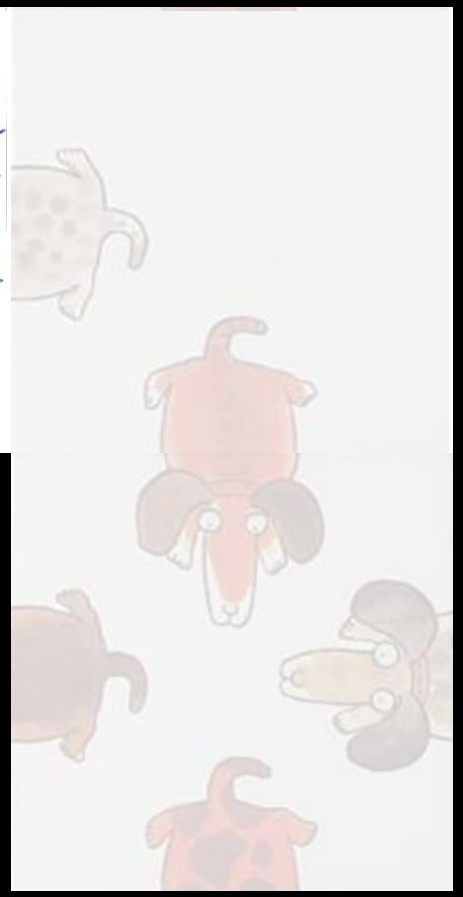


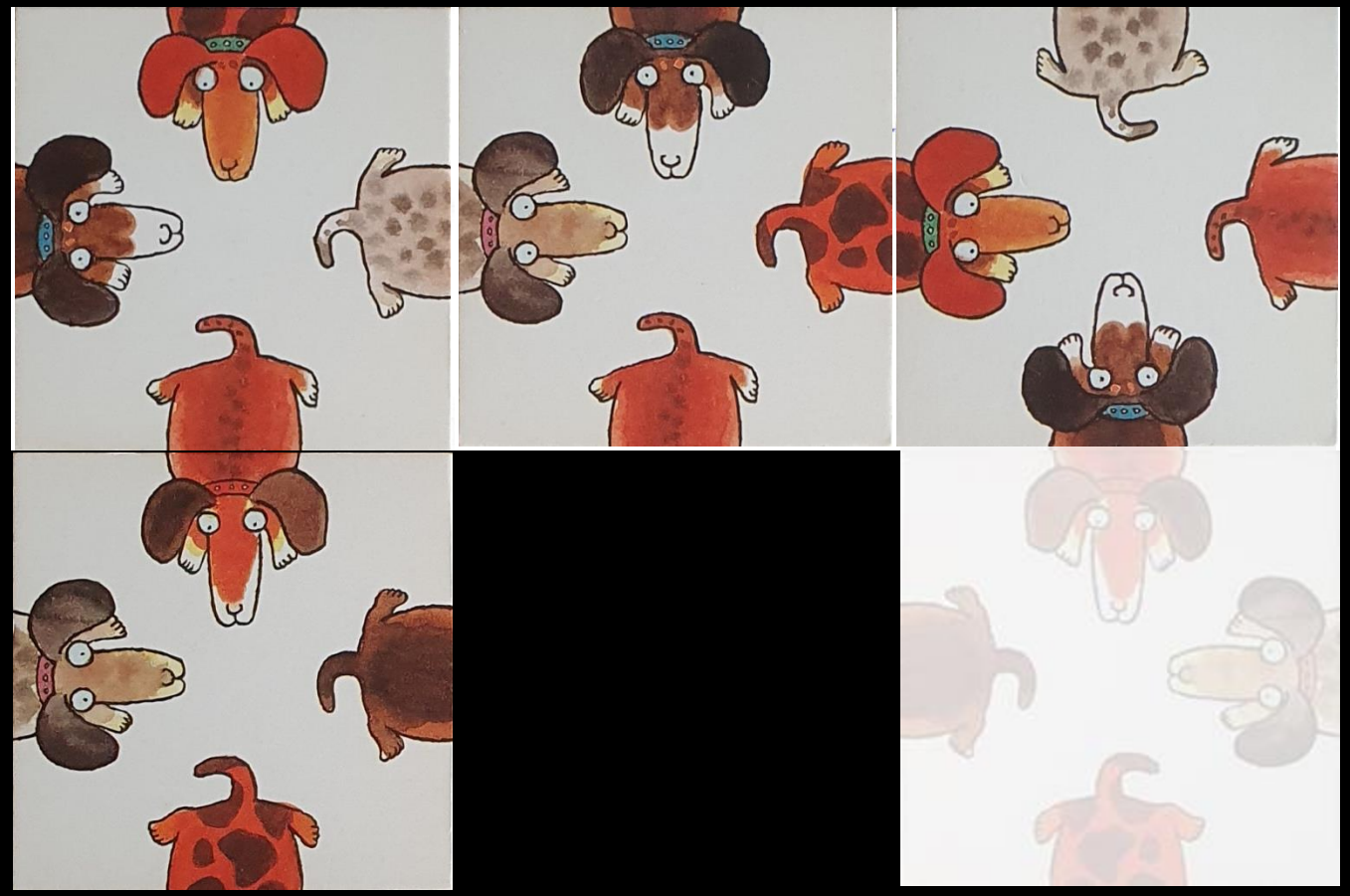
https://blog.ploeh.dk

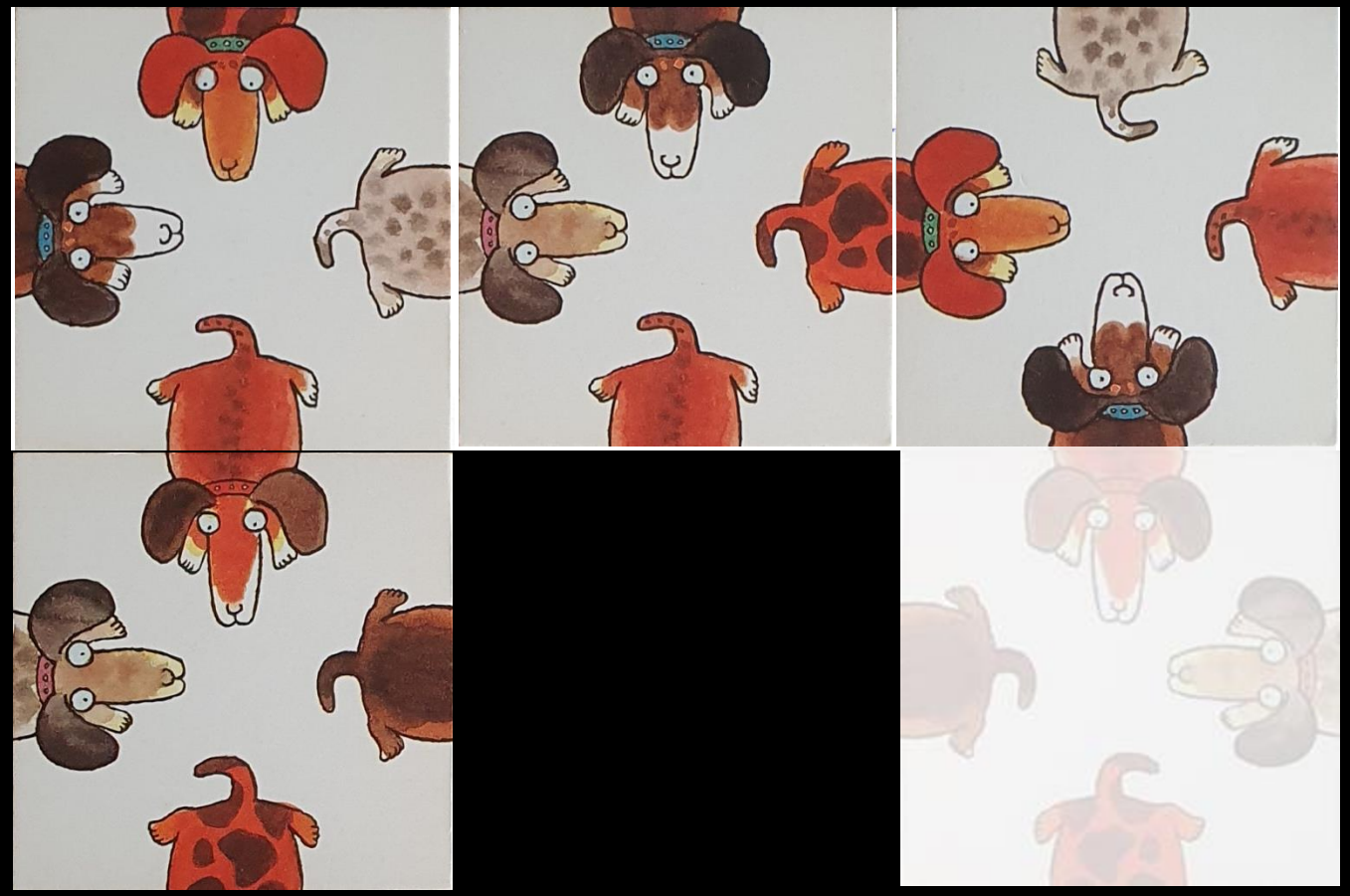


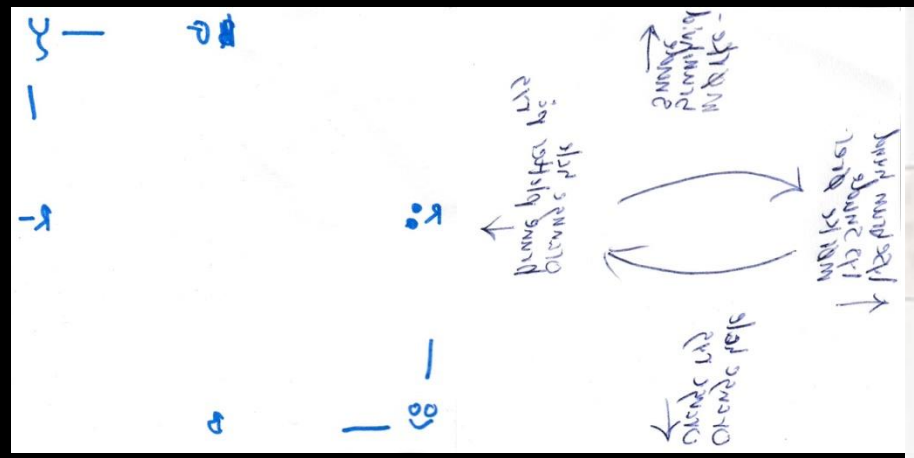
https://blog.ploeh.dk

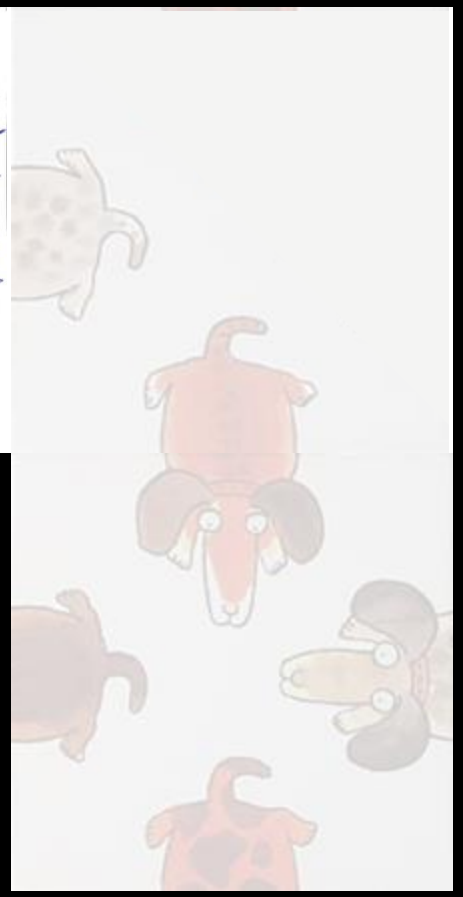


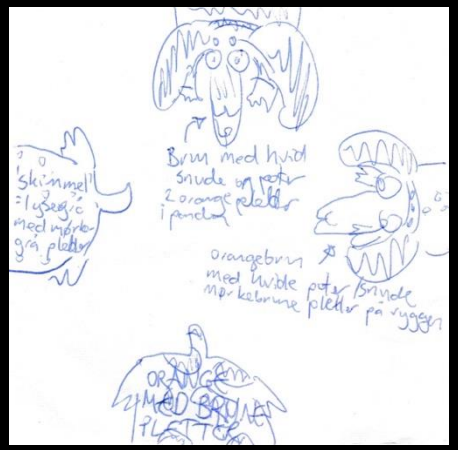








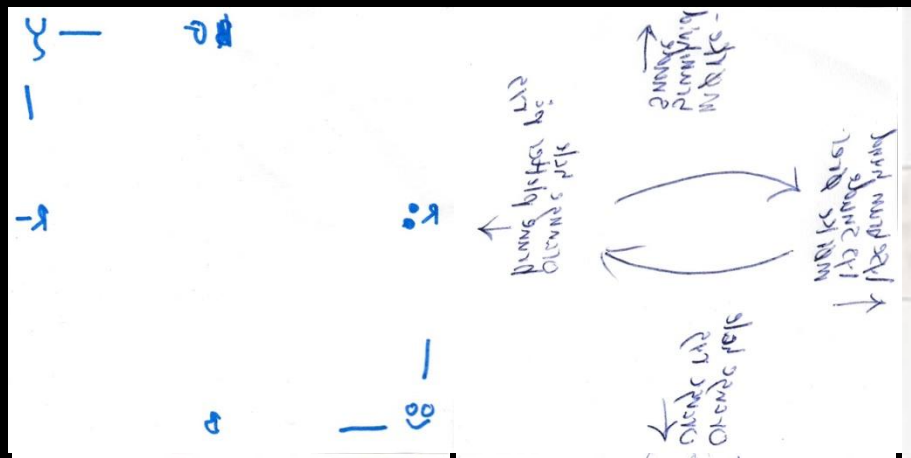




https://blog.ploeh.dk

Brown hair
Spotted head
Grey head
Brown hair
Short tail

Grey head
Brown tail
Spotted head
Burnt umber tail

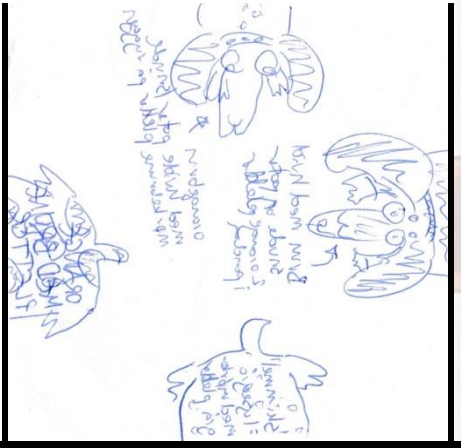


Brown tail

Spotted head

Grey head

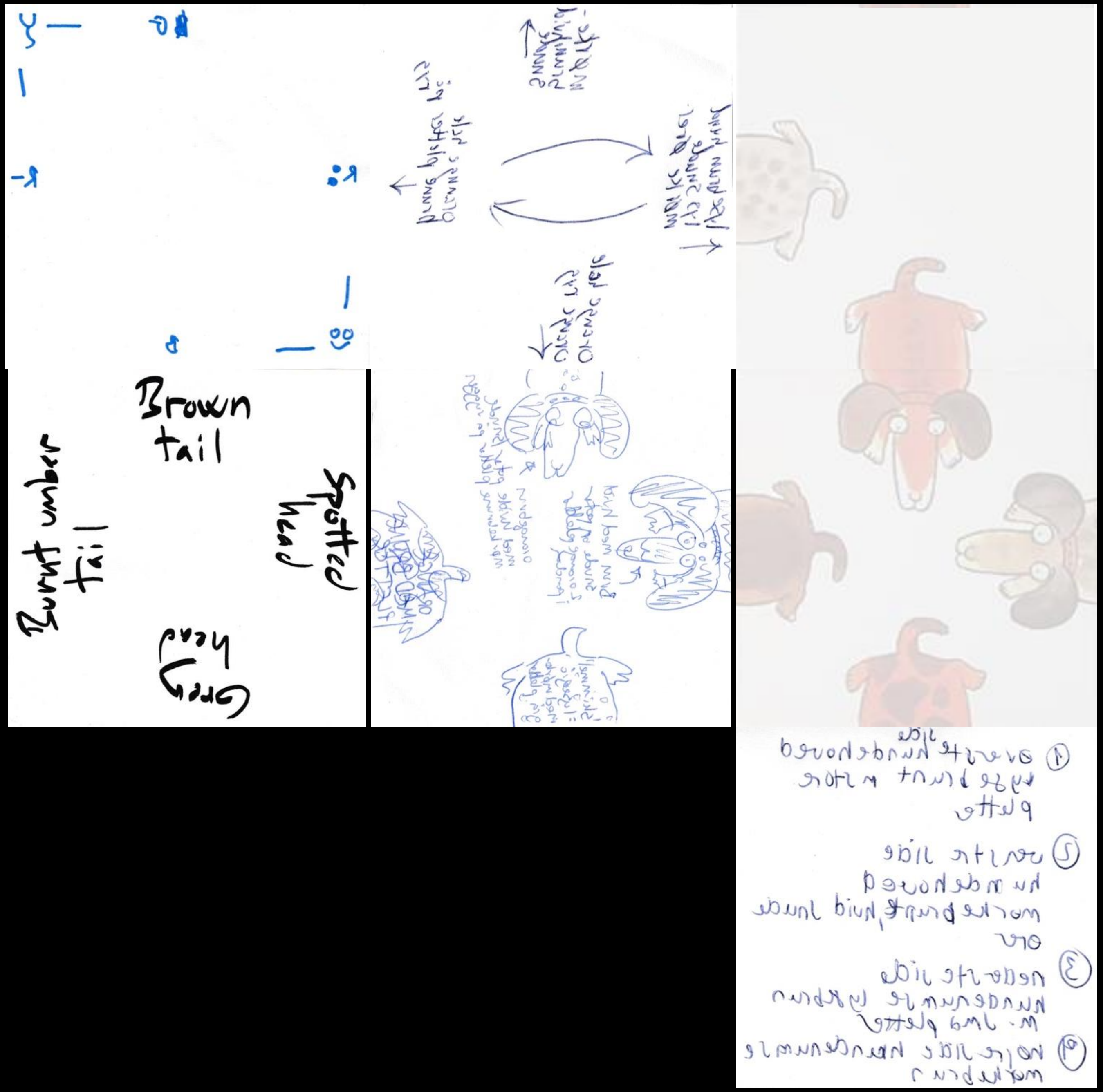
Burnt umber tail

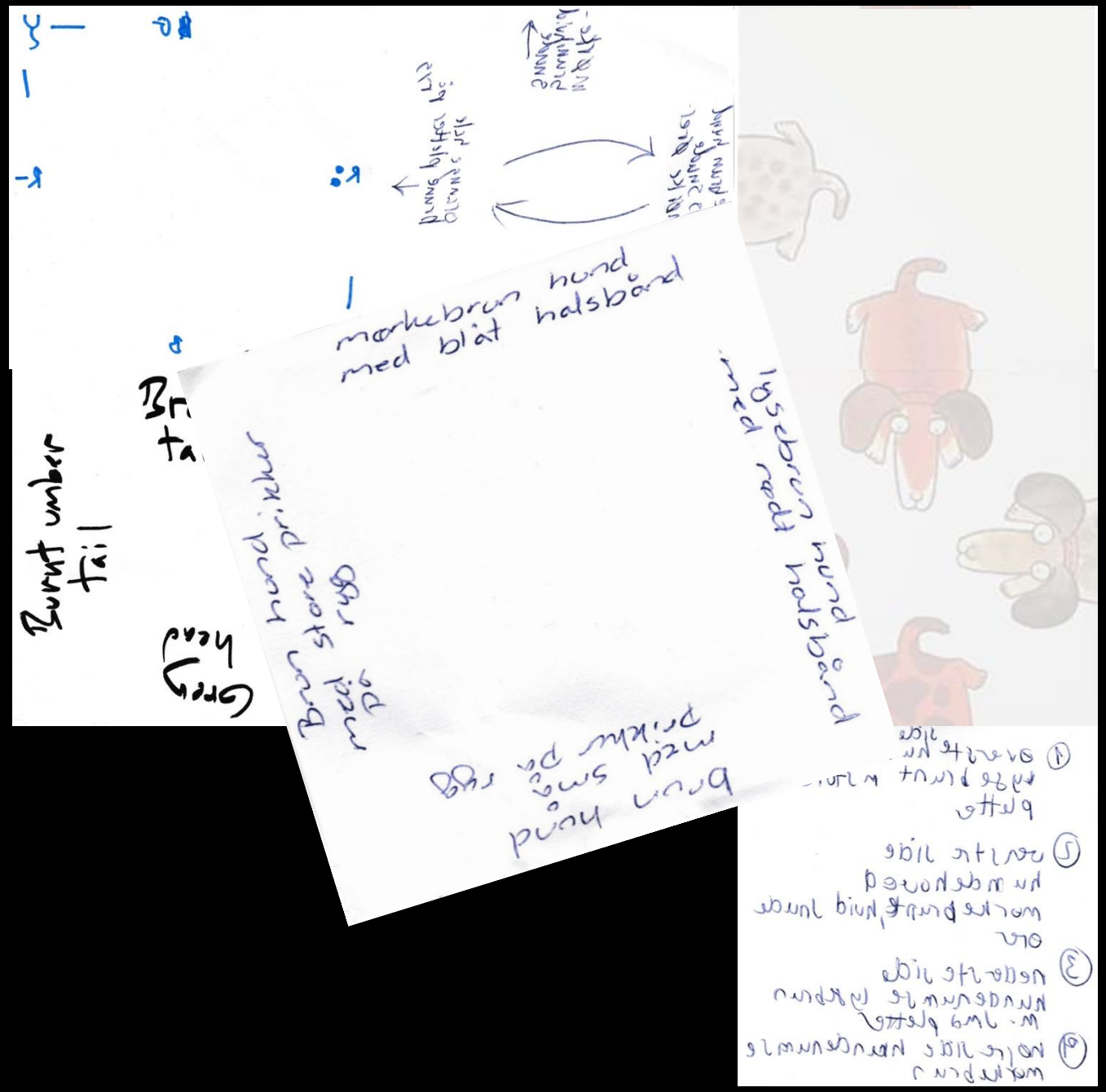




1. øverste side hundehoved lyse brunt m store plette
2. venstre side hundehoved mørke brunt, hvid snude over
3. nederste side hundenumse lysebrun m. små plette
4. højre side hundenumse mørkebrun

https://blog.ploeh.dk


Brown tail
Burnt umber tail
Spotted head
Grey head



https://blog.ploeh.dk

https://blog.ploeh.dk

https://blog.ploeh.dk

https://blog.ploeh.dk

https://blog.ploeh.dk

Grey head

Brown tail

Spotted head

Burnt umber tail

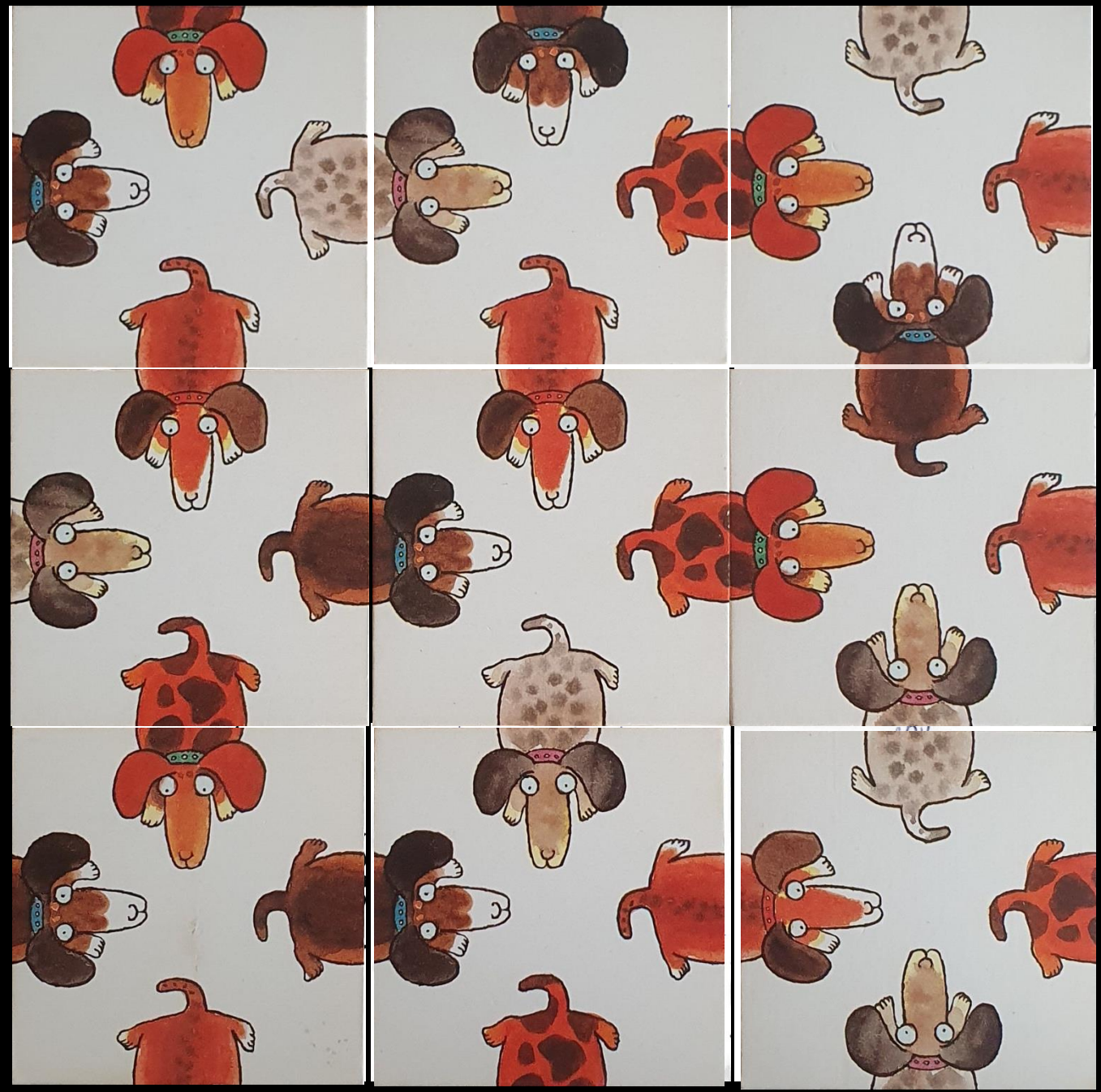
https://blog.ploeh.dk

https://blog.ploeh.dk

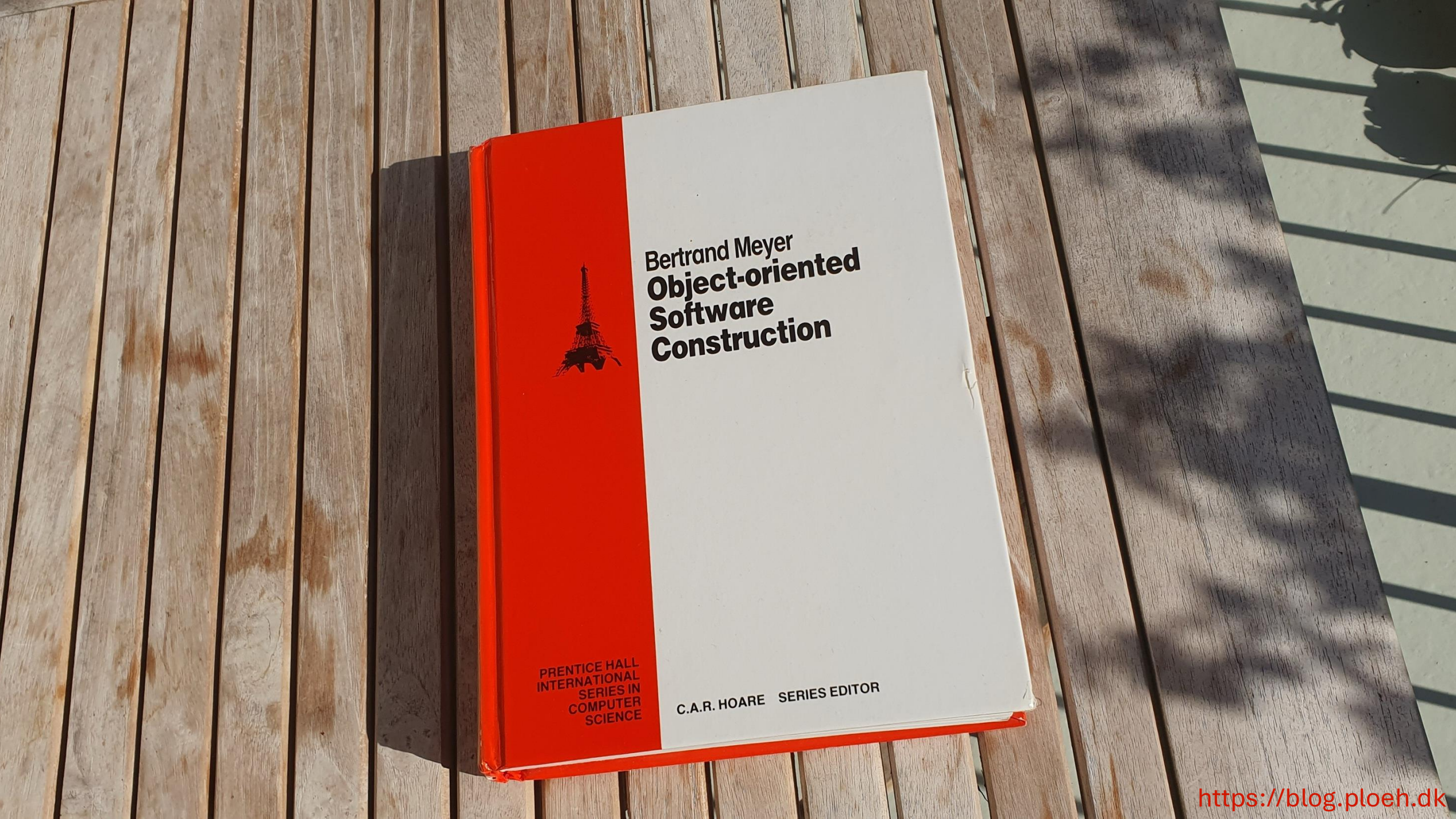
https://blog.ploeh.dk

Grey head
Brown tail
Spotted head
Burnt umber tail

https://blog.ploeh.dk

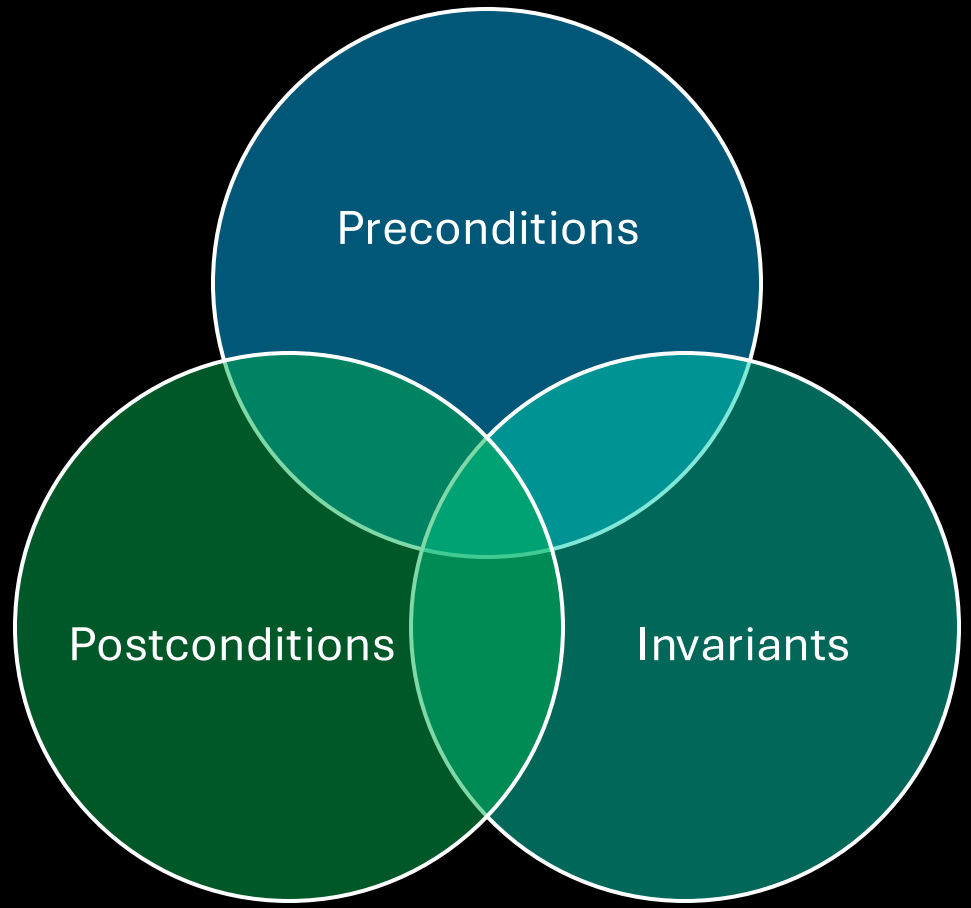


Encapsulation

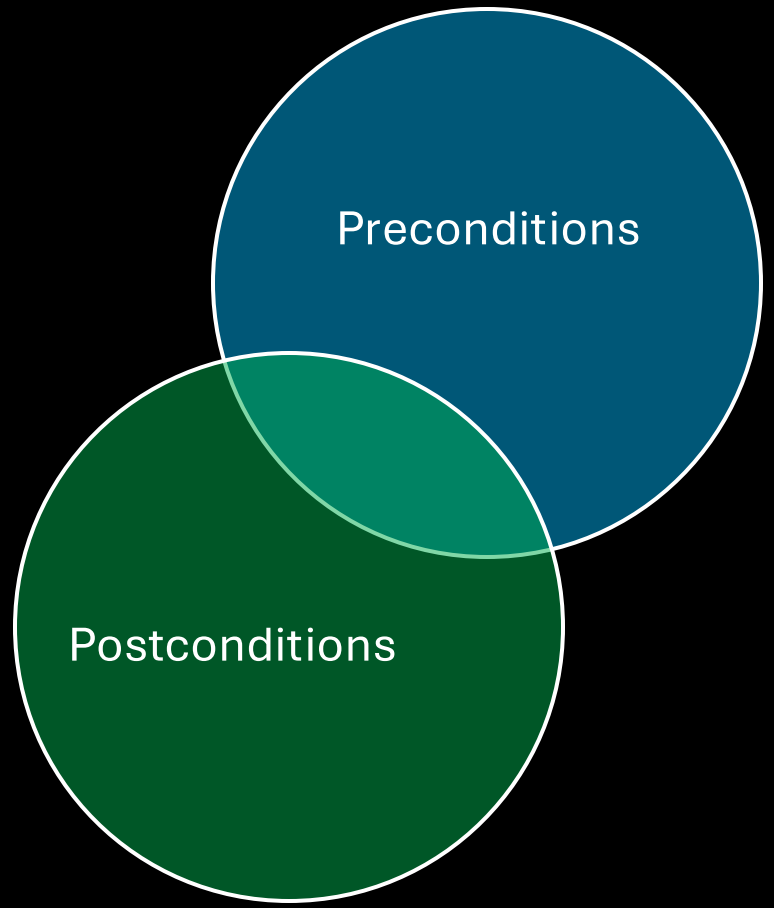


https://blog.ploeh.dk

Contract



Contract



Preconditions: input is a number

Postconditions: output is colour information

```
class RandomColorDimmer {
  constructor(inner, dimFactor = 1) {
    this.inner = inner;
    this.dimFactor = dimFactor;
  }

  getPetalColor(level) {
    const color = this.inner.getPe
    return d3.color(color).darker(
  }

  getCenterColor(level) {
    const color = this.inner.getCe
    return d3.color(color).darker()
  }
}
```

```
darker(k?: number): RGBColor | HSLColor
```

A color space dependent number to determine, how much darker the returned color should be.

Returns a darker copy of this color. If k is specified, it controls how much darker the returned color should be. If k is not specified, it defaults to 1.

```
darker(k?: number): RGBColor | HSLColor
```

```
darker(k?: number): RGBColor | HSLColor
```

```
type Color =
    RGBColor of int * int * int | HSLColor of int16 * byte * byte

let darker(k: float option): Color = // ...
```

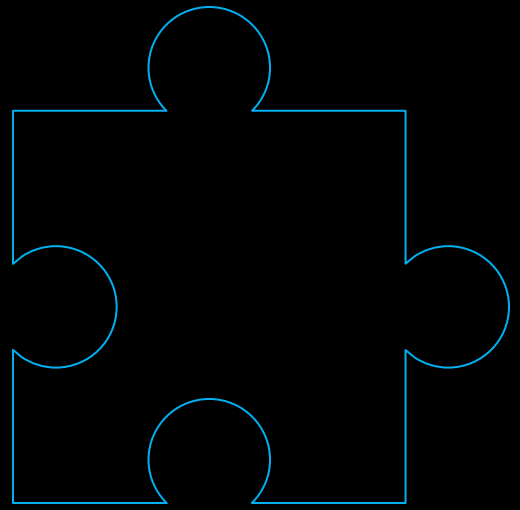
Static types to the rescue?

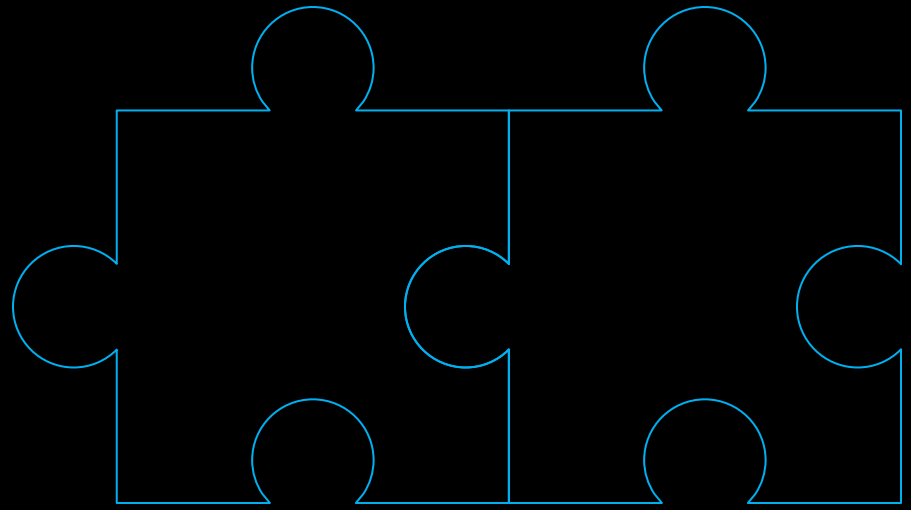
```
public void Reserve(
    string id,
    string at,
    string email,
    string name,
    int quantity)
{
    // ...
}
```

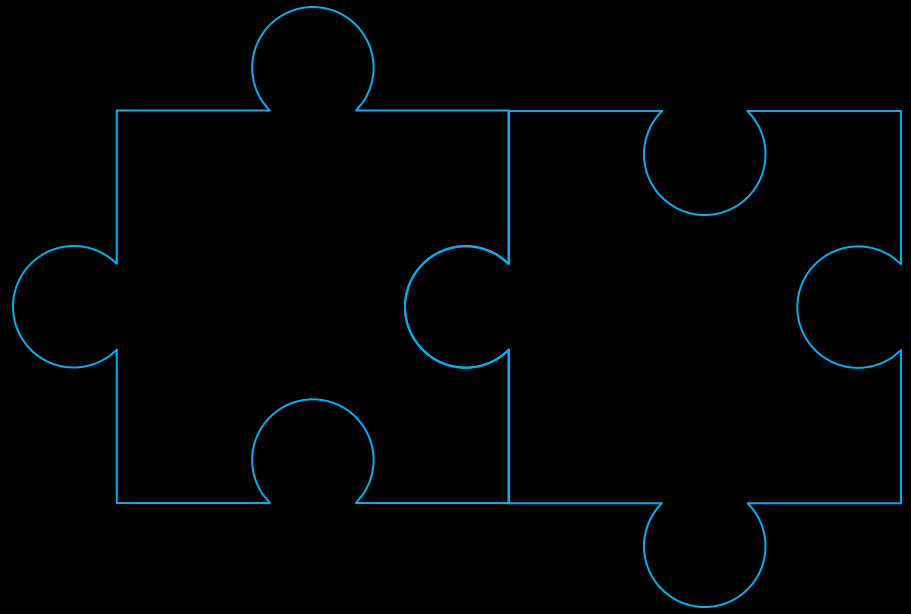
```
public void Reserve(Reservation reservation)
{
    // ...
}
```

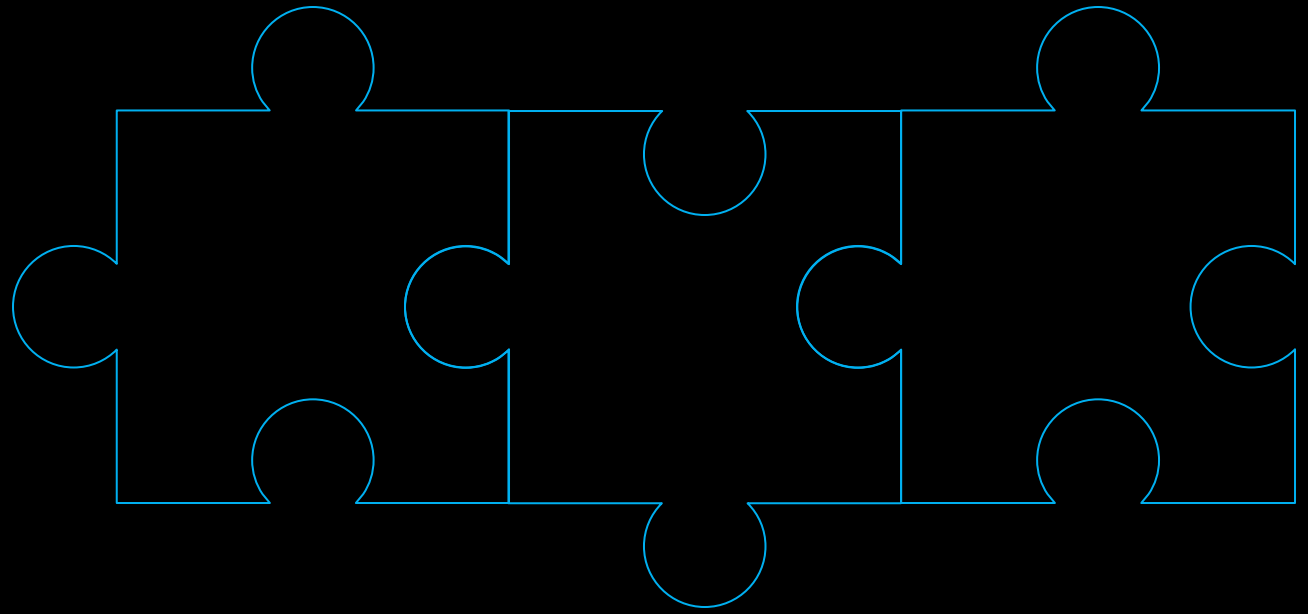
```
public class Reservation
{
    // ...
    public string Id { get; set; }
    public string At { get; set; }
    public string Email { get; set; }
    public string Name { get; set; }
    public int Quantity { get; set; }
    // ...
}
```

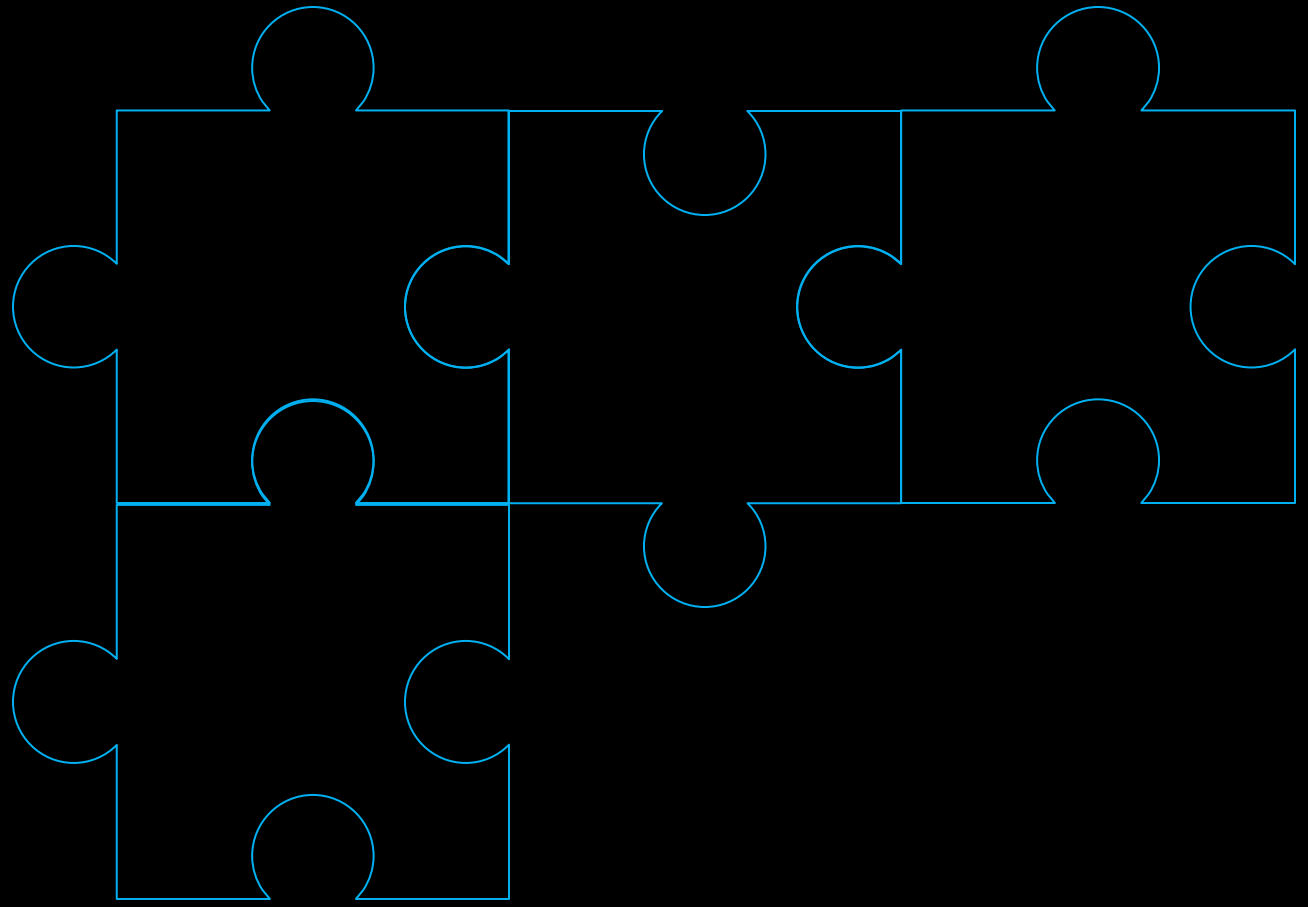
A representation of composition
with stringly typed code

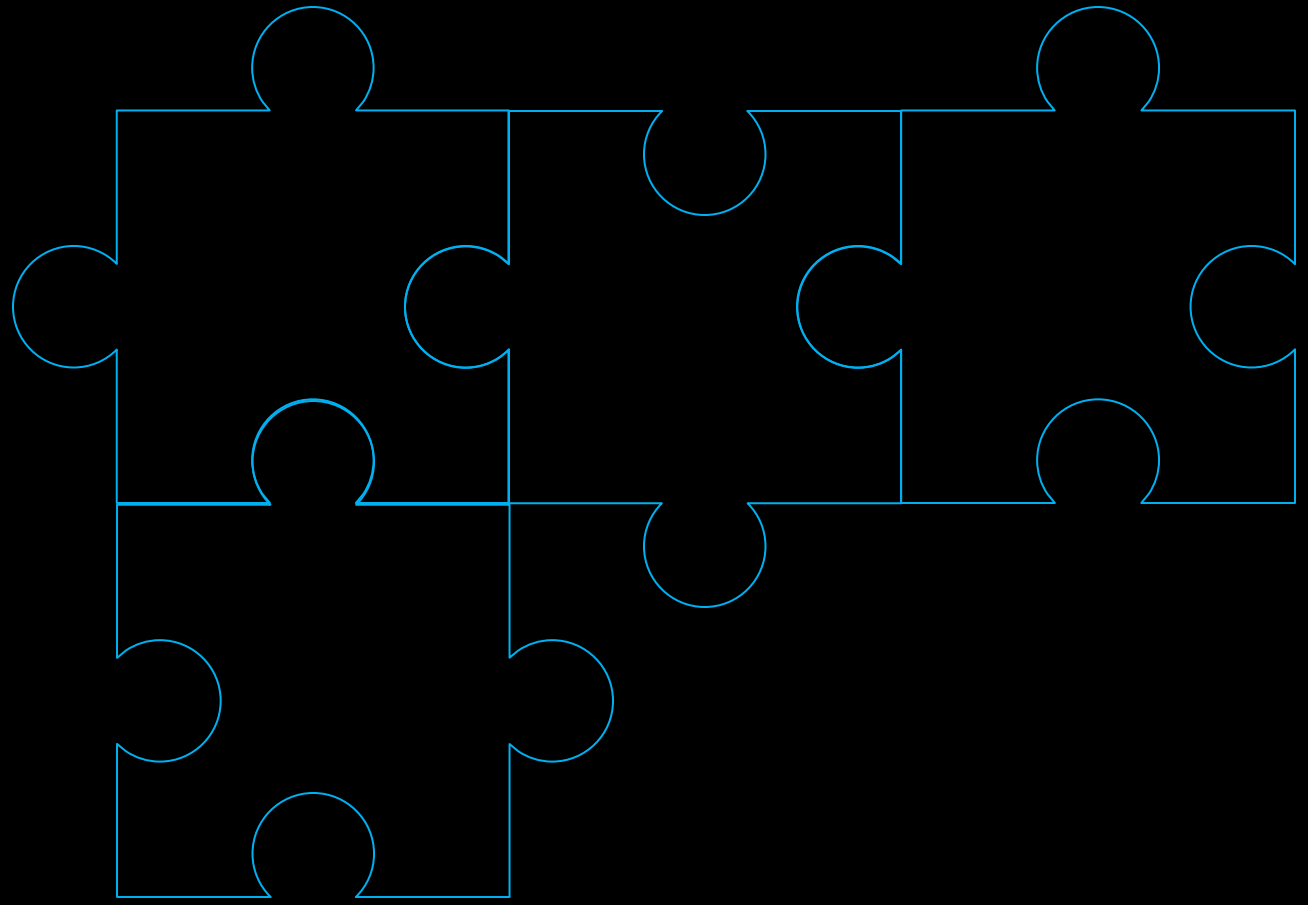


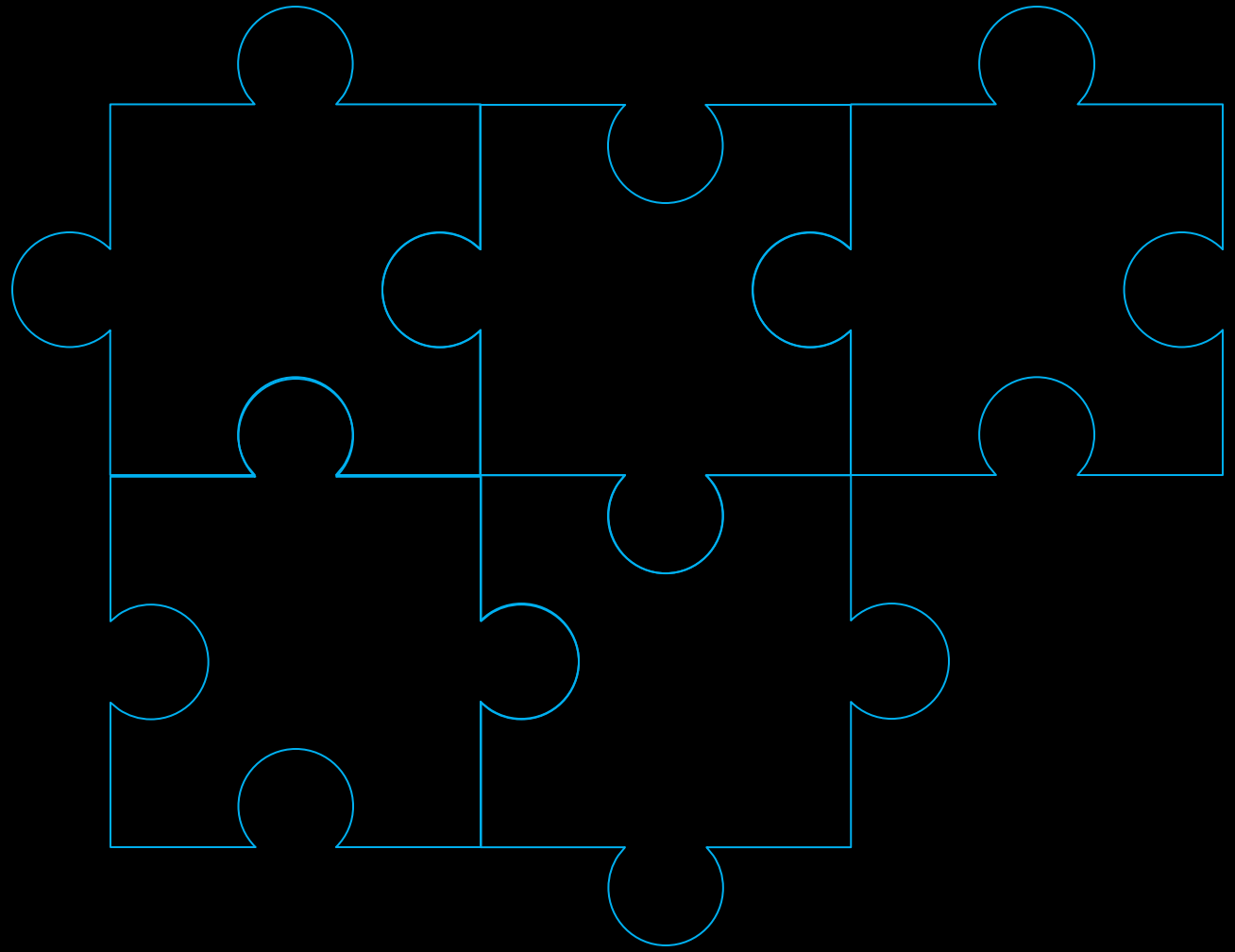


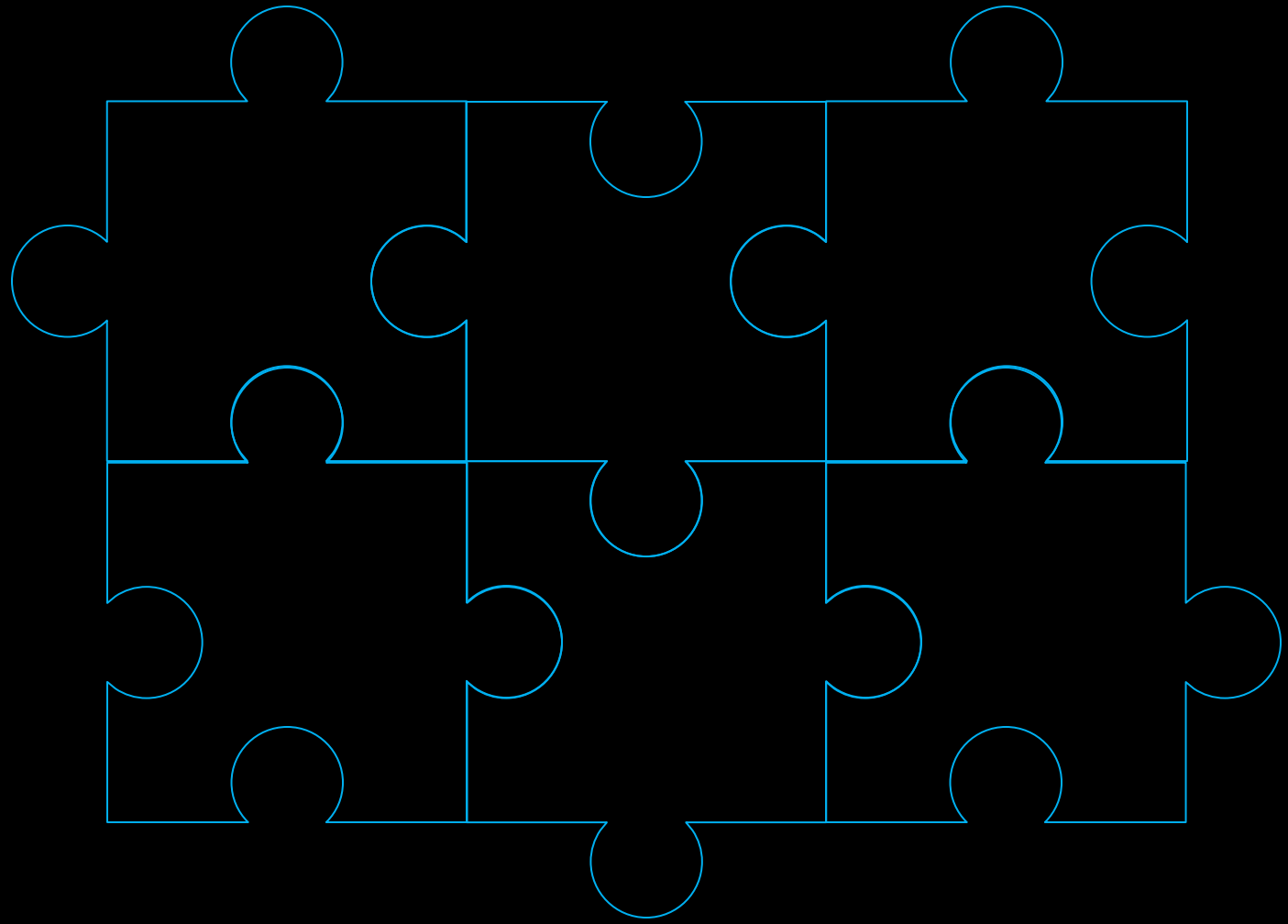


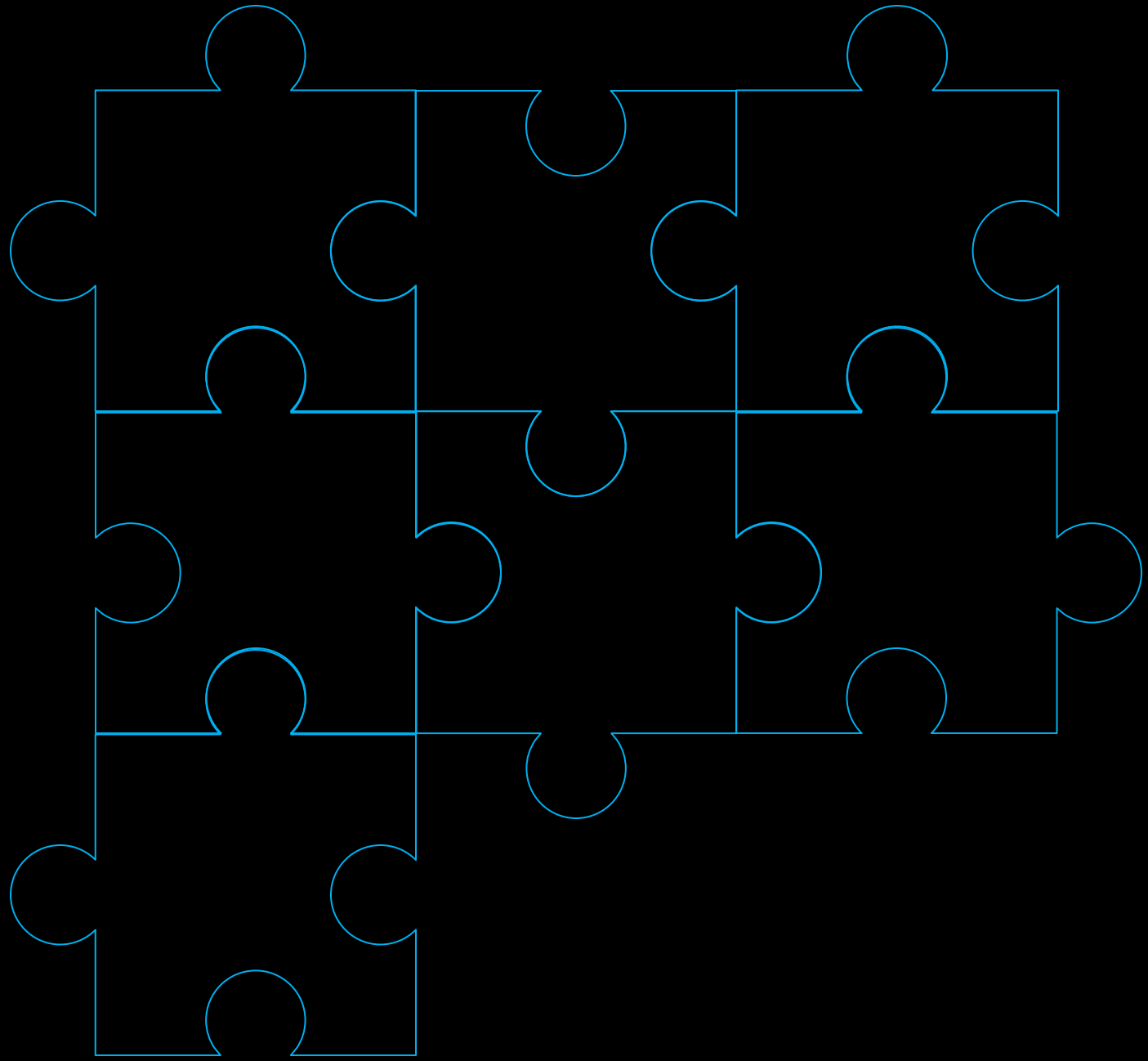


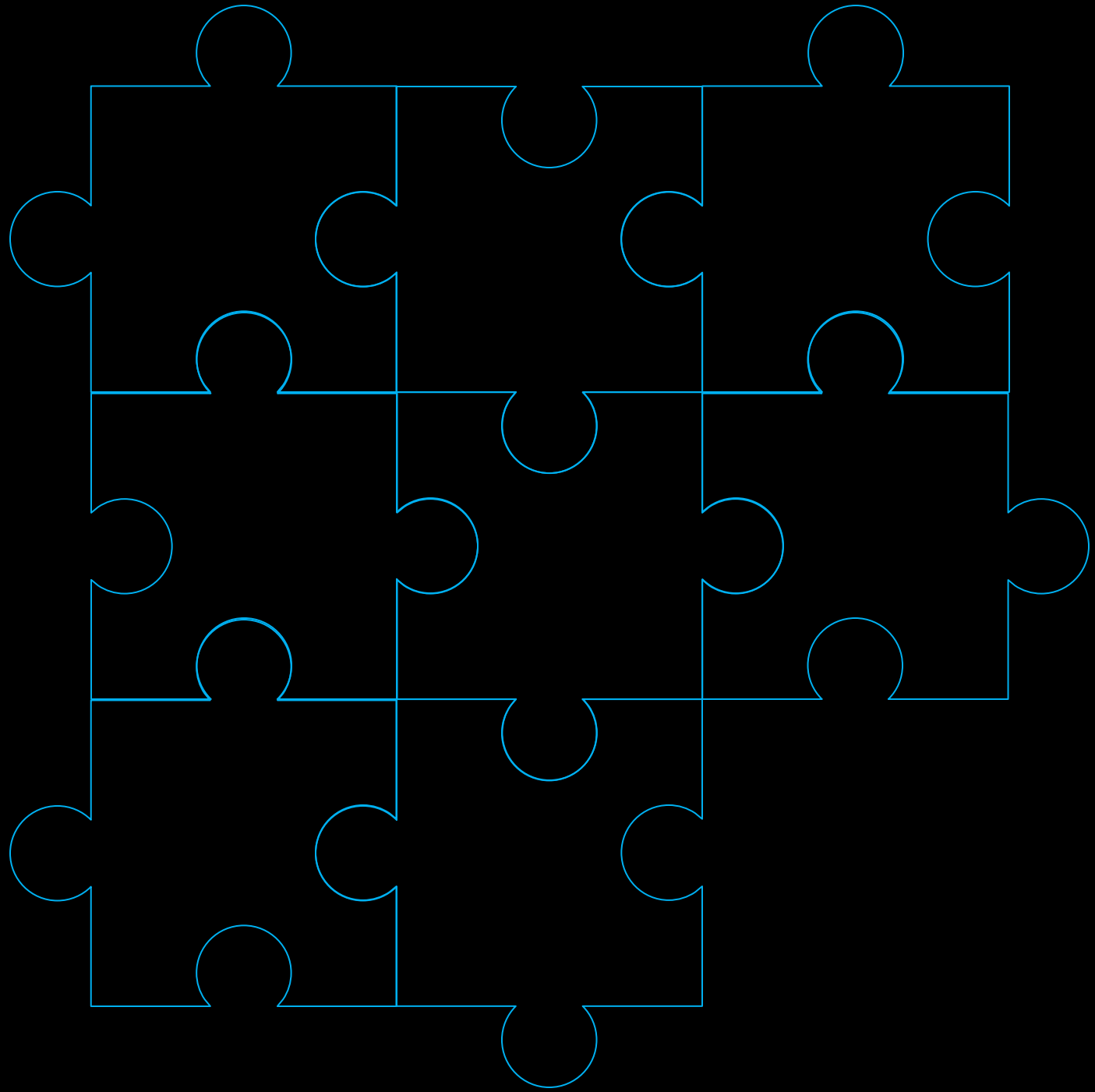


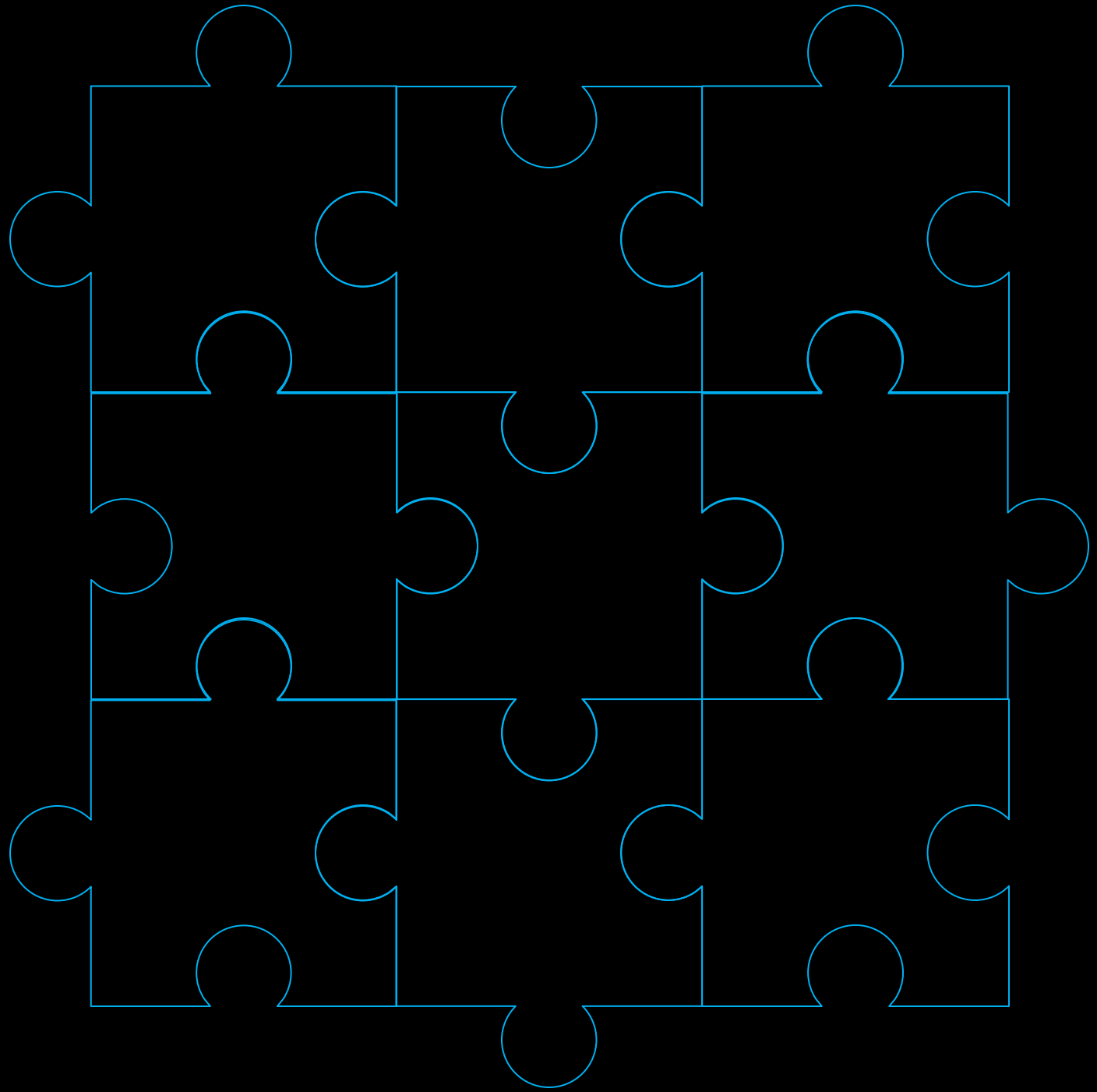


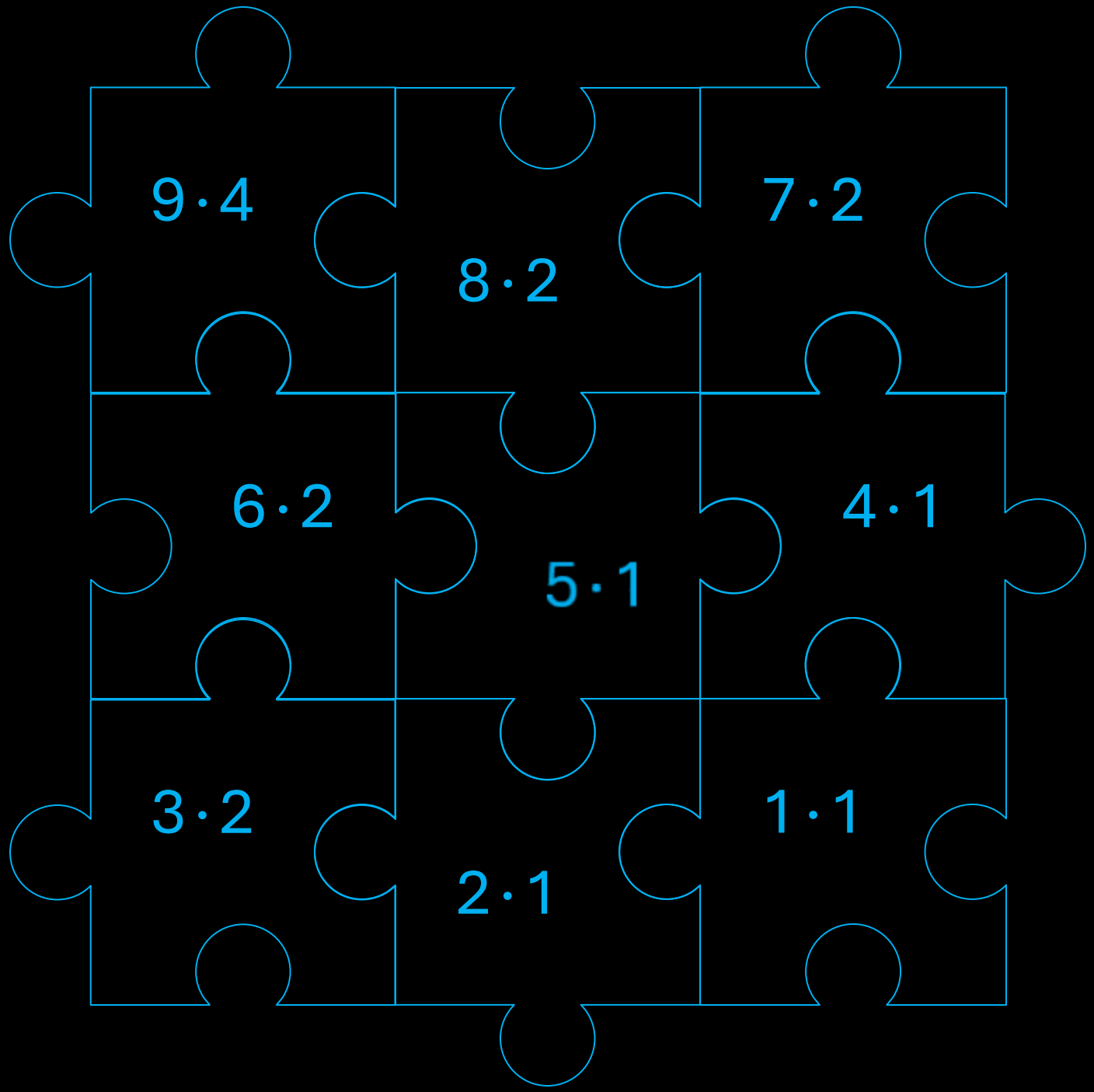











9·4
8·2
7·2
6·2
5·1
4·1
3·2
2·1
1·1

9·4·8·2·7·2·6·2·5·1·4·1·3·2·2·1·1·1

$$9 \cdot 4 \cdot 8 \cdot 2 \cdot 7 \cdot 2 \cdot 6 \cdot 2 \cdot 5 \cdot 1 \cdot 4 \cdot 1 \cdot 3 \cdot 2 \cdot 2 \cdot 1 \cdot 1 \cdot 1$$

$$=$$

$$23{,}224{,}320$$

```
public sealed class Reservation
{
    // ...
    public string Id { get; }
    public string At { get; }
    public string Email { get; }
    public string Name { get; }
    public int Quantity { get; }
    // ...
}
```

```
public sealed class Reservation
{
    // ...
    public Guid Id { get; }
    public DateTime At { get; }
    public string Email { get; }
    public string Name { get; }
    public int Quantity { get; }
    // ...
}
```

```
public sealed class Reservation
{
    // ...
    public Guid Id { get; }
    public DateTime At { get; }
    public Email Email { get; }
    public Name Name { get; }
    public int Quantity { get; }
    // ...
}
```

```
public sealed class Reservation
{
    // ...
    public Guid Id { get; }
    public DateTime At { get; }
    public Email Email { get; }
    public Name Name { get; }
    public NaturalNumber Quantity { get; }
    // ...
}
```

```
internal interface IPeriodVisitor<T>
{
    T VisitYear(int year);
    T VisitMonth(int year, int month);
    T VisitDay(int year, int month, int day);
}
```

```
  Failure (Endo $ \x -> x { inputName = Nothing }, ["no bob and toms all
validateName _ = Failure (mempty, ["name is required"])

validateDoB :: Day -> Input -> Validation (Endo Input, [String]) Day
validateDoB now (Input _ (Just dob) _) | addGregorianYearsRollOver (-12)
  Success dob
validateDoB _ (Input _ (Just _) _) =
  Failure (Endo $ \x -> x { inputDoB = Nothing }, ["get off my lawn"])
validateDoB _ _ = Failure (mempty, ["dob is required"])

validateAddress :: Monoid a => Input -> Validation (a, [String]) String
validateAddress (Input _ _ (Just a)) = Success a
validateAddress _ = Failure (mempty, ["add1 is required"])

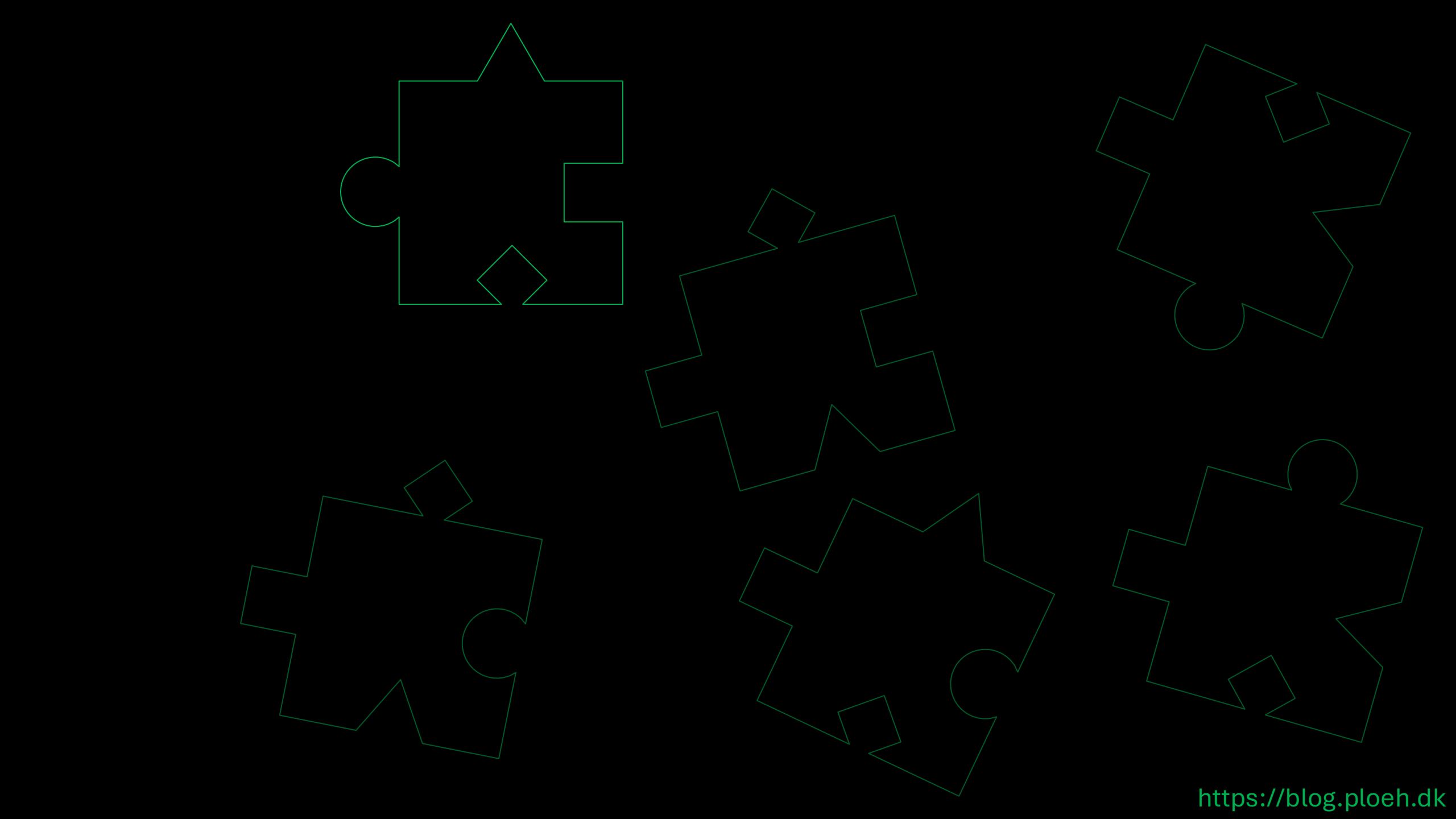
validateInput :: Day -> Input -> Either (Input, [String]) ValidInput
validateInput now args =
  toEither $
  first (first (`appEndo` args)) $
```

```haskell
logClock :: (MonadIO m)
         => (forall x. ClockInstruction (m x) -> m x)
         -> ClockInstruction (m a) -> m a
logClock inner (CurrentTime next) = do
  output <- inner $ CurrentTime return
  liftIO $ writeLogEntry "CurrentTime" () output
  next output
```

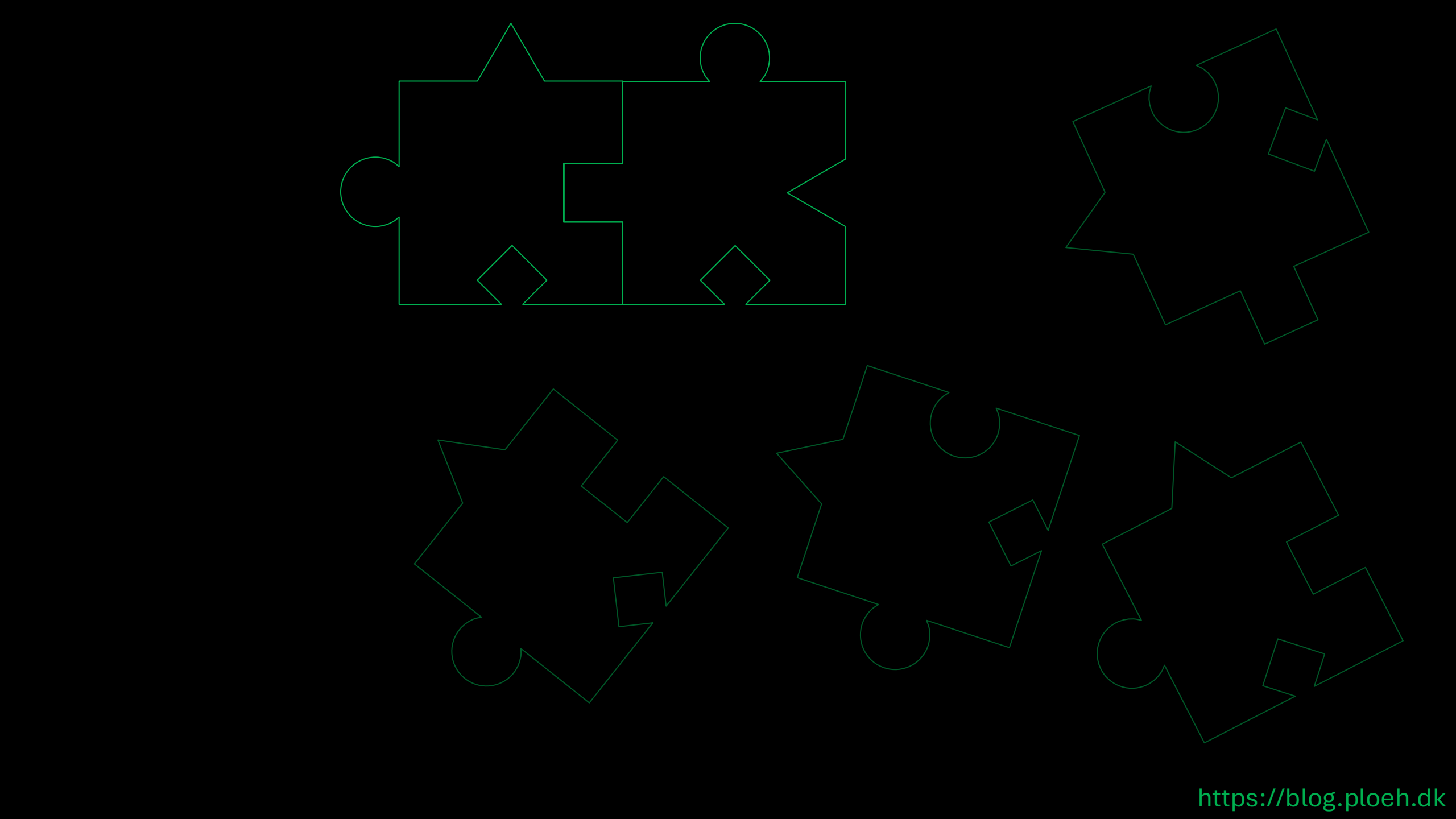
```haskell
sumGrades ls =
  liftA2 (,) NE.head length <$> NE.group
    (sort (uncurry (+) . unPair . fmap (grade . (ls !!)) . Pair <$>
      choices (indices ls)))
```

```
"Matching passwords" ~: do
  pw <- ["password", "12345678", "abcdefgh"]
  let actual = comparePasswords pw pw
  return $ Right pw ~=? actual
```

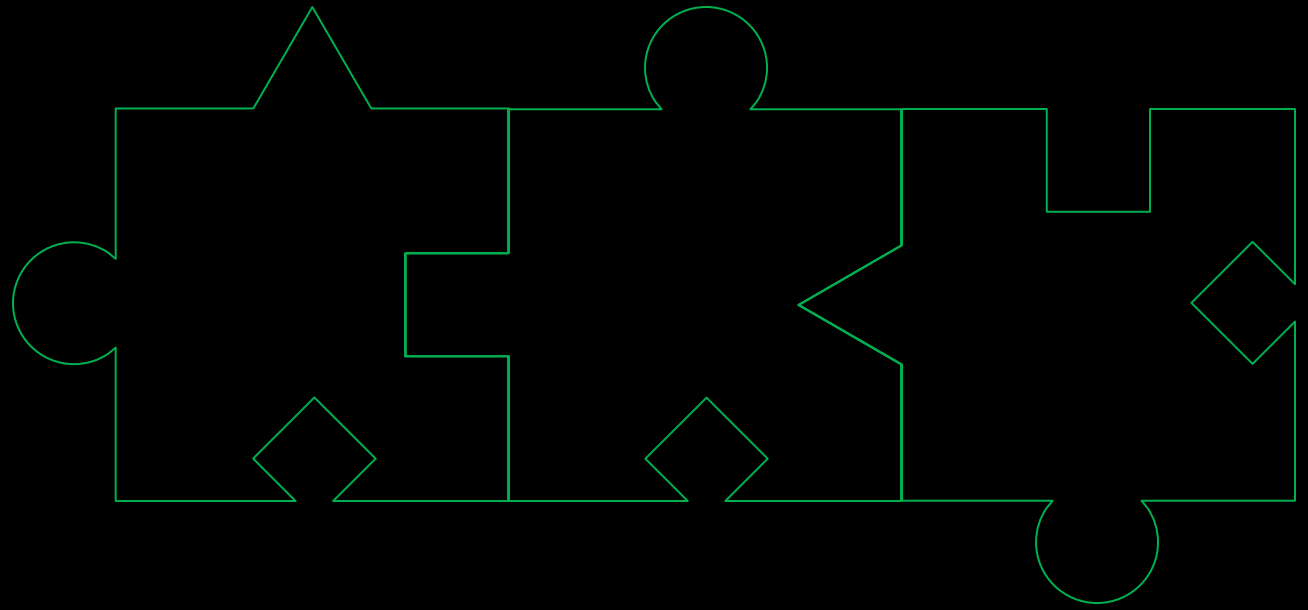
A representation of composition
with strongly typed code

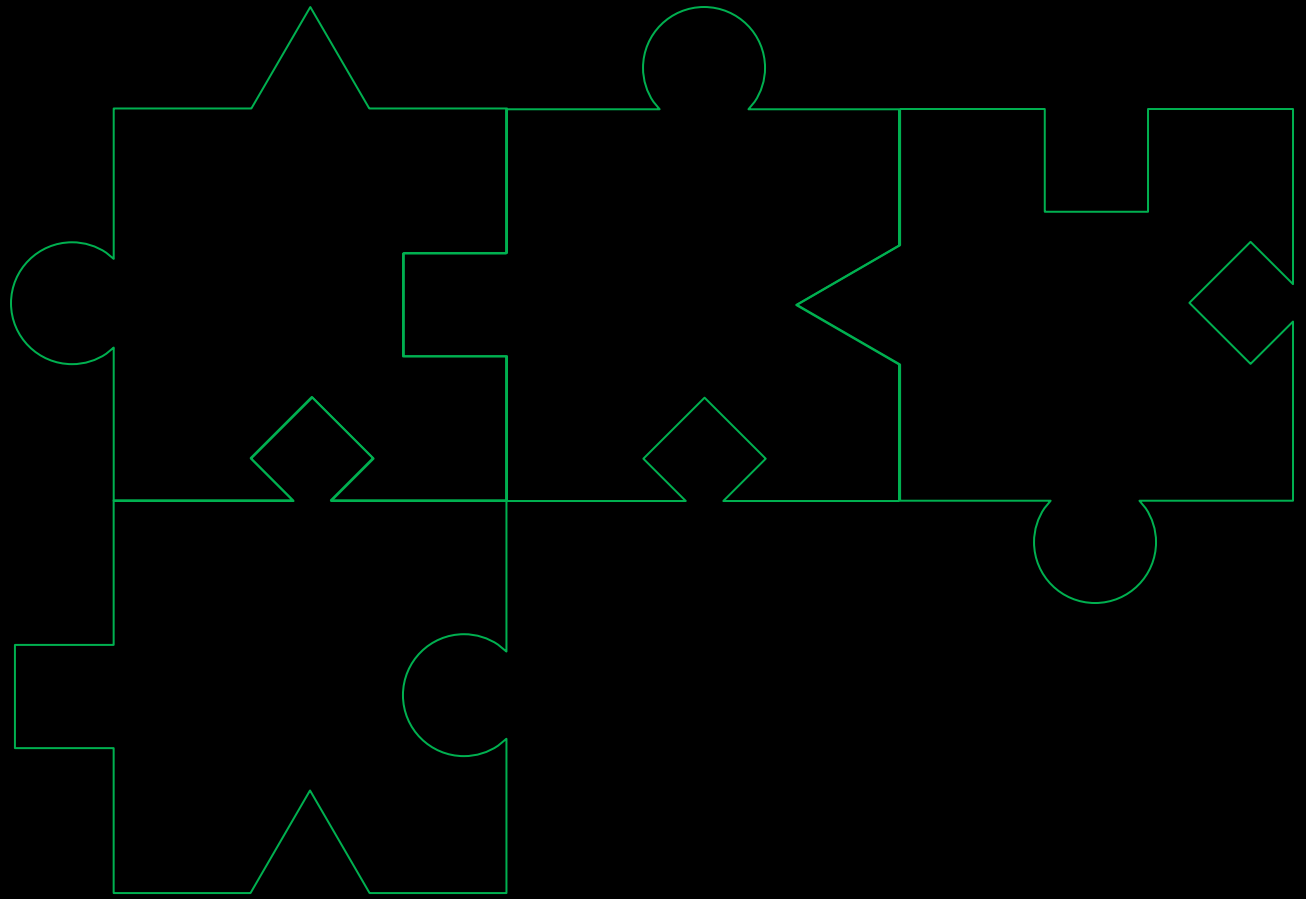


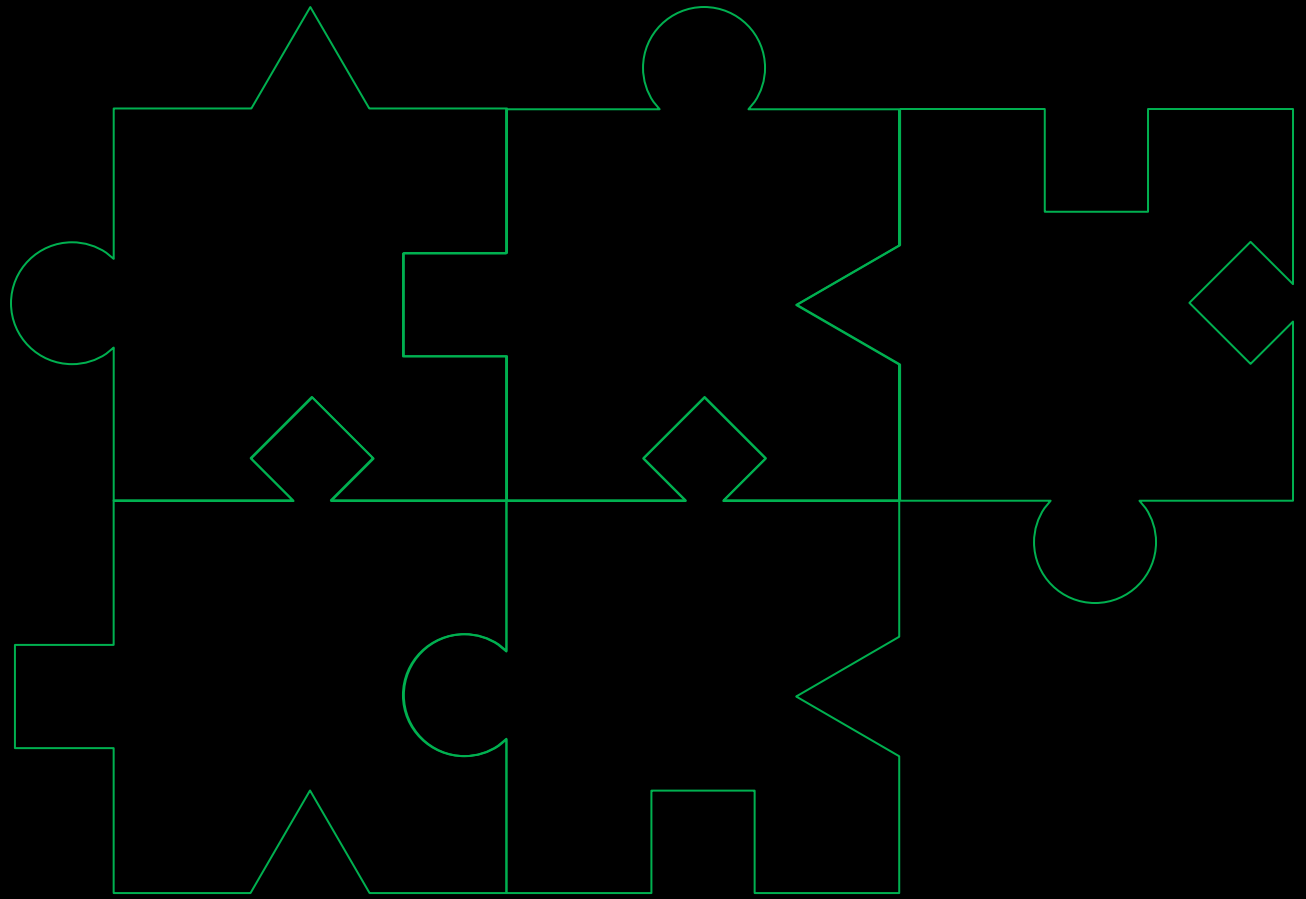
https://blog.ploeh.dk

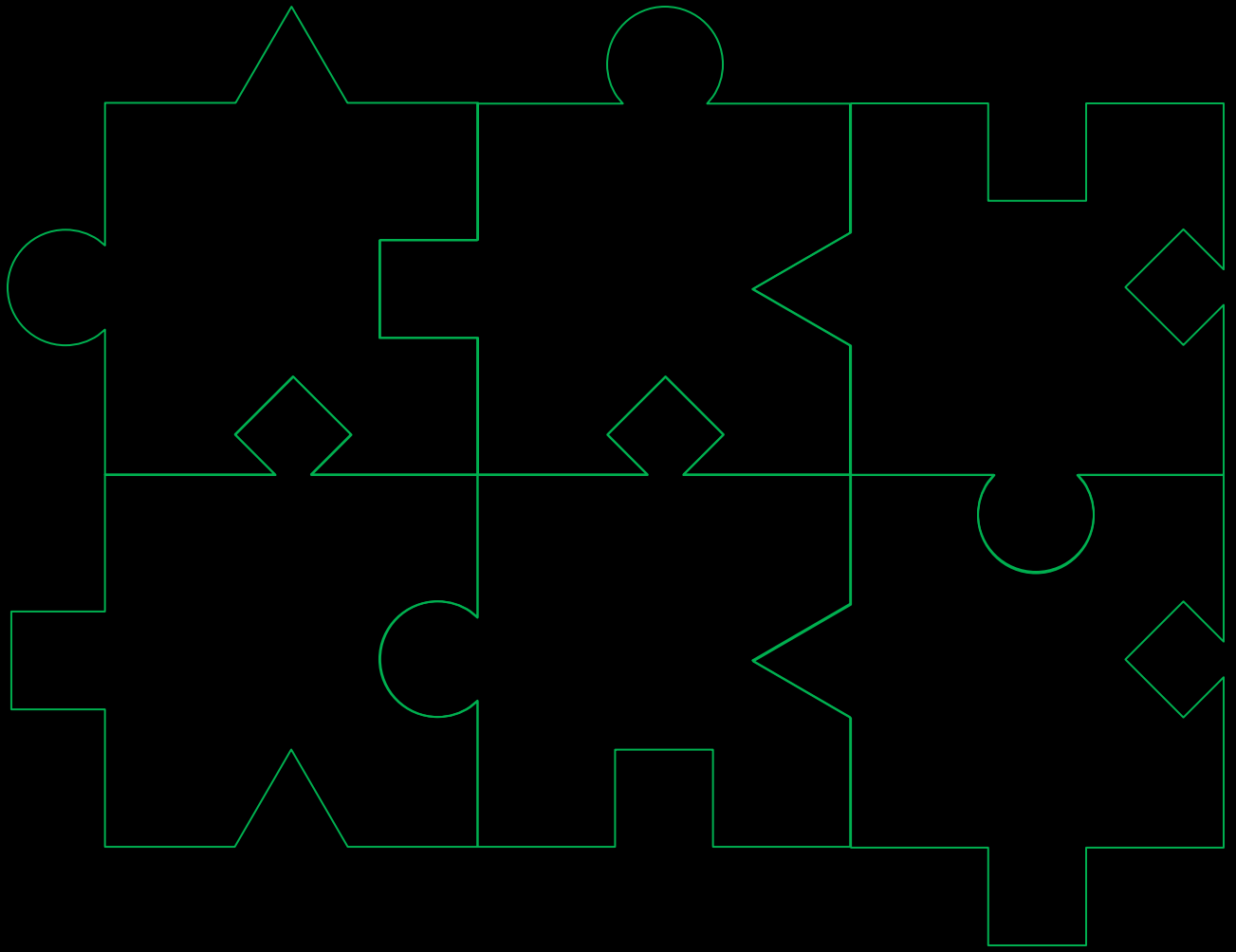


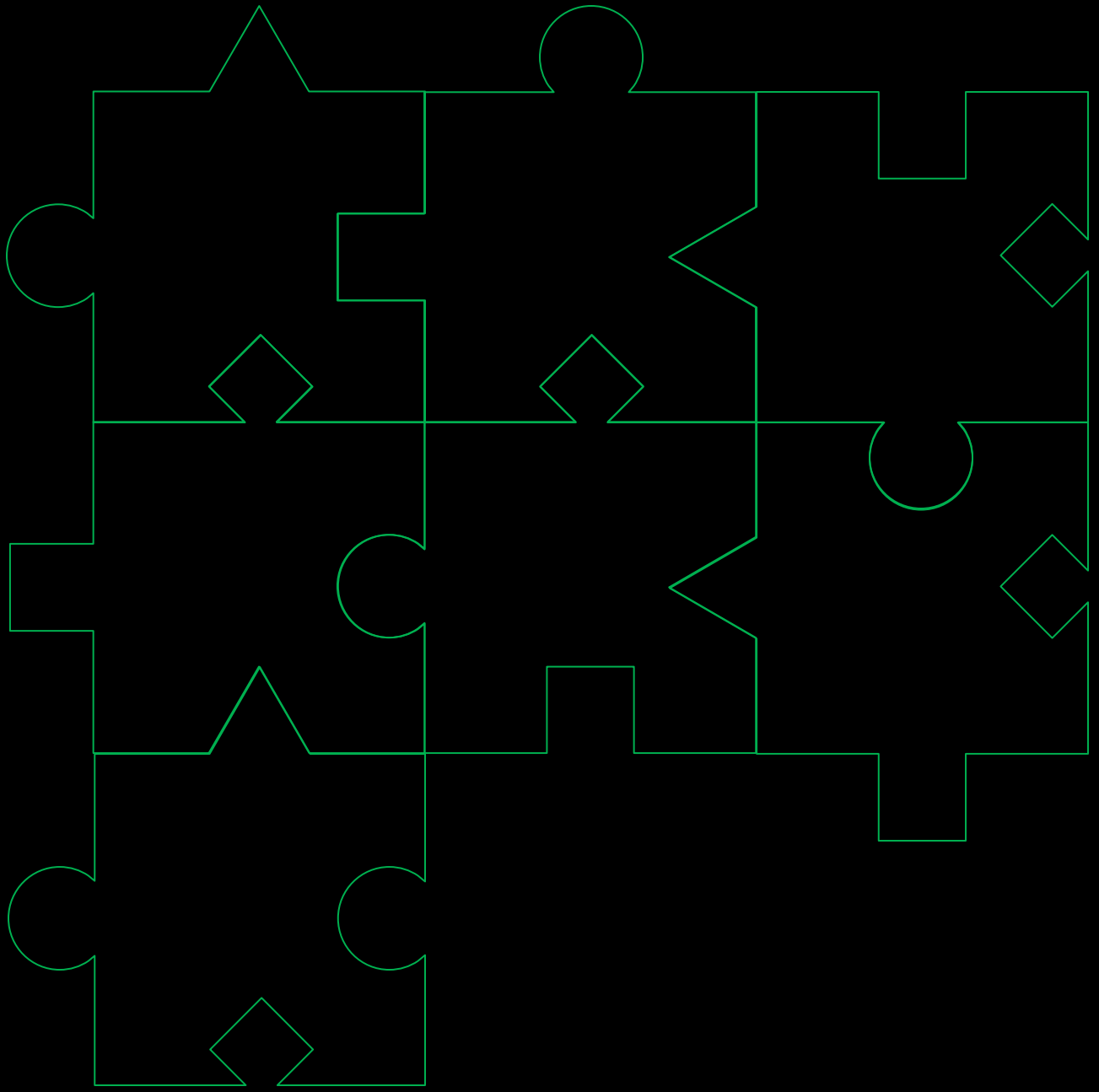
https://blog.ploeh.dk



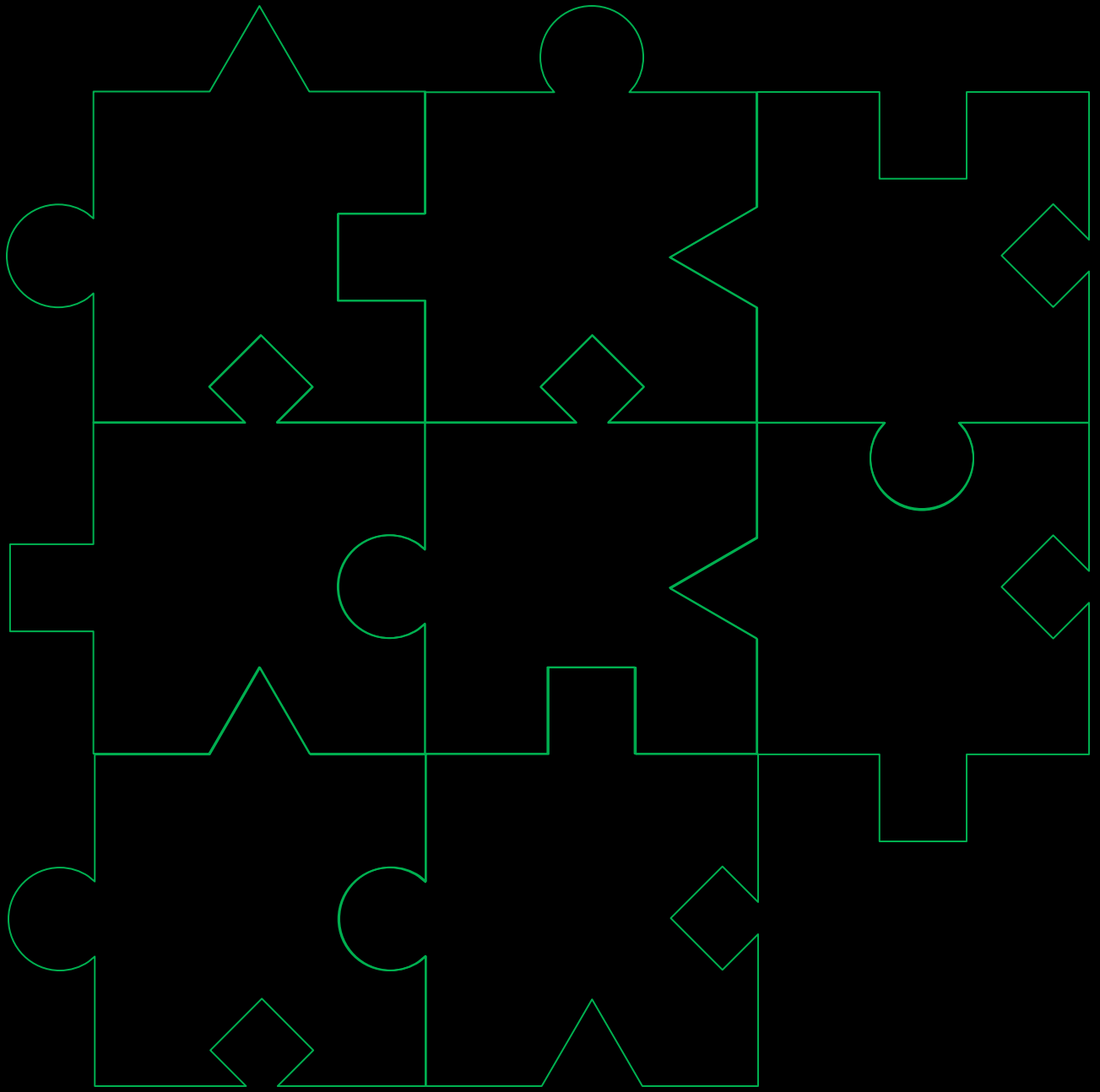




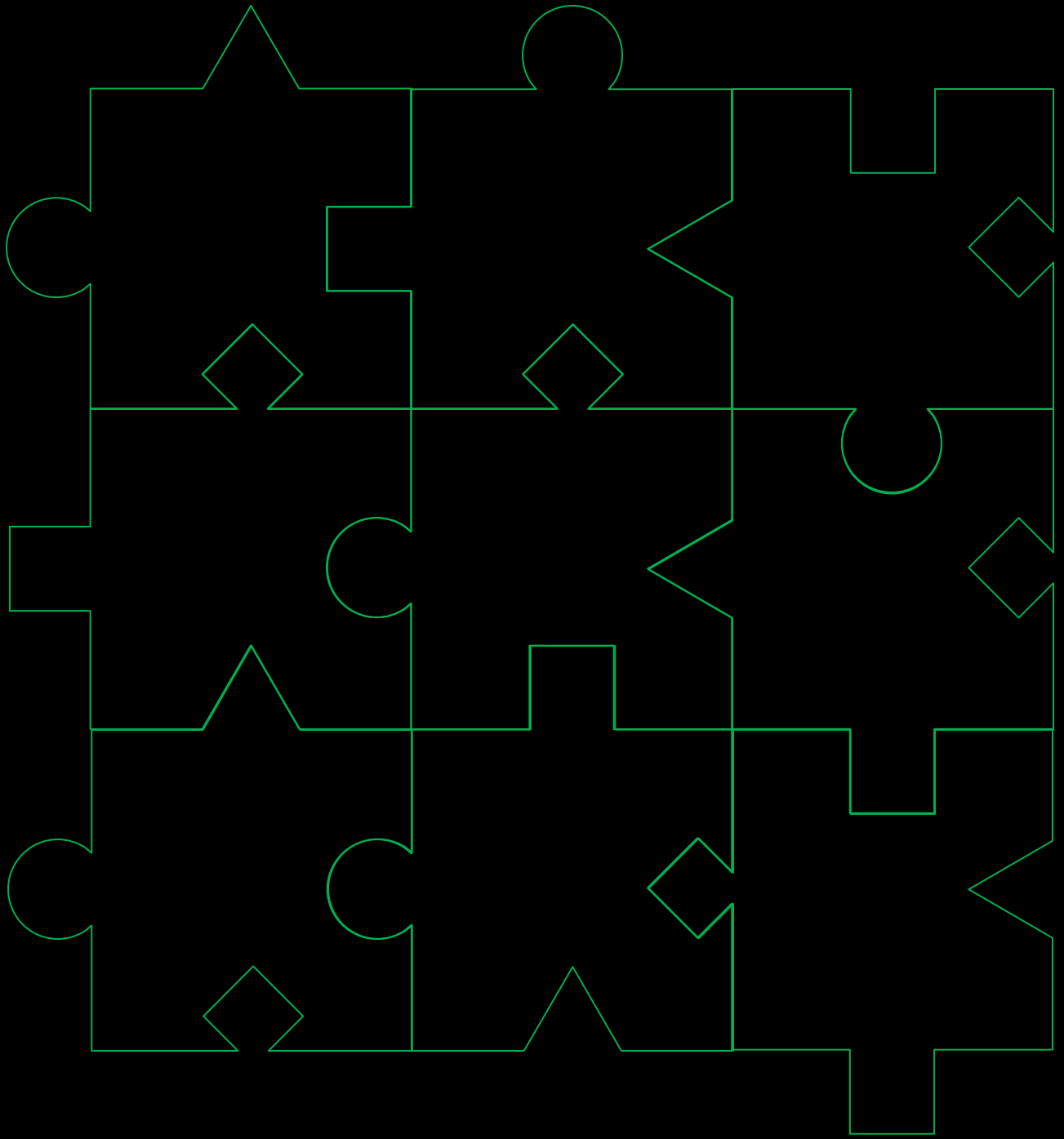




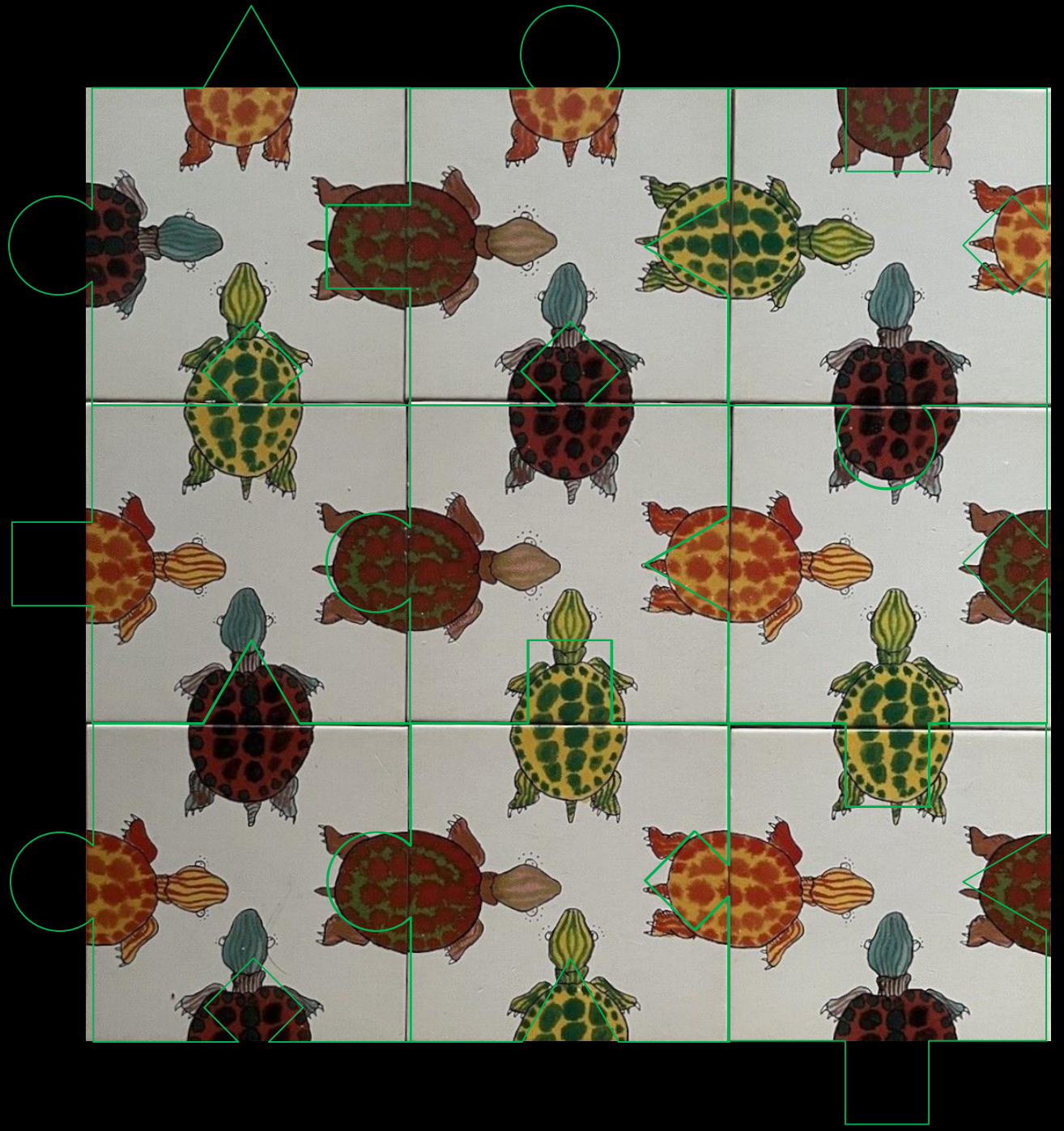
https://blog.ploeh.dk



https://blog.ploeh.dk



https://blog.ploeh.dk



https://blog.ploeh.dk

Static types

Not for correctness

Static types

Not (only) for correctness

Much more for discoverability

Hoogle

(String -> IO Int) -> [String] -> IO [Int]

set:included-with-ghc

Search

Packages

- is:exact
- base

:: (String -> IO Int) -> [String] -> IO [Int] set:included-with-ghc

mapM :: Monad m => (a -> m b) -> [a] -> m [b]

base GHC.Base

`mapM f` is equivalent to `sequence . map f`.

traverse :: (Traversable t, Applicative f) => (a -> f b) -> t a -> f (t b)

base Prelude Data.Traversable

Map each element of a structure to an action, evaluate these actions from left to right, and collect the results. For a version

mapM :: (Traversable t, Monad m) => (a -> m b) -> t a -> m (t b)

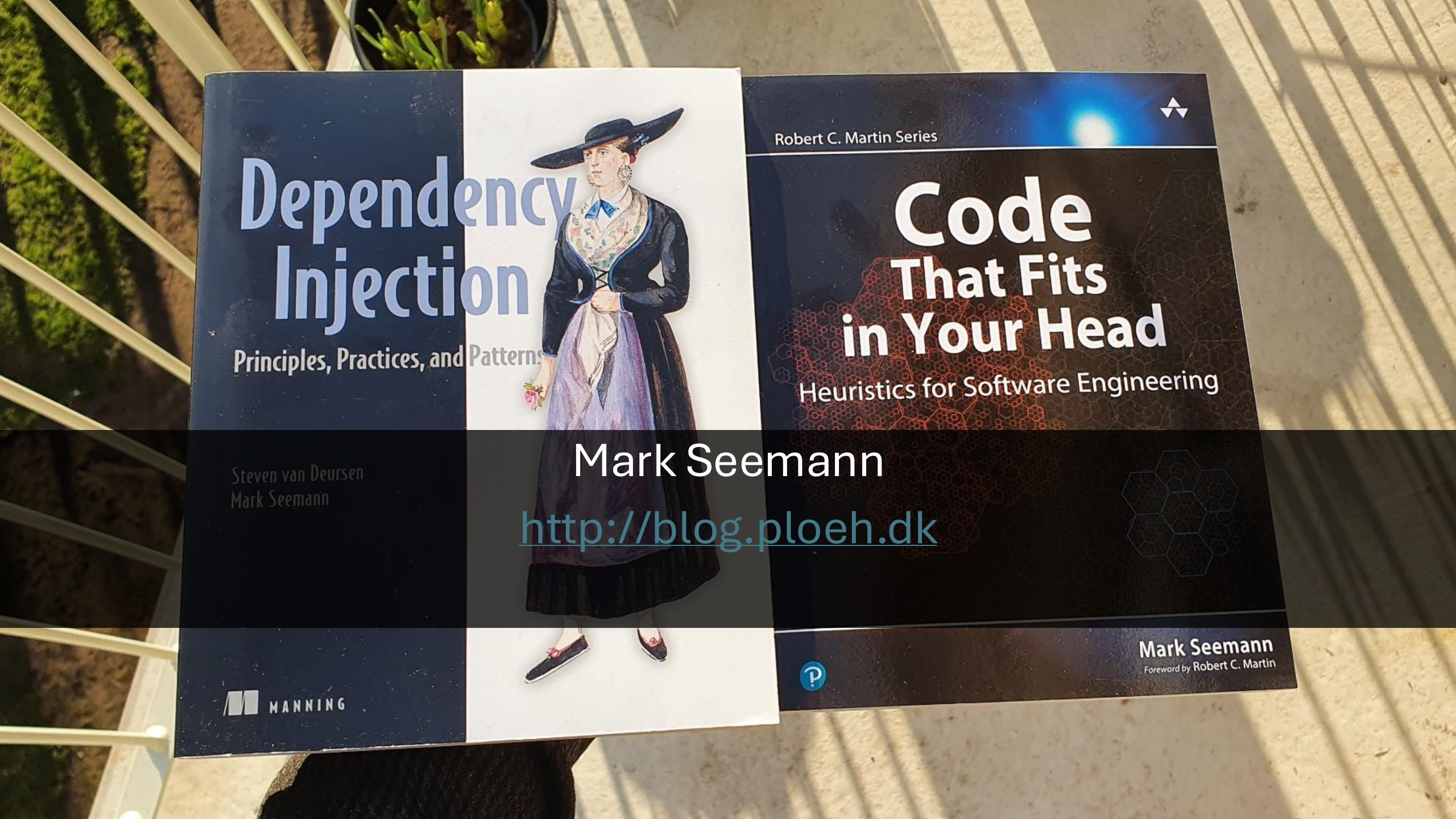
base Prelude Control.Monad Data.Traversable

Map each element of a structure to a monadic action, evaluate these actions from left to right, and collect the results. For a power of Monad.

Static types

Not (only) for correctness

Much more for discoverability



Mark Seemann

http://blog.ploeh.dk