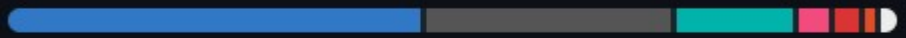
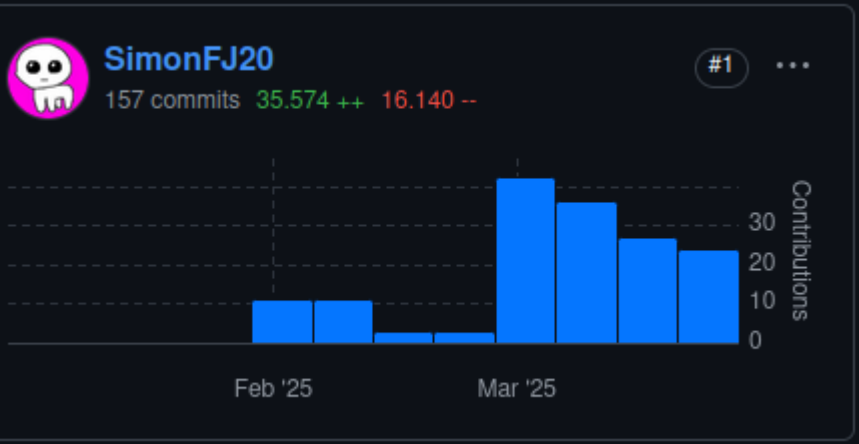
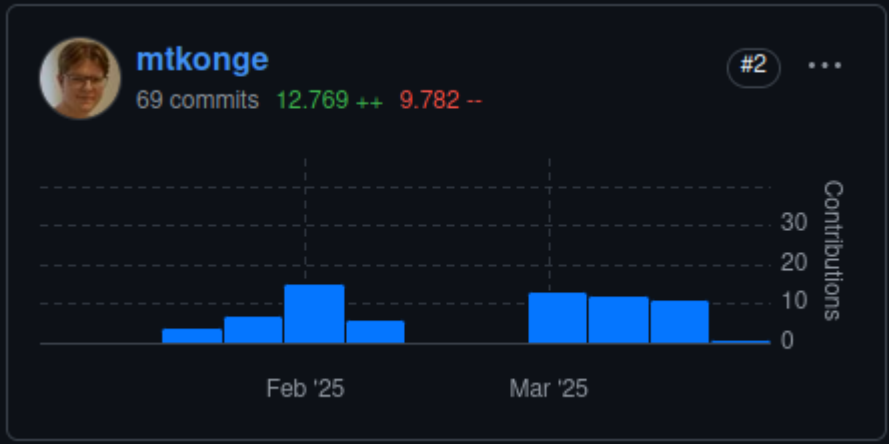
Mobil-, backend- og compiler-udvikling med **Flutter**, **C** og **Slige**

Languages



- **TypeScript** 48.4%
- **C** 28.5%
- **Dart** 13.8%
- **C++** 3.5%
- **CMake** 2.8%
- **HTML** 1.1%
- **Other** 1.9%





Language	Files	Lines	Code	Comments	Blanks
TypeScript	48	11229	10246	57	926
C	22	5475	4806	28	641
Dart	40	3077	2792	2	283
C Header	29	1467	1079	146	242
C++	8	625	437	66	122
CMake	8	545	348	113	84
JavaScript	1	198	175	0	23
JSON	4	127	127	0	0
HTML	2	143	120	15	8
Makefile	3	151	90	20	41
XML	7	121	68	51	2
SQL	1	82	66	0	16
Swift	6	77	55	7	15
YAML	4	164	52	91	21
Python	1	44	32	1	11
Shell	4	51	25	5	21
Xcode Config	8	35	24	8	3
Kotlin	1	5	3	0	2
Markdown	9	489	0	314	175
Plain Text	2	23	0	21	2
Total	208	24128	20545	945	2638

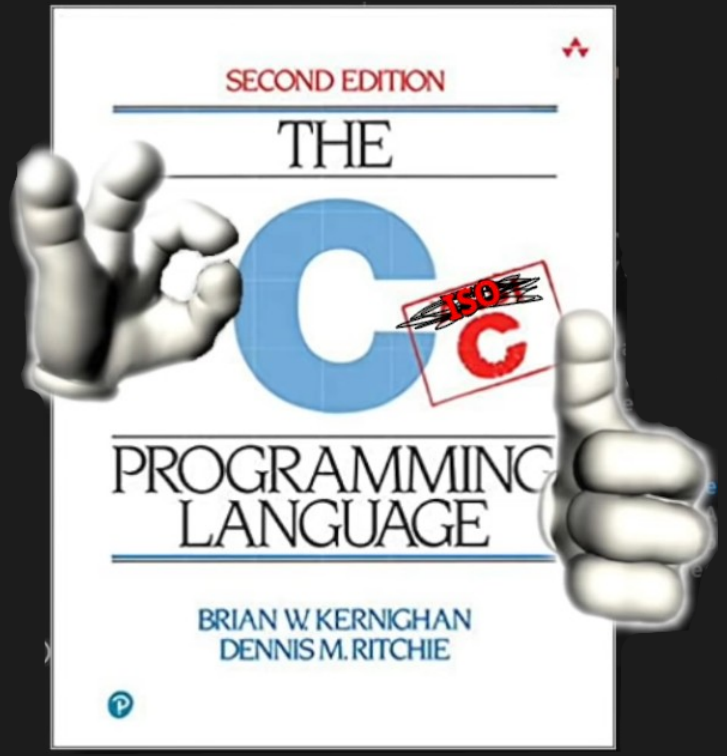
Disposition

- Arbejdsprocess
- **Tech** Stack
- Design-patterns i **Flutter**
- Design-patterns i **Fresh Plaza**
- HTTP-backend i **C**
- **CI** og **Deployment**
- Compiler-udvikling og **Slige**

Arbejdsprocess

- Kravspecifikation
- User stories
- Kunde
- eXtreme programming
- Scrum
- Udvikling af udviklingsprocess

Tech Stack



Flutter

Flutter

- Widget tree
- vs Swift
  - Cross platform vs native
  - Caching-strategier vs hurtig afviklingstid
- vs Maui
  - Abstraktion af Widget tree
  - State-management og data-binding
- Tooling
- Økosystem

Design-patterns i **Flutter**

State management

Mobilapp

App : StatelessWdgt

→ Build() ⇒

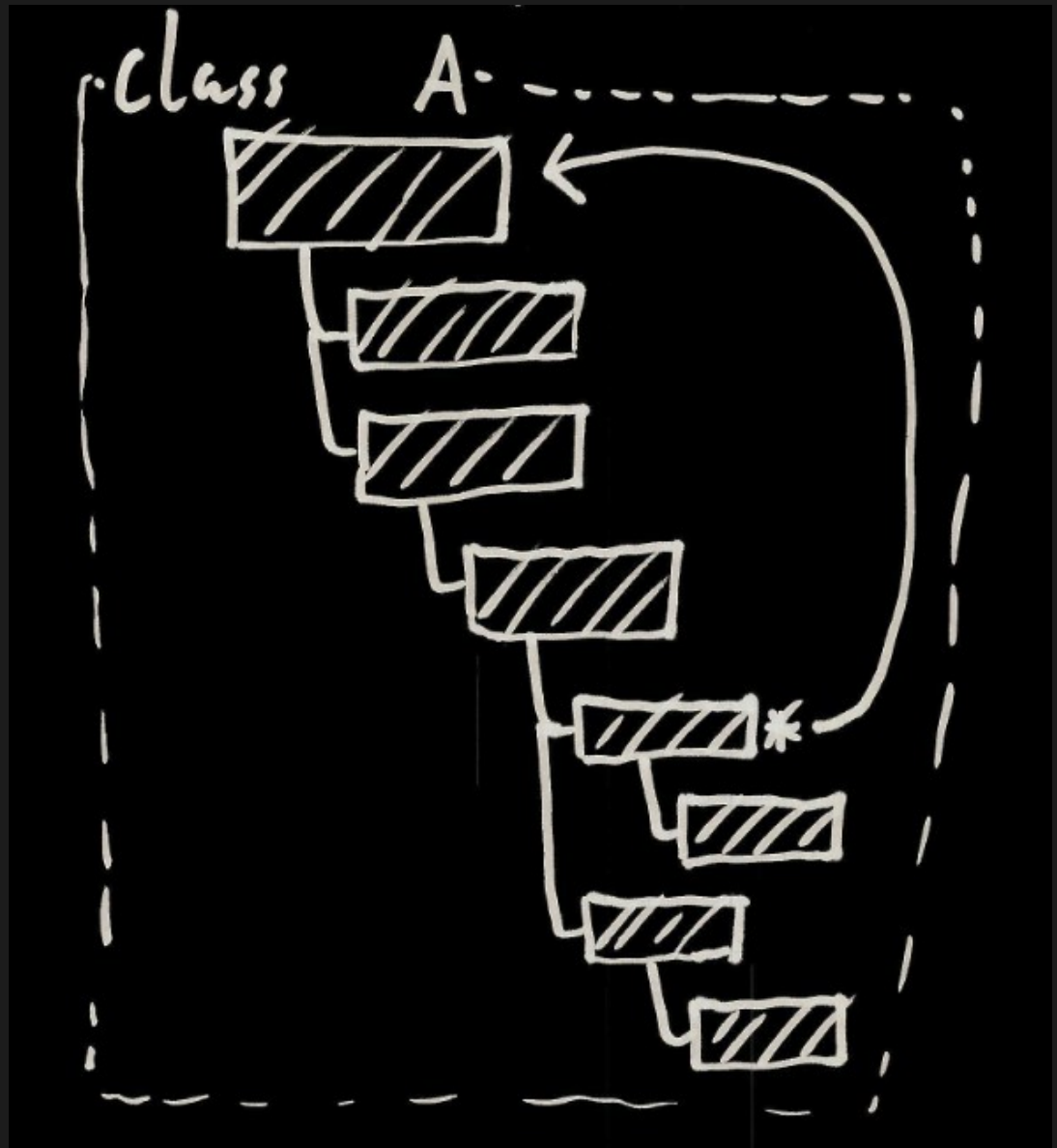
Column

Text(...)

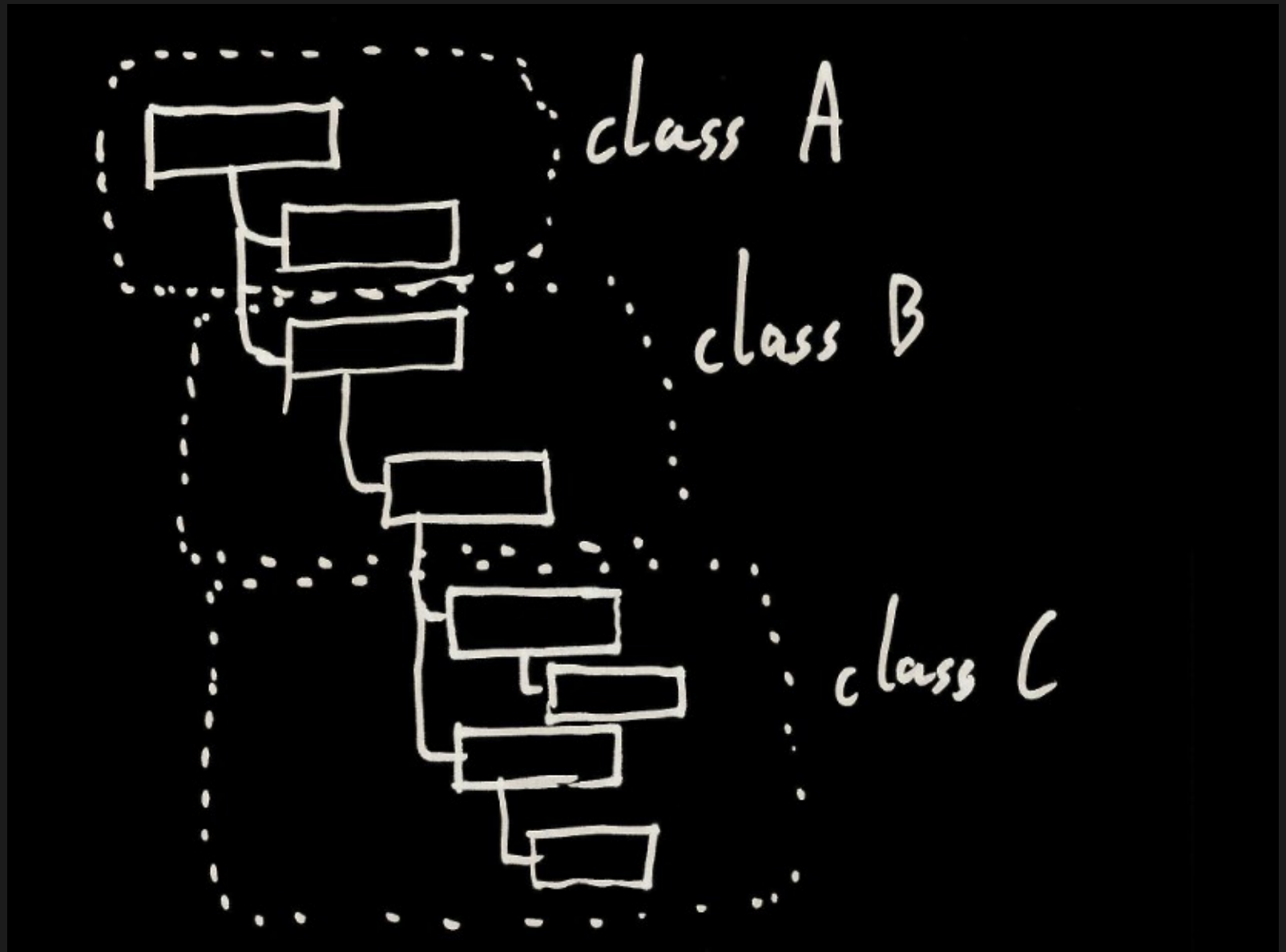
Button(...)

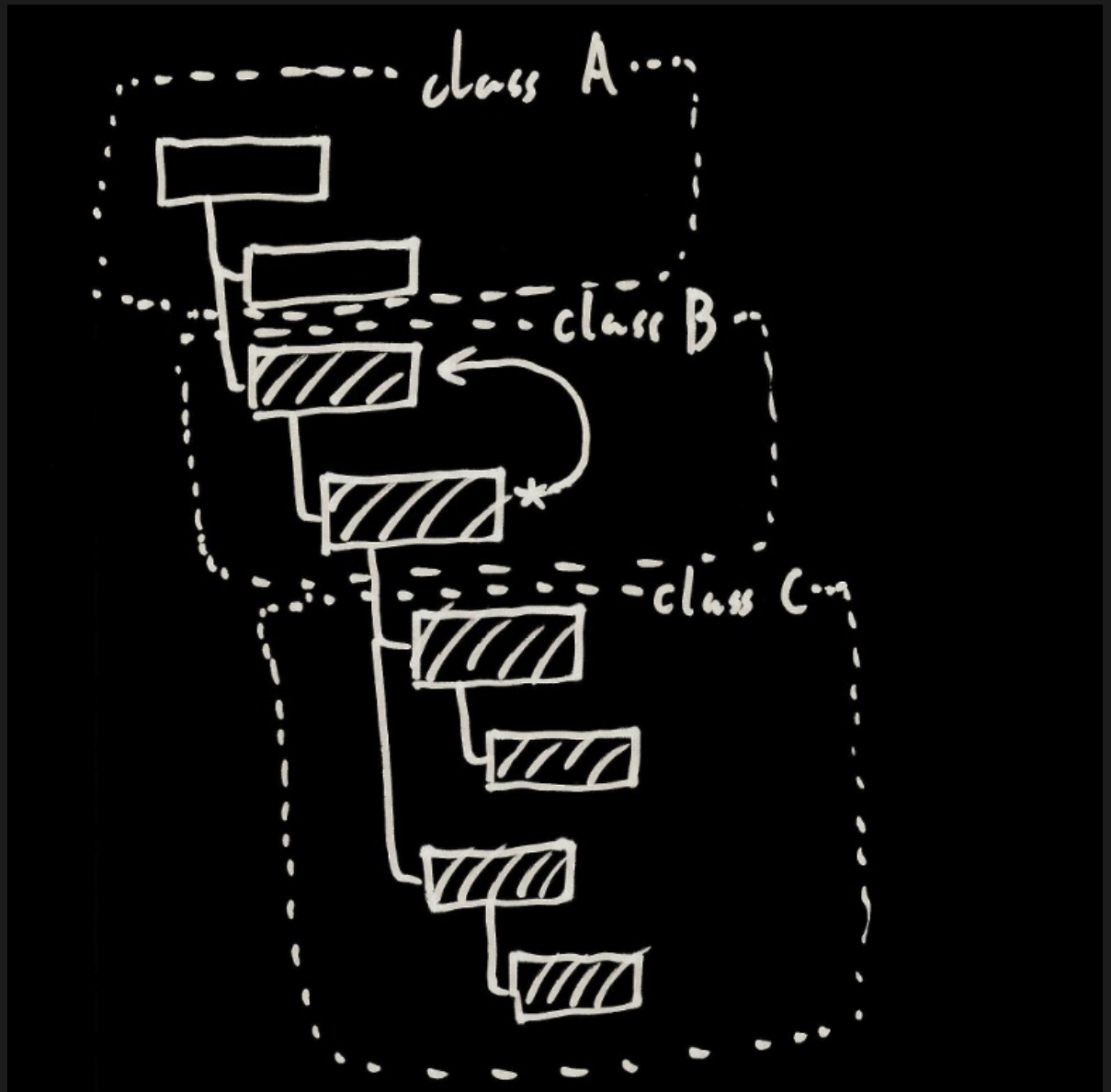
State-management med **setState()**

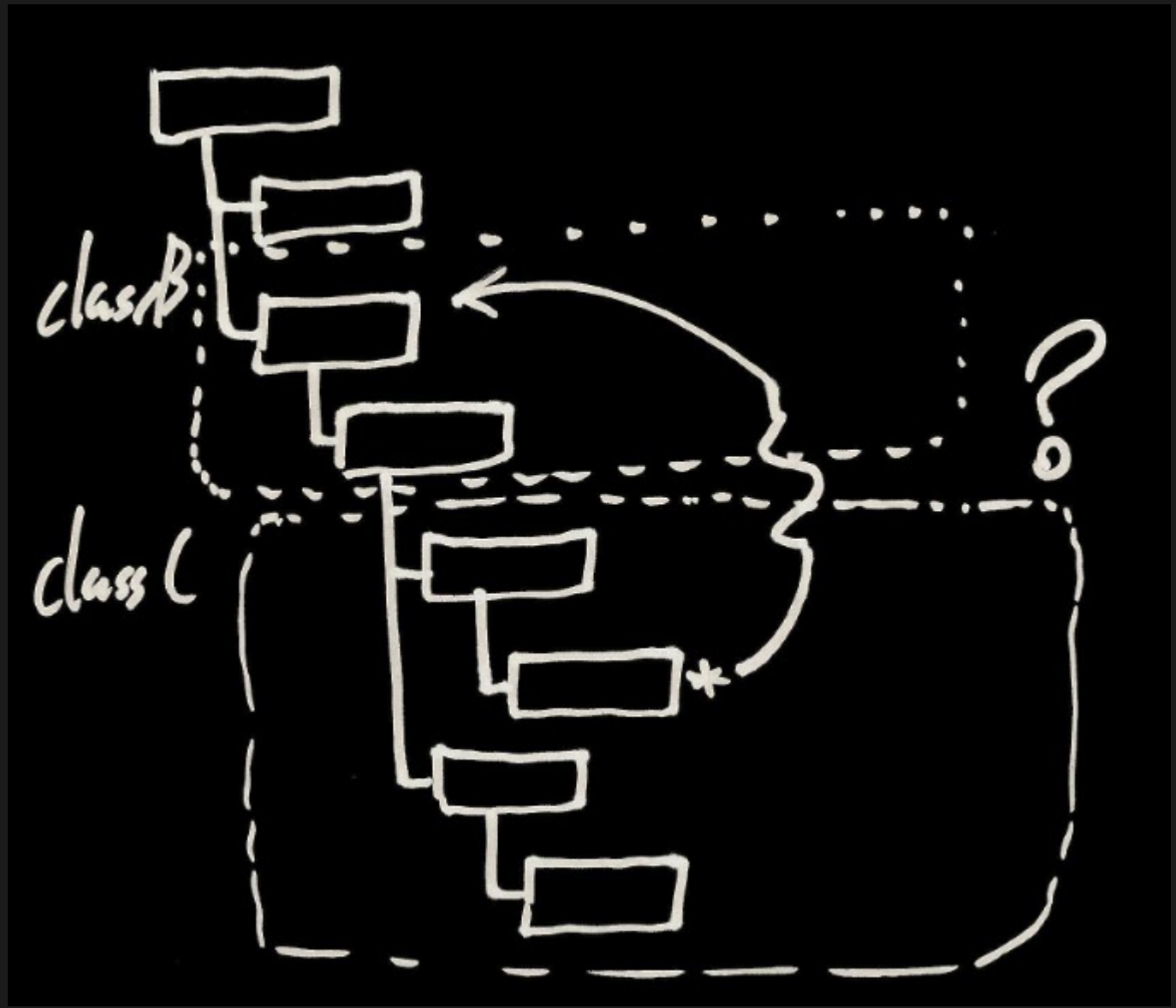
```
App : StatefulWidget
  - counter :: int
  → build() ⇒
      Column
        Text("$counter ...")
        Button(
          click: () ⇒ setState(() {
                       counter += 1;
          }),
        )
```


class A
*

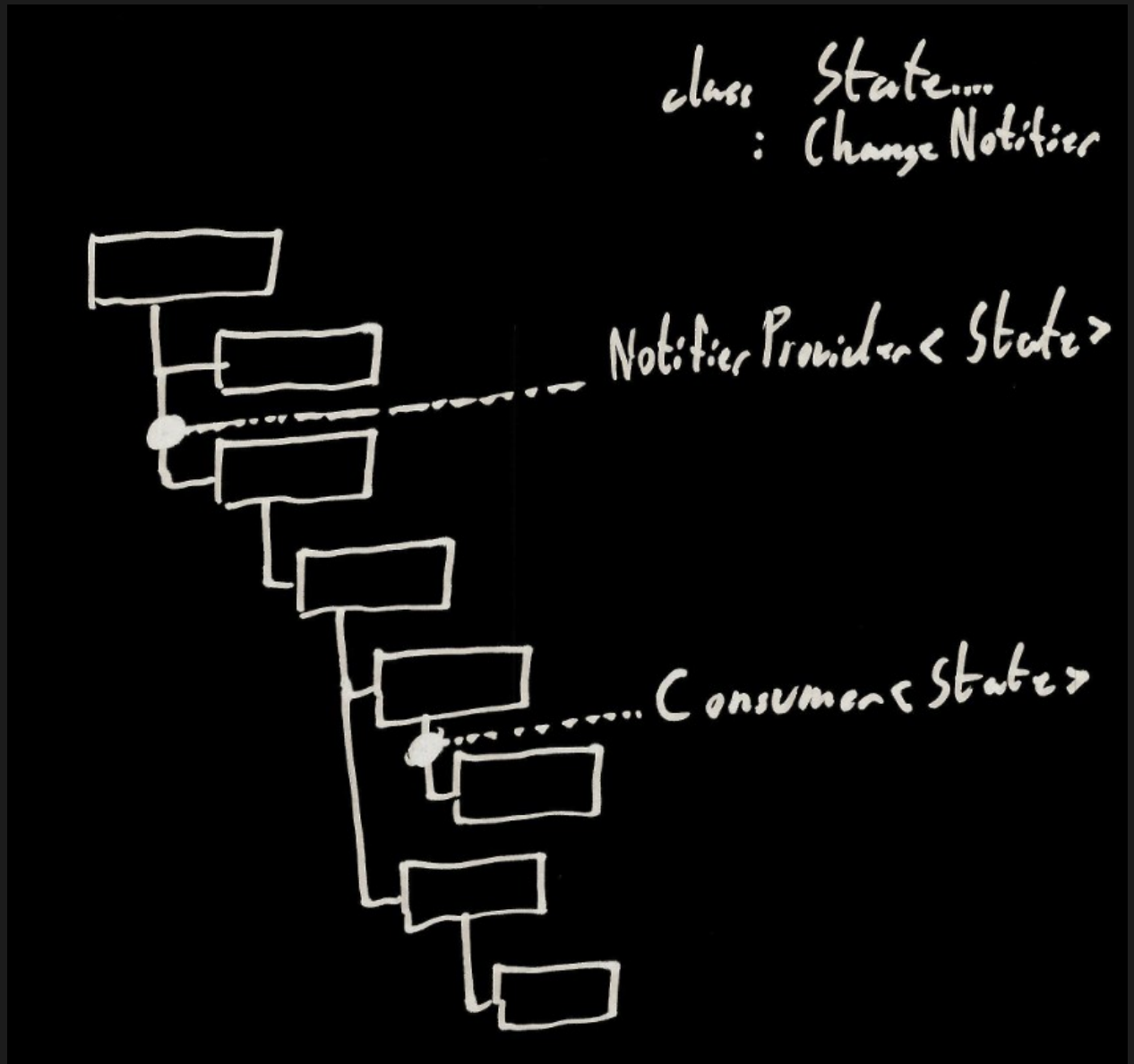
Opdeling af widget-klasser


class A
class B
class C

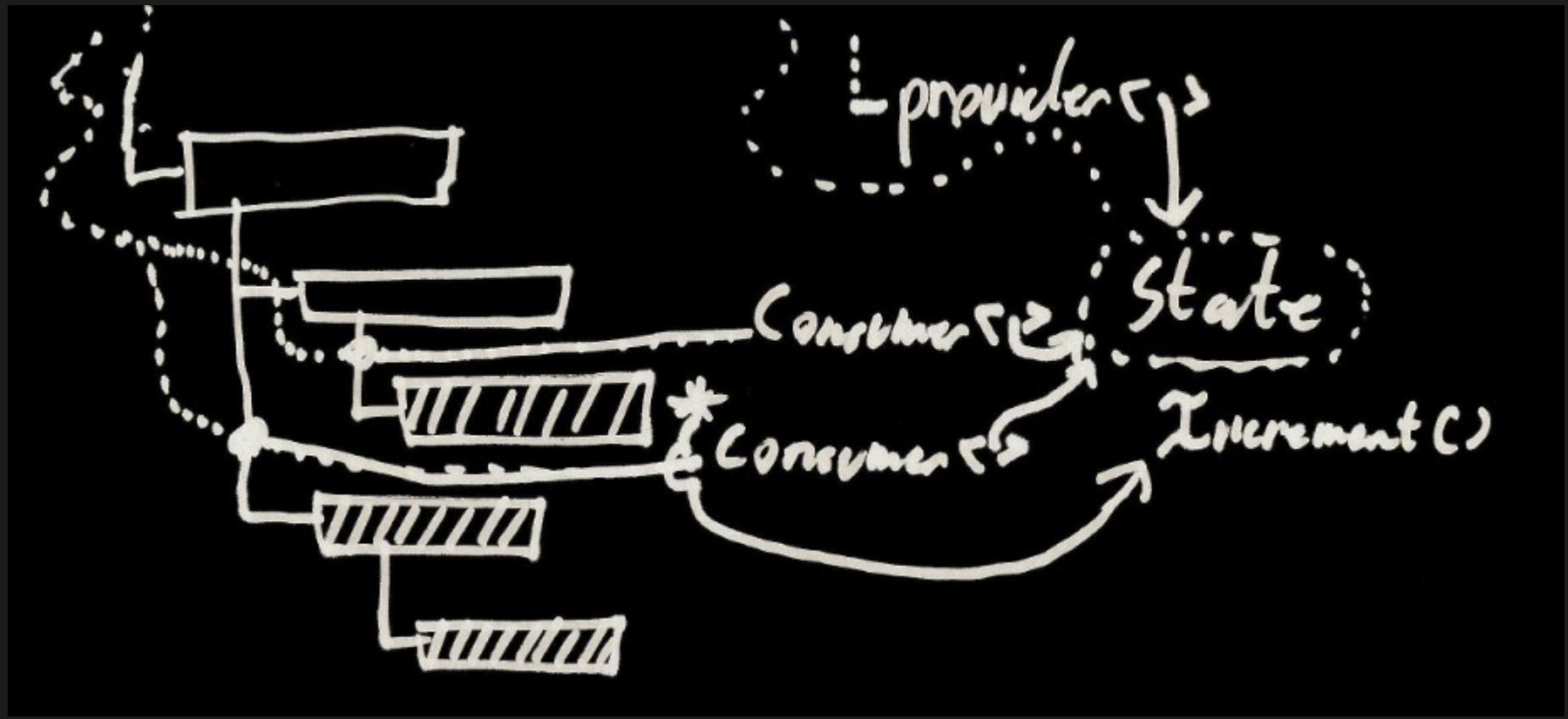

class A
class B
class C

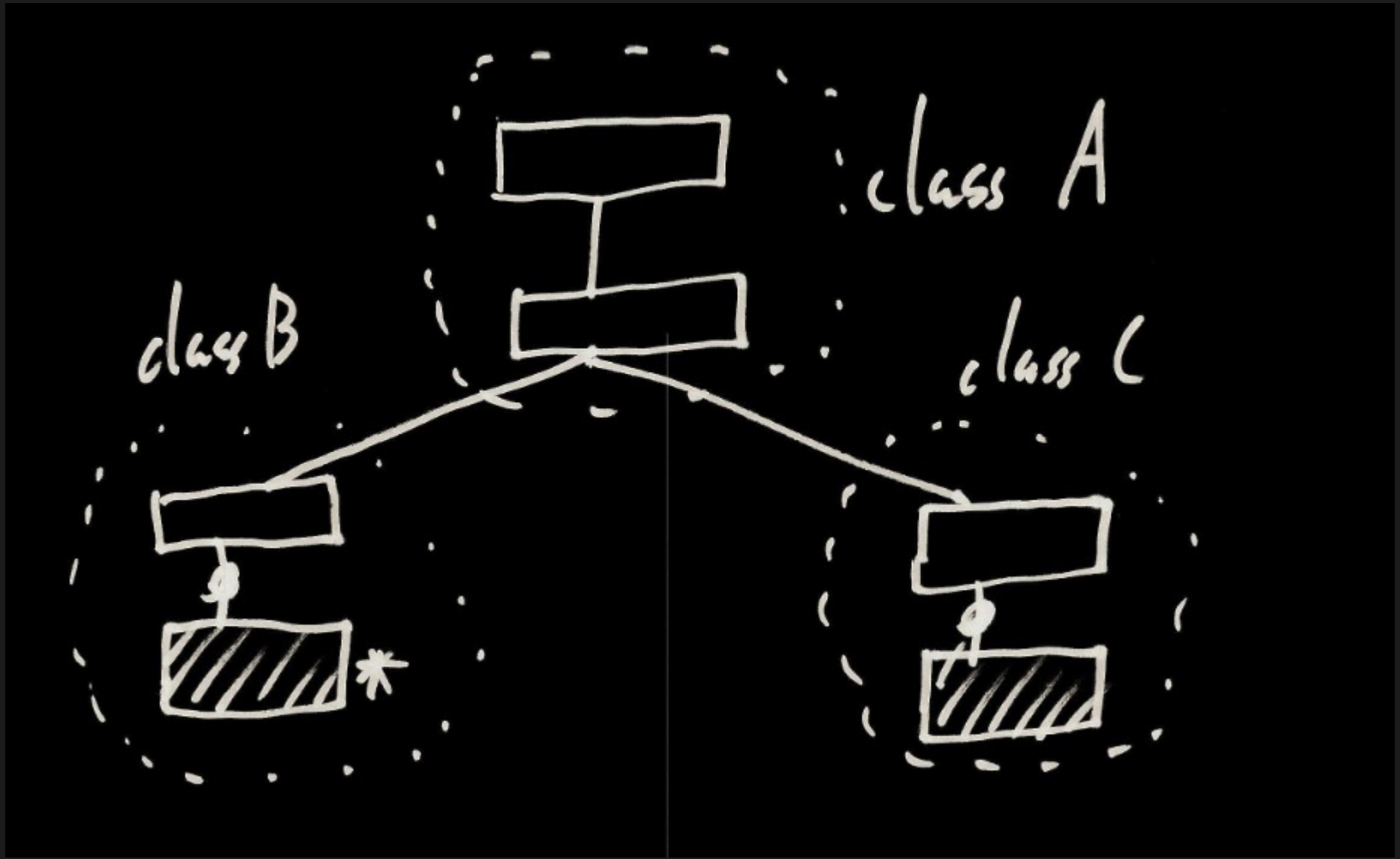

class B
class C
?

State-management med **ChangeNotifierProvider** & **Consumer**


class State...
: ChangeNotifier
NotifierProvider<State>
Consumer<State>

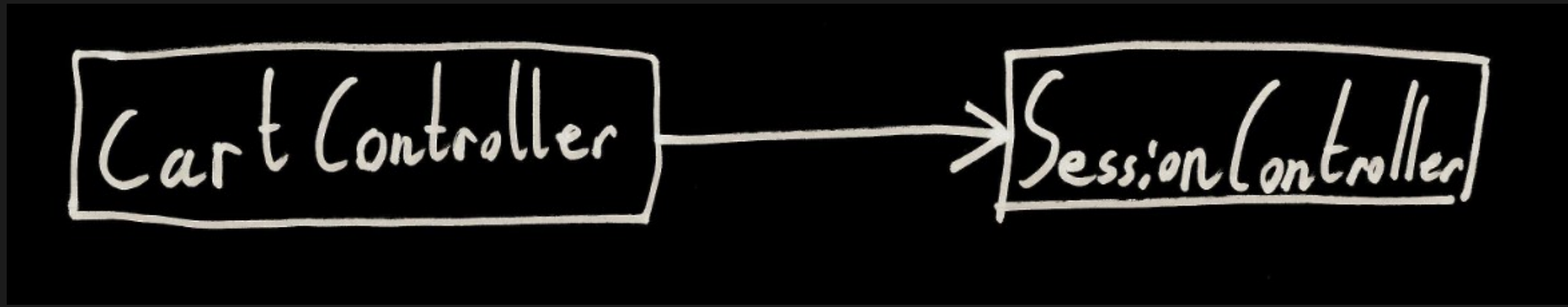
```
class CookieState : ChangeNotifier
  - counter :: int

  → Increment() {
      counter += 1;
      notifyListeners();
    }
```

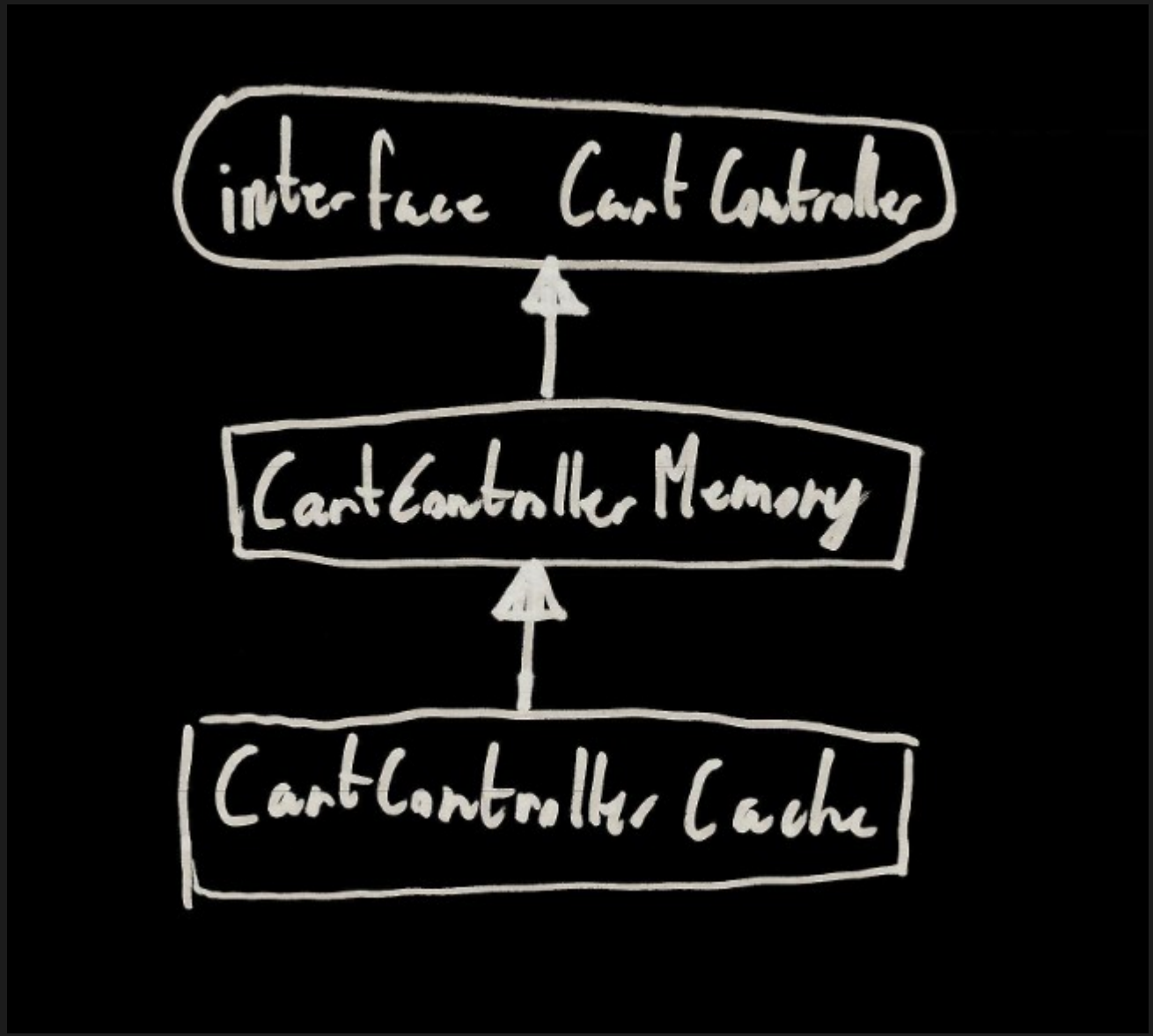

Provider
State
Consumer
Consumer
Increment()

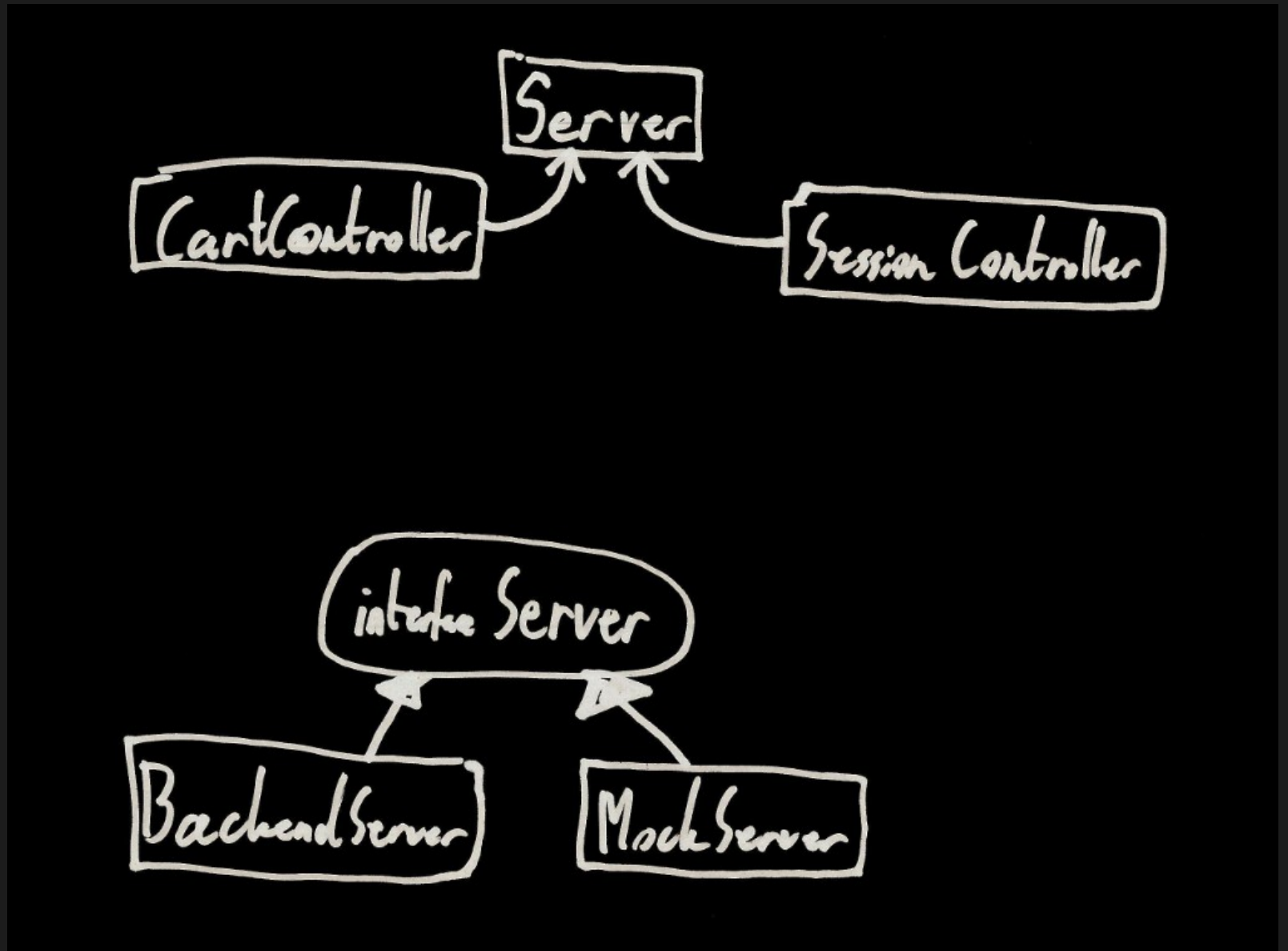

class A
class B
class C

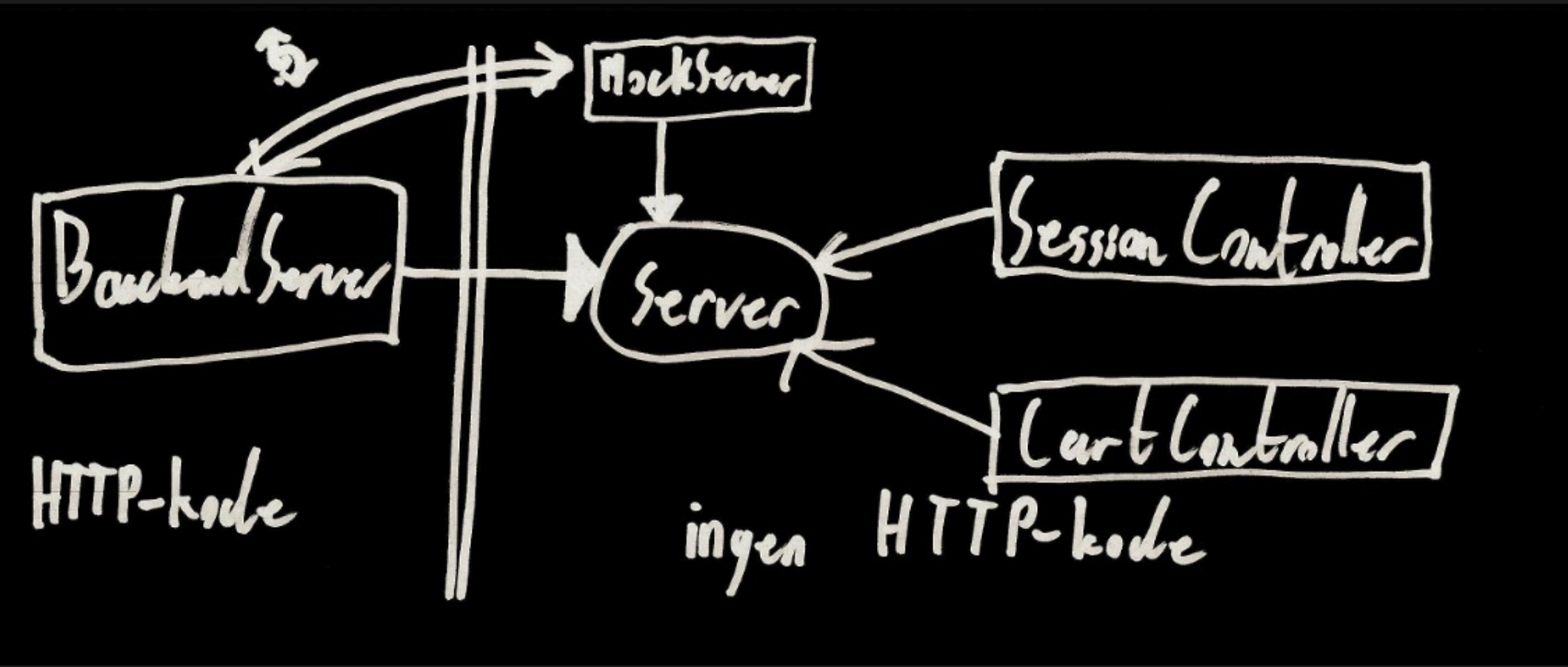
Design-patterns i **Fresh Plaza**

Eksempel med **SessionController** & **CartController**

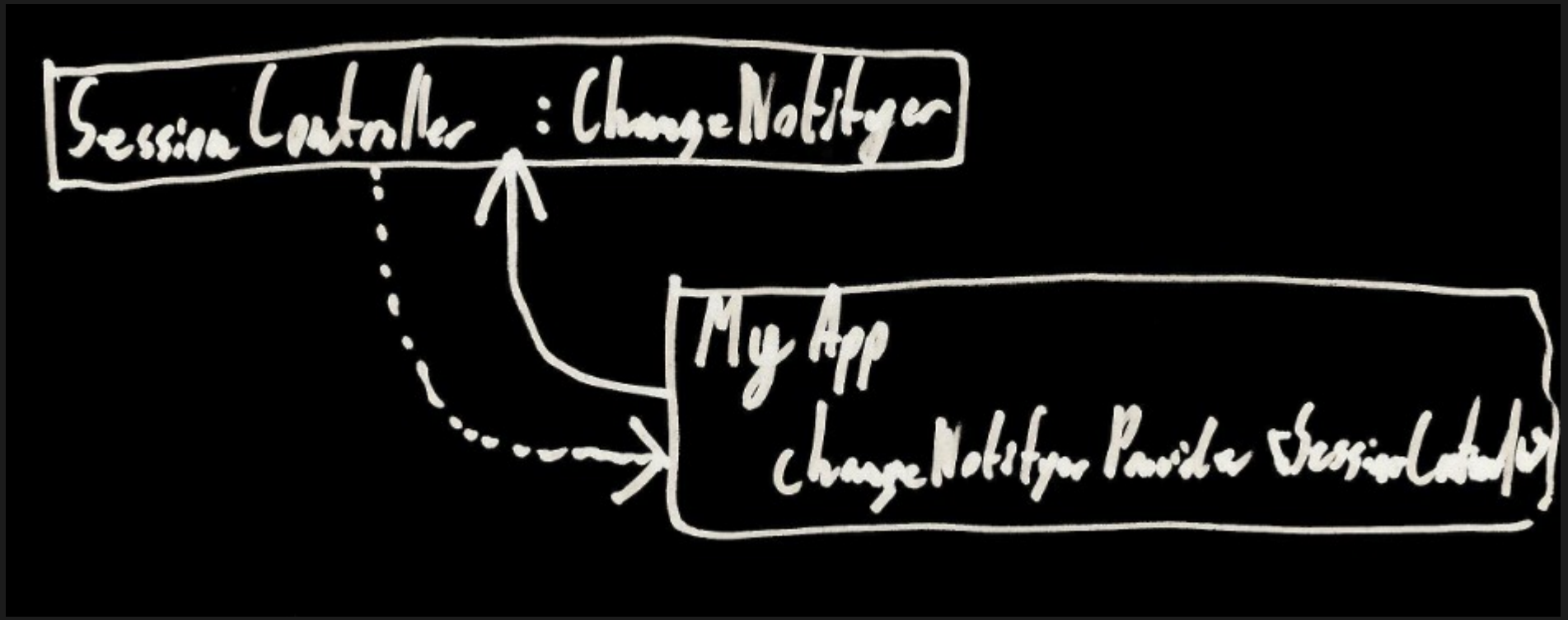

Cart Controller
Session Controller

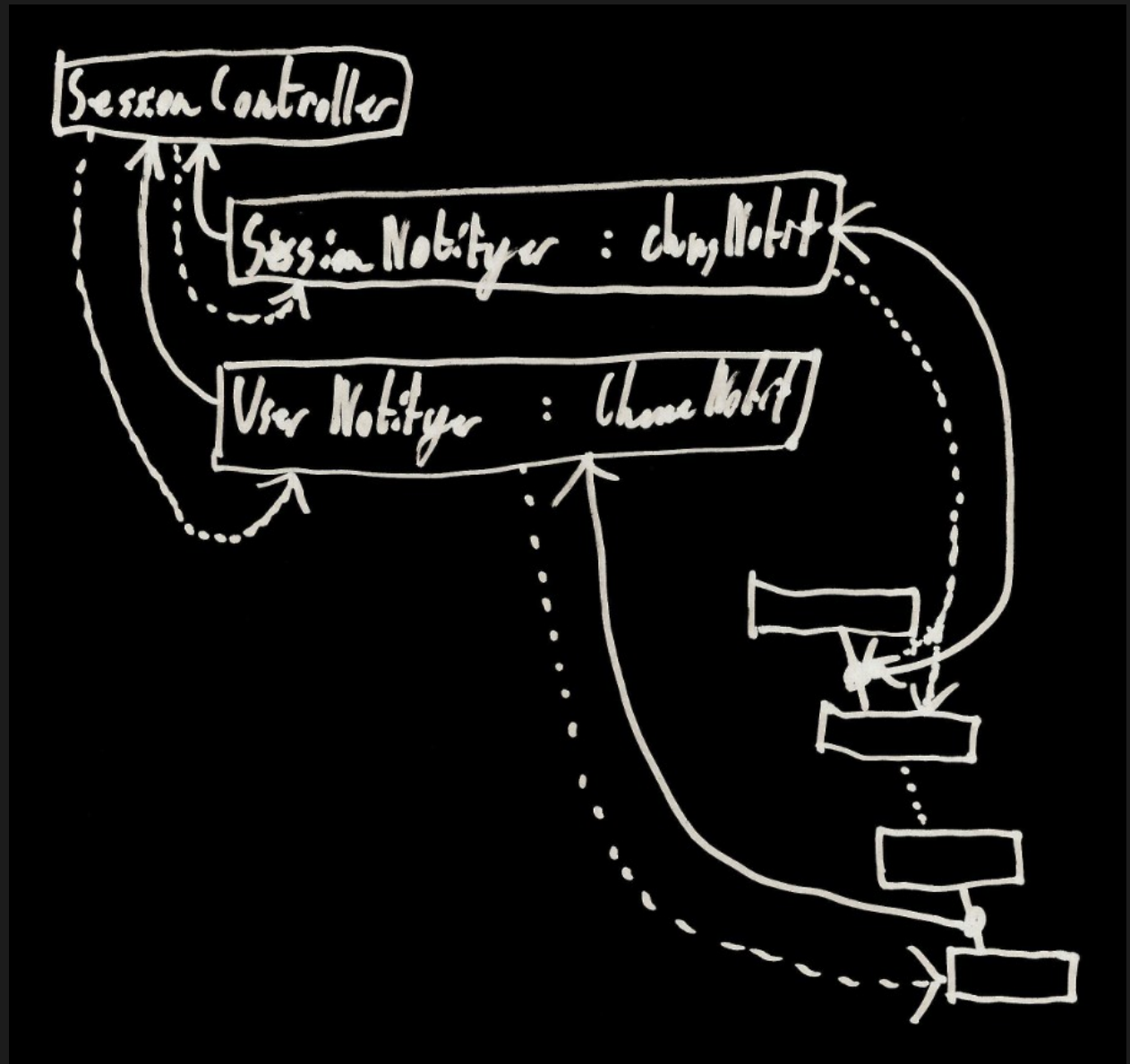

interface CartController
CartControllerMemory
CartControllerCache


Server
CartController
SessionController
interface Server
BackendServer
MockServer


MockServer
BackendServer
Server
SessionController
CartController
HTTP-kode
ingen
HTTP-kode

Eksempel med **SessionController** & **ChangeNotifierProvider**


SessionController : ChangeNotifier
MyApp
ChangeNotifierProvider(SessionController)


Session Controller
Session Notifyer : ChangeNotif
User Notifyer : ChangeNotif

```dart
class SessionController {
  final Server server;
  // ...
  final List<_ChangeListener> _sessionChangeListeners = [];
  final List<_ChangeListener> _userChangeListeners = [];
```

```dart
class CurrentUserProvider extends ChangeNotifier implements _ChangeListener {
  final SessionController controller;

  CurrentUserProvider({required this.controller}) {
    controller._addUserChangeListener(this);
  }
```

```dart
child: Consumer<CurrentUserProvider>(
  builder: (_, provider, ___) {
    final user = provider.controller.user;
    return Text("Saldo: ${formatDkkCents(user.balanceDkkCents)}",
        style: Theme.of(context).textTheme.headlineSmall); // Text
  },
), // Consumer
```

Responsibility management

```dart
class CartControllerMemory extends CartController {
  final Server server;
  final List<CartItem> cart = [];

  final SessionController sessionController;

  CartControllerMemory({required this.server, required this.sessionController});
```

```dart
class SessionController {
  final Server server;

  String? _sessionToken;
  User? _sessionUser;
```

```dart
Future<Result<Null, String>> purchase() async {
  // ...
}
```

```dart
class CartControllerMemory extends CartController {
  final Server server;
  // ...
  final SessionController sessionController;
```

```dart
Future<Result<Null, String>> purchase() async {
  final res = await sessionController.requestWithSession(
      (server, sessionToken) => server.purchaseCart(sessionToken, cart));
  switch (res) {
    case Ok<Null, String>():
      await sessionController.loadUpdatedUser();
      return const Ok(null);
    case Err<Null, String>(value: final message):
      return Err(message);
  }
}
```

```dart
class SessionController {
  final Server server;

  Future<Result<T, String>> requestWithSession<T>(
      RequestFunction func,
  ) async {
    switch (await cookieController.load()) {
      case Err<String, Null>():
        return const Err("unathorized");
      case Ok<String, Null>(value: final sessionToken):
        final result = await func(server, sessionToken);
        if (result case Err<T, String>(value: final message)) {
          if (message == "unauthorized") {
            cookieController.clear();
            _sessionUser = null;
            notifySessionChangeListeners();
            notifyUserChangeListeners();
            return const Err("unathorized");
          }
        }
        return result;
    }
  }
```

Backend

- Skrevet i C
- Dependencies:
  - OpenSSL (passwords)
  - Sqlite3
- Multithreaded HTTP server
- Bespoke JSON serializer/deserializer

C

- vs Node/Express, Deno/Oak
  - Kontrol
  - Performance
  - Abstraktionsniveau
  - Applikations- vs Systemudvikling
  - Manuel/automatik

```c
typedef struct {
    int64_t id;
    char* name;
    char* email;
    char* password_hash;
    int64_t balance_dkk_cent;
} User;
```

```
    char* user_to_json_string(const User* m)
    {
        static_assert(sizeof(User) == 40, "model has changed");

        String string;
        string_construct(&string);
        string_pushf(&string,
            "{"
            "\"id\":%ld,"
            "\"name\":\"%s\","
            "\"email\":\"%s\","
            "\"password_hash\":\"%s\","
            "\"balance_dkk_cent\":%ld"
            "}",
            m->id,
            m->name,
            m->email,
            m->password_hash,
            m->balance_dkk_cent);

        char* result = string_copy(&string);
        string_destroy(&string);
        return result;
    }
```

```
558     int users_register_req_from_json(UsersRegisterReq* m, const JsonValue* json)
559     {
560         static_assert(sizeof(UsersRegisterReq) == 24, "model has changed");
561
562         ObjField fields[] = {
563             { "name", JsonType_String },
564             { "email", JsonType_String },
565             { "password", JsonType_String },
566         };
567         if (!OBJ_CONFORMS(json, fields))
568             return -1;
569         *m = (UsersRegisterReq) {
570             .name = GET_STR("name"),
571             .email = GET_STR("email"),
572             .password = GET_STR("password"),
573         };
574         return 0;
575     }
```

```
DbRes db_user_insert(Db* db, const User* user)
{
    static_assert(sizeof(User) == 40, "model has changed");

    sqlite3* connection;
    CONNECT;
    DbRes res;

    sqlite3_stmt* stmt;
    int prepare_res = sqlite3_prepare_v2(connection,
        "INSERT INTO users (name, email, password_hash, balance_dkk_cent) "
        "VALUES (?, ?, ?, ?)",
        -1,
        &stmt,
        NULL);
    if (prepare_res != SQLITE_OK) {
        REPORT_SQLITE3_ERROR();
        res = DbRes_Error;
        goto l0_return;
    }

    sqlite3_bind_text(stmt, 1, user->name, -1, SQLITE_STATIC);
    sqlite3_bind_text(stmt, 2, user->email, -1, SQLITE_STATIC);
    sqlite3_bind_text(stmt, 3, user->password_hash, -1, SQLITE_STATIC);
    sqlite3_bind_int64(stmt, 4, user->balance_dkk_cent);

    int step_res = sqlite3_step(stmt);
    if (step_res != SQLITE_DONE) {
        REPORT_SQLITE3_ERROR();
        res = DbRes_Error;
        goto l0_return;
    }

    res = DbRes_Ok;
l0_return:
    if (stmt)
        sqlite3_finalize(stmt);
    DISCONNECT;
    return res;
}
```

Udgangspunkt i Express.js

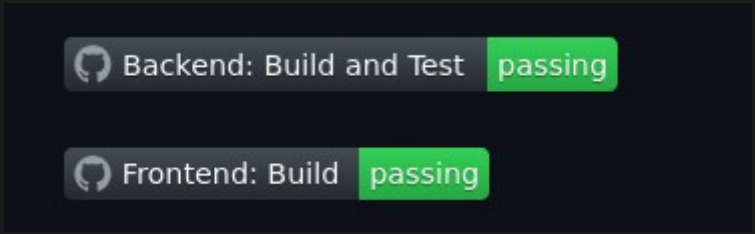
```
const app = express();

app.get("/products/all", (req, res) => {
    res.json({
        ok: true,
        products: [
            { id: 1, name: "Letmælk", price: 1200 },
            { id: 2, name: "Smør", price: 2400 },
        ],
    })
});

app.listen(8080);
```

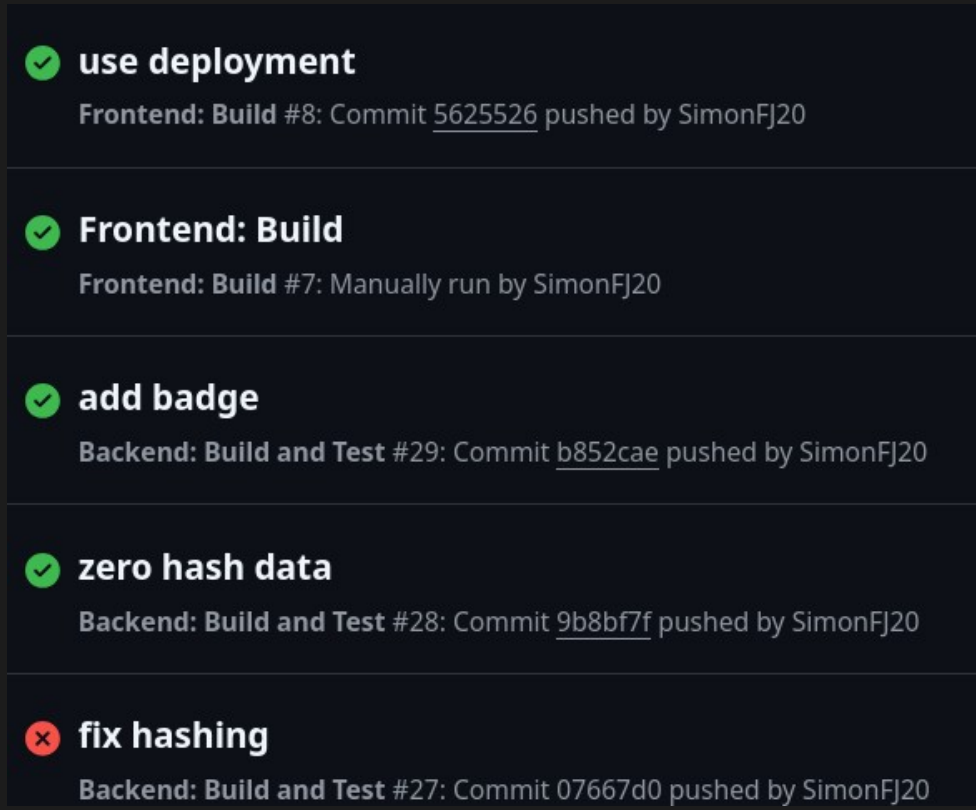
```c
int main(void)
{
    HttpServer* server = http_server_new((HttpServerOpts){
        .port = 8080,
        .workers = 8, /* 8 worker threads*/
    });

    http_server_get(server, "/products/all", get_products_all_handler);

    http_server_listen(server);

    http_server_free(server);
}
```

```
void get_products_all_handler(HttpCtx* ctx)
{
    RESPOND_JSON(ctx, 200,
        "{"
            "\"ok\":true,"
            "\"products\":["
                "{\"id\":1,\"name\":\"Letmælk\",\"price\":1200},"
                "{\"id\":2,\"name\":\"Smør\",\"price\":2400}"
            "]"
        "}");
}
```

CI og Deployment

- Automatisk build og test
- Automatisk deployment





Slige

- Slige
  - Compiler-system
- SBC (Slige backup compiler)
  - Løb tør for tid
  - Færre features
  - Ringere kode

Slige-sproget

- Inspireret af **Rust**
- **Basale** features
- God interop med **C**

```
15 fn main() -> int {
16     let i = 0;
17
18     while i < 10 {
19         println("Hello\ world");
20         i = i + 1;
21     }
22     return 0;
23 }
```

SBC

Slige Backup Compiler

1 uge

~3000 linjer kode

Human-readable

```
fn main() -> int {
    let a = 5;
    print("Hello\ world");
    return 0
}
```

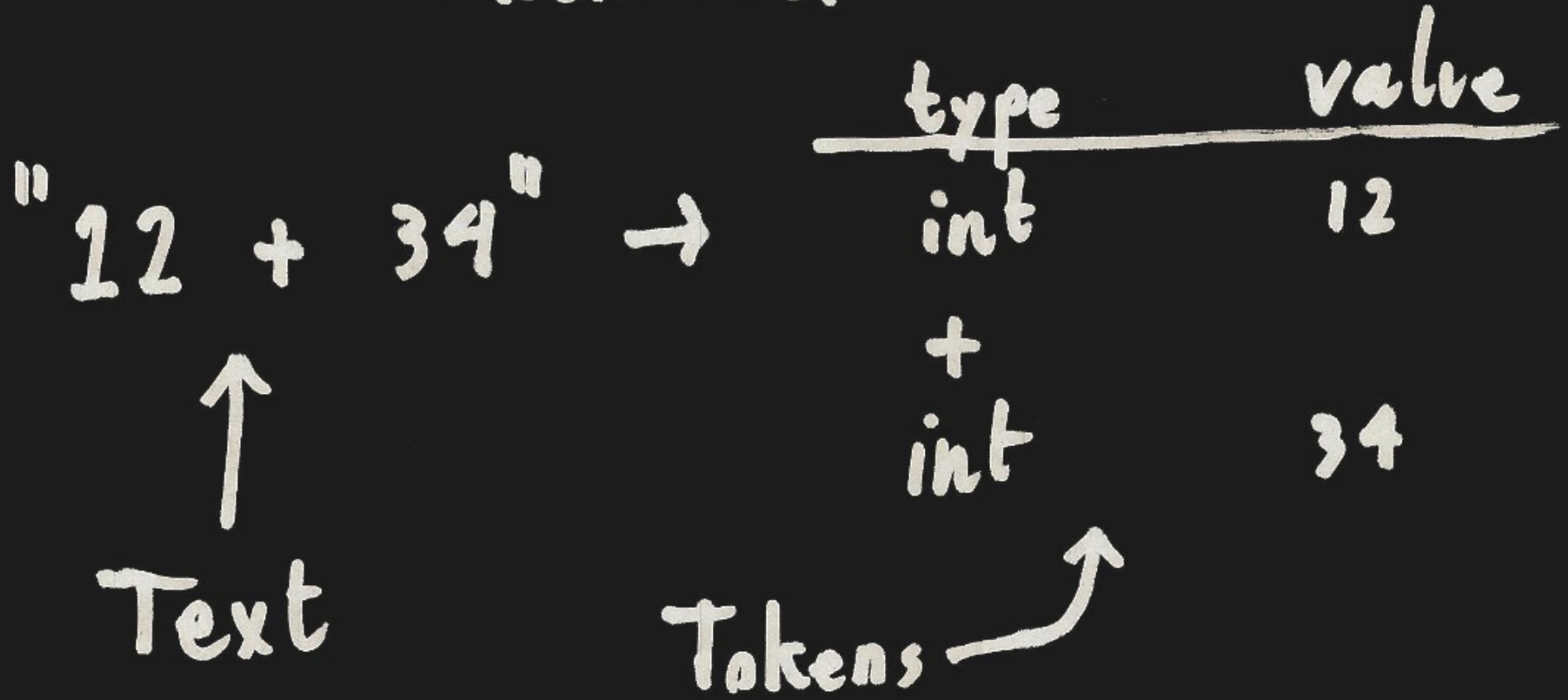
Machine code

```
main:
    sub rsp, 8
    mov [rsp], 5
    call print
    mov rax, 0
    ret
    .
    .
    .
```

```
2 fn main() -> int {
3
4     let i = 3 + 5;
5
6     println("Hello\ world");
7
8     return 0;
9 }
```

```
 8 sbc__main:
 9     push rbp
10     mov rbp, rsp
11     sub rsp, 16
12     jmp .L0
13 .L0:
14     mov rax, 3
15     mov rdi, 5
16     add rax, rdi
17     mov QWORD [rbp-16], rax
18     mov rax, sbc__string_0
19     push rax
20     call sbc__println
21     push rax
22     pop rax
23     pop rdi
24     mov QWORD [rbp-8], 0
25     jmp .exit
26 .exit:
27     mov rax, QWORD [rbp-8]
28     mov rsp, rbp
29     pop rbp
30     ret
```

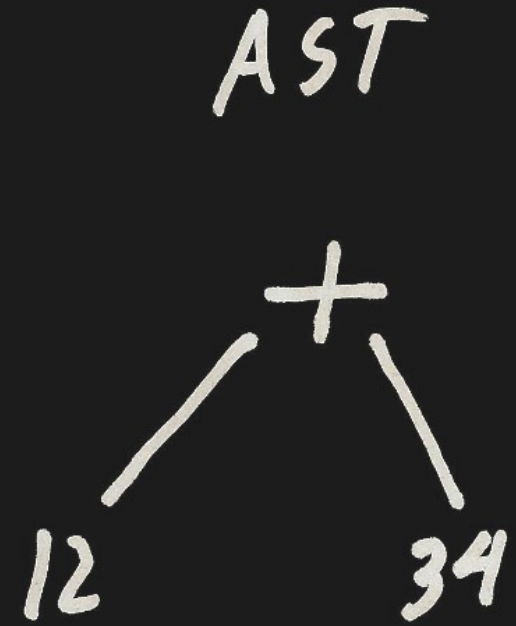
Tokenizer



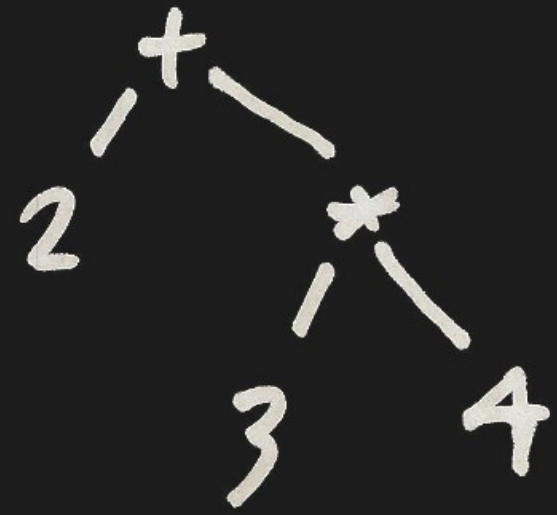
Parser

Tokens

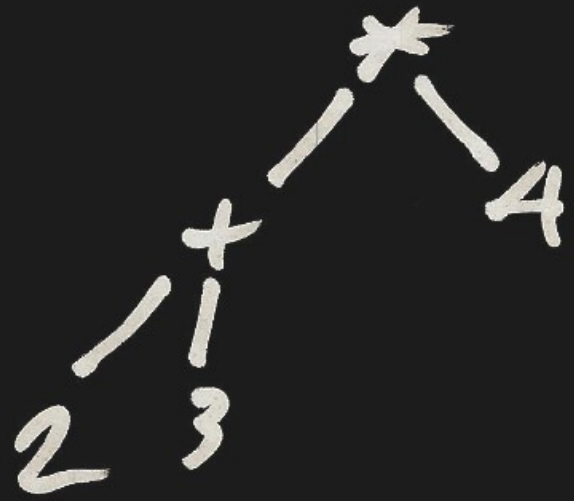
```
[
  int(12),
  +,
  int(34),
]
```



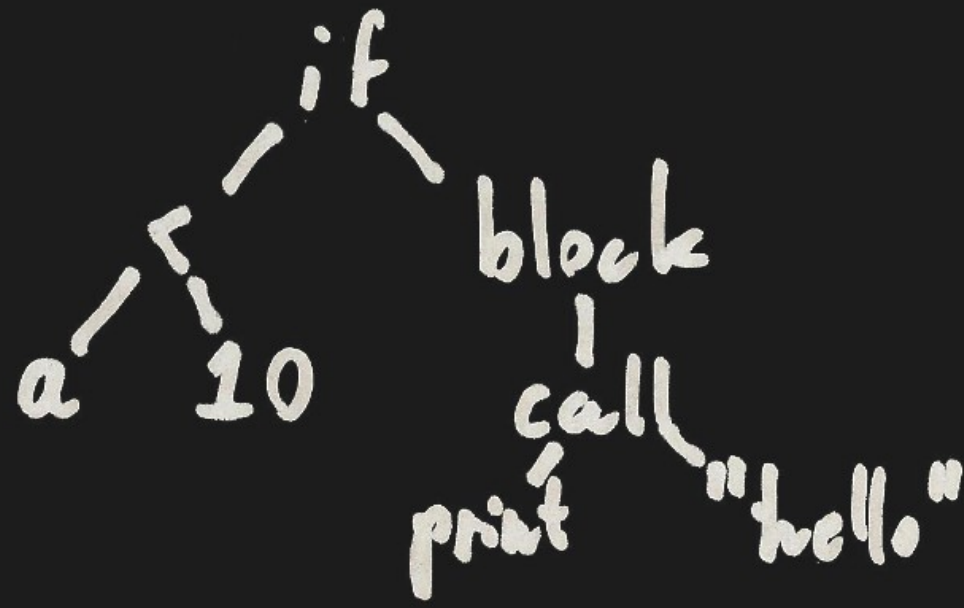
2 + 3 * 4



(2+3) * 4



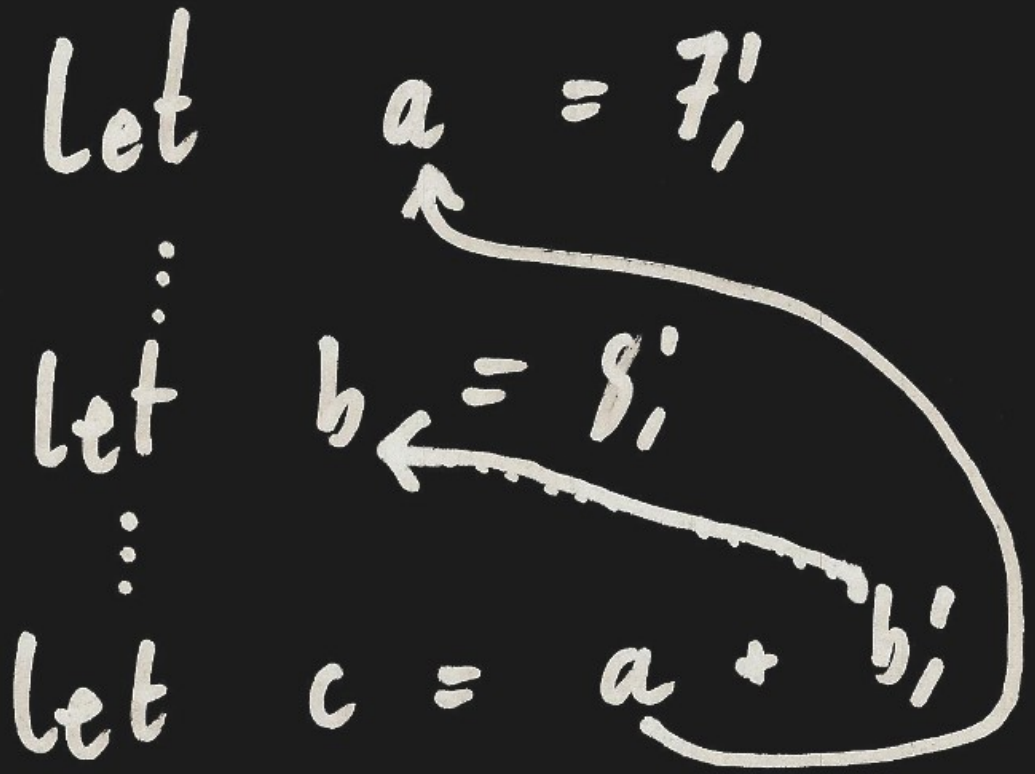
```
if a < 10 {
    print("hello.");
}
```



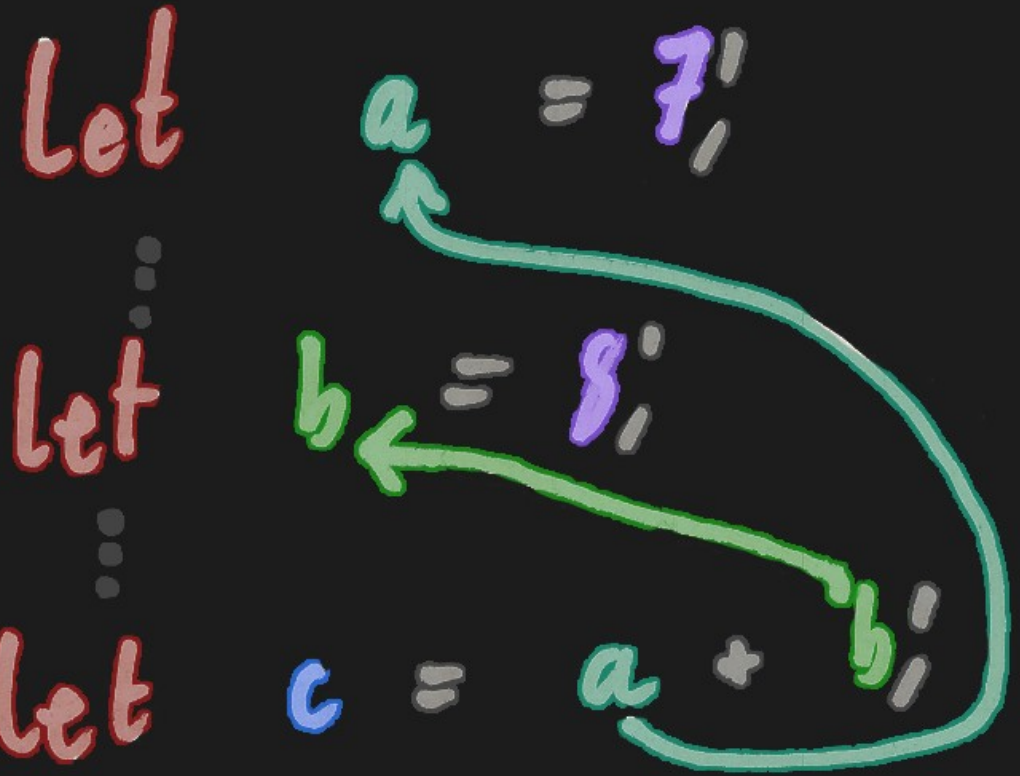
```
if a < 10 {
    println("Hello");
}
```

```
IfStmt {
    condition: LessThan {
        left: IdentExpr("a"),
        right: IntExpr(10),
    },
    body: Block([
        CallExpr {
            callee: IdentExpr("println"),
            arguments: [
                String("Hello"),
            ],
        }
    ]),
}
```

Symbol Resolution



Symbol Resolution



Type Checking

Checking

```
let v: int = 5;
```

type of 5 is 'int'

'int' assignable to 'int'?

```
let k = 7;
```

k's type is 7; type = int

Interence

Type Checking

Checking

```
let v: int = 5;
```

type of 5 is 'int'

'int' assignable to 'int'? ✓

```
let k = 7;
```

k's type is 7; type = int

Inference

Lowering (part 1)

AST

→

MIR

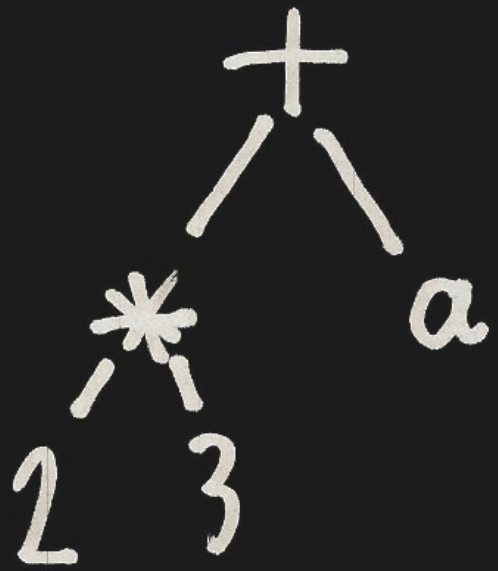
Mid-level

Intermediary

Representation

2*3+a

AST



MIR

```
push 2
push 3
mul
load %a
add
```

```
 3 fn main() -> int {
 4     let i = 0;
 5
 6     while i < 10 {
 7         println("Hello");
 8
 9         i = i + 1;
10     }
11     return 0;
12 }
```

```
main:
    %0 :: int
    %1 :: int
    .b0:
        push (int) 0
        store %1
        goto .b2
    .b2:
        load %1
        push (int) 10
        lt
        goto .b3, .b4
    .b3:
        push (*str) "Hello"
        push (fn(*str) -> int) println
        call 1
        pop
        load %1
        push (int) 1
        add
        store %1
        goto .b2
    .b4:
        push (int) 0
        store %0
        return
```

Lowering (part 2)
(elaboration)

MIR ⟶ LIR

Mid-level Intermediary Representation

Low-level Intermediary Representation

2 * 3 + a

MIR

```
push 2
push 3
mul
load %.a
add
```

Stack based

Register based

LIR

```
mov %.1, 2
push %.1
kill %.1
mov %.2, 3
push %.2
kill %.2
pop %.3
pop %.4
add %.3, %.4
⋮
```

```
main:
    %0 :: int
    %1 :: int
    .b0:
        push (int) 0
        store %1
        goto .b2
    .b2:
        load %1
        push (int) 10
        lt
        goto .b3, .b4
    .b3:
        push (*str) "Hello"
        push (fn(*str) -> int) println
        call 1
        pop
        load %1
        push (int) 1
        add
        store %1
        goto .b2
    .b4:
        push (int) 0
        store %0
        return
```

```
sbc__main:
    alloc_local %0, 8
    alloc_local %1, 8
.L0:
    store_val [%1], 0
    jmp .L1
.L1:
    load %4, [%1]
    mov_int %5, 10
    lt %4, %5
    kill %5
    jnz_reg %4, .L2
    kill %4
    jmp .L3
.L2:
    mov_string %9, string+0
    push %9
    kill %9
    call_fn sbc__println, 1
    pop %12
    pop %13
    kill %13
    kill %12
    load %15, [%1]
    mov_int %16, 1
    add %15, %16
    kill %16
    store_reg [%1], %15
    kill %15
    jmp .L1
.L3:
    store_val [%0], 0
    ret
```

Lowering (part 3)

(code generation)

LIR

Low-level

Intermediary

Representation

→

X86_64

assembly

ISA

Instruction set

Architecture

```
sbc__main:
    alloc_local %0, 8
    alloc_local %1, 8
.L0:
    store_val [%1], 0
    jmp .L1
.L1:
    load %4, [%1]
    mov_int %5, 10
    lt %4, %5
    kill %5
    jnz_reg %4, .L2
    kill %4
    jmp .L3
.L2:
    mov_string %9, string+0
    push %9
    kill %9
    call_fn sbc__println, 1
    pop %12
    pop %13
    kill %13
    kill %12
    load %15, [%1]
    mov_int %16, 1
    add %15, %16
    kill %16
    store_reg [%1], %15
    kill %15
    jmp .L1
.L3:
    store_val [%0], 0
    ret
```

```
 8 sbc__main:
 9     push rbp
10     mov rbp, rsp
11     sub rsp, 16
12     jmp .L0
13 .L0:
14     mov QWORD [rbp-16], 0
15     jmp .L1
16 .L1:
17     mov rax, QWORD [rbp-16]
18     mov rdi, 10
19     cmp rax, rdi
20     setl al
21     cmp rax, 0
22     jne .L2
23     jmp .L3
24 .L2:
25     mov rax, sbc__string_0
26     push rax
27     call sbc__println
28     push rax
29     pop rax
30     pop rdi
31     mov rax, QWORD [rbp-16]
32     mov rdi, 1
33     add rax, rdi
34     mov QWORD [rbp-16], rax
35     jmp .L1
36 .L3:
37     mov QWORD [rbp-8], 0
38     jmp .exit
39 .exit:
40     mov rax, QWORD [rbp-8]
41     mov rsp, rbp
42     pop rbp
43     ret
```

Register Allocation

LIR

```
mov %1, 3
mov %2, 4
add %1, %2
mov %3, 5
add %1, %3
```

```
mov rax, 3
mov rdi, 4
add rax, rdi
mov rdi, 5
add rax, rdi
```

rax: %1

rdi: %2 %3

```
mov_int %5, 3
mov_int %6, 4
add %5, %6
kill %6
mov_int %9, 5
add %5, %9
kill %9
store_reg [%2], %5
kill %5
```

```
11      mov rax, 3
12      mov rdi, 4
13      add rax, rdi
14      mov rdi, 5
15      add rax, rdi
16      mov QWORD [rbp-24], rax
```

Optimization (part 1)

MIR →  → MIR

- Dead code elimination
- Jump elision

```
main:
    %0 :: int
    %1 :: int
    .b0:
        push (int) 0
        store %1
        goto .b2
    .b1:
        return
    .b2:
        load %1
        push (int) 10
        lt
        goto .b3, .b4
    .b3:
        push (*str) "Hello"
        push (fn(*str) -> int) println
        call 1
        pop
        load %1
        push (int) 1
        add
        store %1
        goto .b2
    .b4:
        push (int) 0
        store %0
        goto .b1
    .b5:
        goto .b1
```

```
main:
    %0 :: int
    %1 :: int
    .b0:
        push (int) 0
        store %1
        goto .b2
    .b2:
        load %1
        push (int) 10
        lt
        goto .b3, .b4
    .b3:
        push (*str) "Hello"
        push (fn(*str) -> int) println
        call 1
        pop
        load %1
        push (int) 1
        add
        store %1
        goto .b2
    .b4:
        push (int) 0
        store %0
        return
```

Optimization (part 2)

LIR →  → LIR

- Stack spill reduction
- Instruction reduction

```
sbc__main:
    alloc_local %0, 8
    alloc_local %1, 8
.L0:
    mov_int %2, 0
    push %2
    kill %2
    pop %3
    store_reg [%1], %3
    kill %3
    jmp .L1
.L1:
    load %4, [%1]
    push %4
    kill %4
    mov_int %5, 10
    push %5
    kill %5
    pop %6
    pop %7
    lt %7, %6
    push %7
    kill %6
    kill %7
    pop %8
    jnz_reg %8, .L2
    kill %8
    jmp .L3
.L2:
    mov string %9, string+0
```

```
sbc__main:
    alloc_local %0, 8
    alloc_local %1, 8
.L0:
    store_val [%1], 0
    jmp .L1
.L1:
    load %4, [%1]
    mov_int %5, 10
    lt %4, %5
    kill %5
    jnz_reg %4, .L2
    kill %4
    jmp .L3
.L2:
    mov string %9, string+0
```

```
fn test(iterations: int) -> int {
    let acc = 0;
    let i = 0;
    while i < iterations {
        acc = acc + i + acc % 3;
        i = i + 1;
    }
    return acc;
}
```

```
int64_t test(int64_t iterations)
{
    int64_t acc = 0;
    int64_t i = 0;
    while (i < iterations) {
        acc += i + acc % 3;
        i += 1;
    }
    return acc;
}
```

```
sbc__test:
    push rbp
    mov rbp, rsp
    sub rsp, 24
    jmp .L0
.L0:
    mov QWORD [rbp-16], 0
    mov QWORD [rbp-24], 0
    jmp .L1
.L1:
    mov rax, QWORD [rbp-24]
    mov rdi, QWORD [rbp+16]
    cmp rax, rdi
    setl al
    cmp rax, 0
    jne .L2
    jmp .L3
.L2:
    mov rax, QWORD [rbp-16]
    mov rdi, QWORD [rbp-24]
    add rax, rdi
    push rax
    mov rax, QWORD [rbp-16]
    mov rdi, 3
    push rax
    push rdi
    call sbc__mod
    mov rax, rax
    pop rdi
    add rdi, rax
    mov QWORD [rbp-16], rdi
    mov rax, QWORD [rbp-24]
    mov rdi, 1
    add rax, rdi
    mov QWORD [rbp-24], rax
    jmp .L1
.L3:
    mov rax, QWORD [rbp-16]
    mov QWORD [rbp-8], rax
    jmp .exit
.exit:
    mov rax, QWORD [rbp-8]
    mov rsp, rbp
    pop rbp
    ret
```

```
test:
        push    rbp
        mov     rbp, rsp
        mov     qword ptr [rbp - 8], rdi
        mov     qword ptr [rbp - 16], 0
        mov     qword ptr [rbp - 24], 0
.LBB0_1:
        mov     rax, qword ptr [rbp - 24]
        cmp     rax, qword ptr [rbp - 8]
        jge     .LBB0_3
        mov     rax, qword ptr [rbp - 24]
        mov     qword ptr [rbp - 32], rax
        mov     rax, qword ptr [rbp - 16]
        mov     ecx, 3
        cqo
        idiv    rcx
        mov     rax, qword ptr [rbp - 32]
        add     rax, rdx
        add     rax, qword ptr [rbp - 16]
        mov     qword ptr [rbp - 16], rax
        mov     rax, qword ptr [rbp - 24]
        add     rax, 1
        mov     qword ptr [rbp - 24], rax
        jmp     .LBB0_1
.LBB0_3:
        mov     rax, qword ptr [rbp - 16]
        pop     rbp
        ret
```

```c
void route_post_carts_purchase(HttpCtx* ctx)
{
    // ...
    CartsPurchaseReq req;
    // ...
    ProductPriceVec prices;
    // ...
    int64_t total_dkk_cent
        = sbc_calculate_total_price((int64_t)item_amount, &req.items, &prices);

```

```rust
#[c_export("sbc_calculate_total_price")]
fn calculate_total_price(item_amount: int, items: *(), prices: *()) -> int {
```

```
#[c_export("sbc_calculate_total_price")]
fn calculate_total_price(item_amount: int, items: *(), prices: *()) -> int {
    let total_dkk_cent = 0;

    let i = 0;
    while i < item_amount {
        let price = prices_get_price(prices, i);
        let amount = items_get_amount(items, i);

        total_dkk_cent = total_dkk_cent + price * amount;

        i = i + 1;
    }

    return total_dkk_cent;
}
```

```
19 #[c_function("sbcs_prices_get_price")]
20 fn prices_get_price(prices: *(), i: int) -> int {}
21
22 #[c_function("sbcs_items_get_amount")]
23 fn items_get_amount(items: *(), i: int) -> int {}
```

```
113 int64_t sbcs_prices_get_price(const ProductPriceVec* prices, int64_t i)
114 {
115     return prices->data[i].price_dkk_cent;
116 }
117 int64_t sbcs_items_get_amount(const CartsItemVec* items, int64_t i)
118 {
119     return items->data[i].amount;
120 }
```

- S. F. Jakobsen