Loeng 8

Spring MVC,
Valideerimine (JSR 303)

Kordamine

- Mis kasu on Spring Core raamistikust?

OrderRowDao

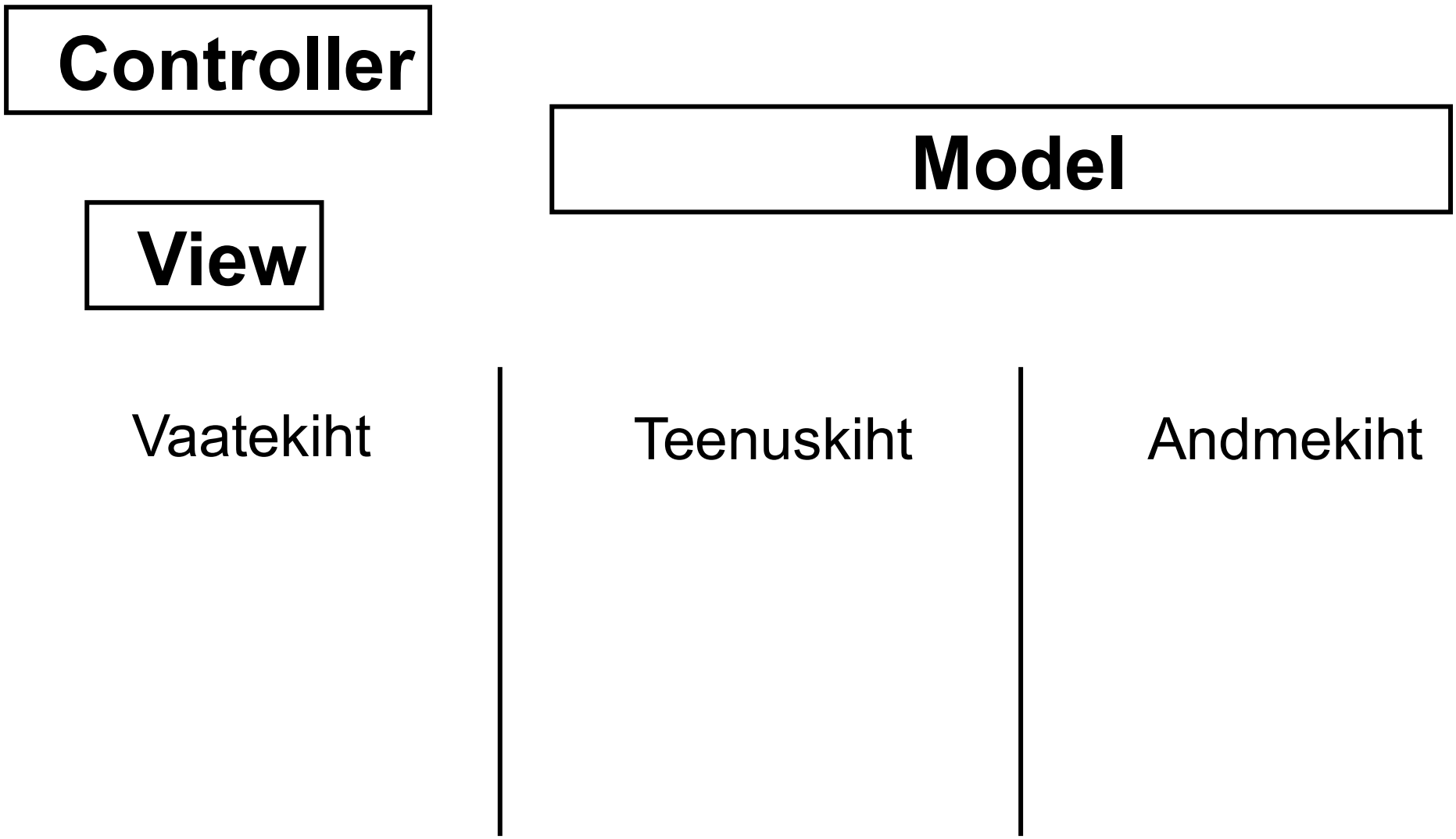
- Ridade salvestamiseks eraldi Dao või mitte?

Spring Mvc

Model View Controller (Mvc)

- **Model**: äriloogika
- **View**: esitlus
- **Controller**: päringute tõlgendamine, andmete ettevalmistus

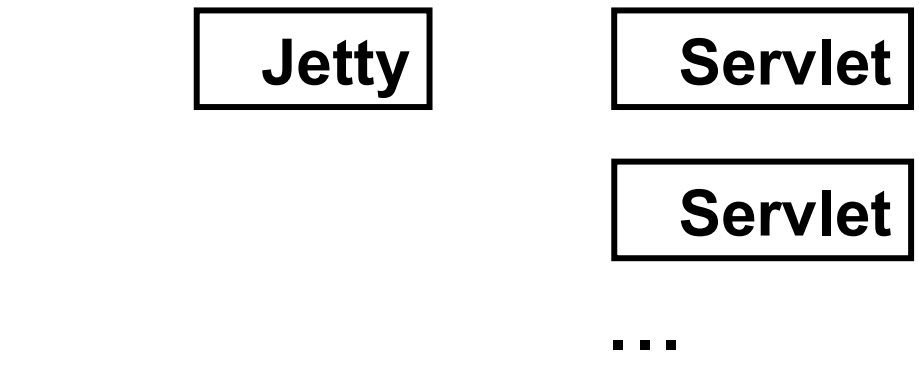
Kihiline arhitektuur



Spring Mvc kontroller

```java
@RestController
public class OrderController {

    private OrderDao dao;

    public OrderController(OrderDao dao) {
        this.dao = dao;
    }

    @PostMapping("api/orders")
    public String saveOrder(Order order) {
        return dao.save(order);
    }
```

Servlet vs. Spring Mvc



Servlet

Spring Mvc

Spring Mvc

- Integreerib Spring-i veebirakendusse
- Sunnib peale Mvc arhitektuuri
- Pakub mugavusmeetodeid ja palju muud kasulikku funktsionaalsust

Vajalikud teegid

```
implementation group: 'org.springframework',
        name: 'spring-webmvc',
        version: '5.2.9.RELEASE'

implementation group: 'org.springframework',
        name: 'spring-jdbc',
        version: '5.2.9.RELEASE'
```

Integreerib Spring-i veebirakendusse

```
ApplicationContext ctx =
        new AnnotationConfigApplicationContext(Config.class);

DataSource dataSource = ctx.getBean(DataSource.class);
```

```
@RestController
public class ReportController {

    private ReportService reportService;

    public ReportController(ReportService dao) {
        this.dao = dao;
    }
```

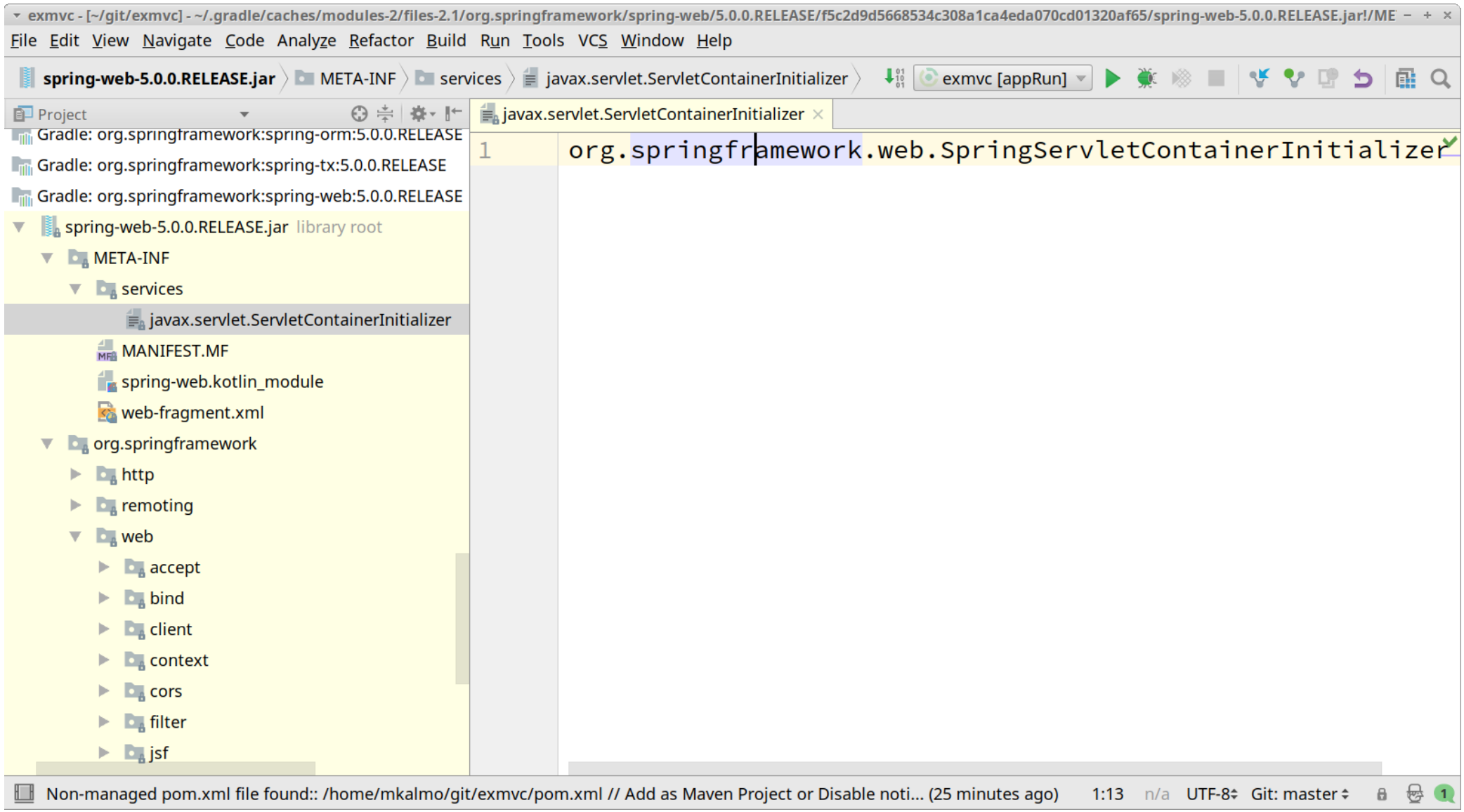
Raamistiku laadimine

```
@WebServlet("/hello")
public class HelloServlet extends HttpServlet { ...
```

Demo

```
@MyController
public class Controller {

    private Dao dao;

    public Controller(Dao dao) {
        this.dao = dao;
    }

    @MyPath("/api/orders")
    public String orderList() {
        return ...;
    }
```

Raamistiku laadimine



Raamistiku laadimine

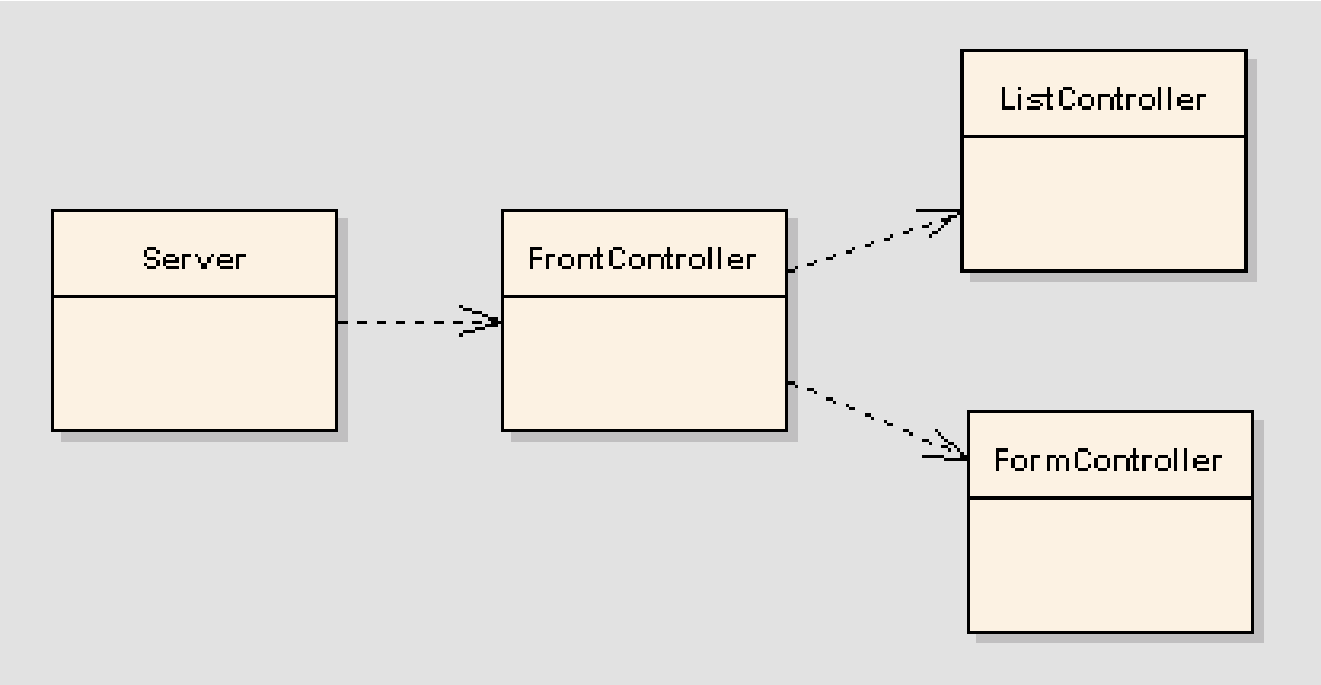
WebApplicationInitializer

org.springframework.web.context.AbstractContextLoaderInitializer
    org.springframework.web.servlet.support.AbstractDispatcherServletInitializer
        org.springframework.web.servlet.support.AbstractAnnotationConfigDispatcherServletInitializer

Raamistiku laadimine

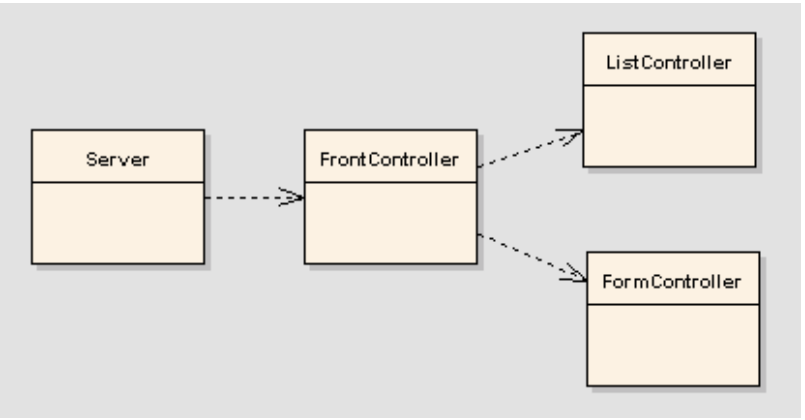
```
public class ApplicationInitializer extends
        AbstractAnnotationConfigDispatcherServletInitializer {

    @Override
    protected String[] getServletMappings() {
        return new String[] { "/api/*" };
    }

    @Override
    protected Class<?>[] getServletConfigClasses() {
        return new Class[] { MvcConfig.class };
    }

    ...
}
```

Raamistiku laadimine



Spring Mvc kontroller

```
@RestController
public class CustomerController {

    @GetMapping("customers")
    public List<Customer> list() {
        return dao.getAllCustomers();
    }
```




```
@Override
protected String[] getServletMappings() {
    return new String[] { "/api/*" };
}
```

http://localhost:8080/api/customers

Mapping

```
@RestController
public class CustomerController {

    @PostMapping("customers")
    public void save(...

    @GetMapping("customers")
    public List<Customer> list(...

    @DeleteMapping("customers")
    public void deleteAll(...

    ...
```

Kontrolleriteks jagamine

- CustomerController
- UserController
- ProductController
- Jne.

Sisendid

```
@PostMapping("customers")
public void save(@RequestBody Customer customer) { ...
```

Sisendid

```
@GetMapping("customers/{id}")
public Customer getById(@PathVariable Long id) {
    return dao.findById(id);
}
```

http://localhost:8080/api/customers/42

Sisendid

```
@GetMapping("customers/search")
public List<Customer> search(
            @RequestParam(defaultValue = "") String key) {

    return dao.search(key);
}
```

http://localhost:8080/api/customers/search?key=Jill

Sisendid

```
@GetMapping("customers/search")
public List<Customer> search(
                @RequestParam Optional<String> key) {

    return ...;
}
```

Konfiguratsioon

```
public class ApplicationInitializer extends
        AbstractAnnotationConfigDispatcherServletInitializer {

    ...

    @Override
    protected Class<?>[] getServletConfigClasses() {
        return new Class[] { MvcConfig.class };
    }

    ...
}
```

Konfiguratsioon

```
@EnableWebMvc
@Configuration
@ComponentScan(basePackages = {"hw8.customer"})
@PropertySource("classpath:/application.properties")
public class MvcConfig {

    @Bean
    public DataSource dataSource(Environment env) {
        ...
    }

    ...
```

Konfiguratsioon (@EnableWebMvc)

```
@RestController
public class CustomerController {

    @GetMapping("customers")
    public List<Customer> list() {
        return dao.getAllCustomers();
    }
```

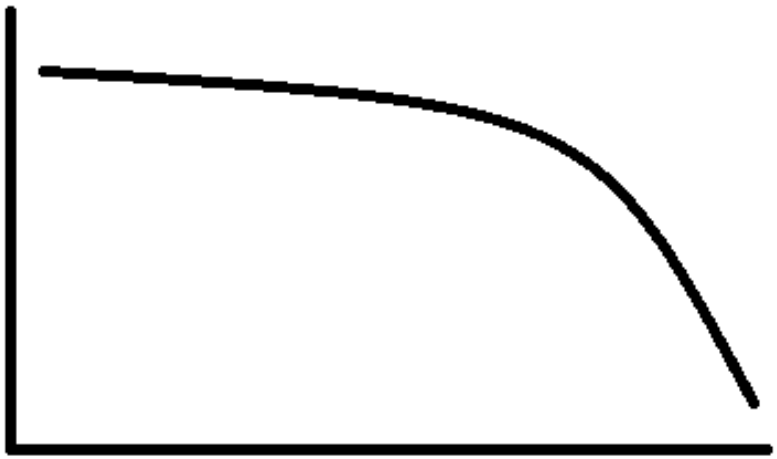
Suuremate projektide arendamisest

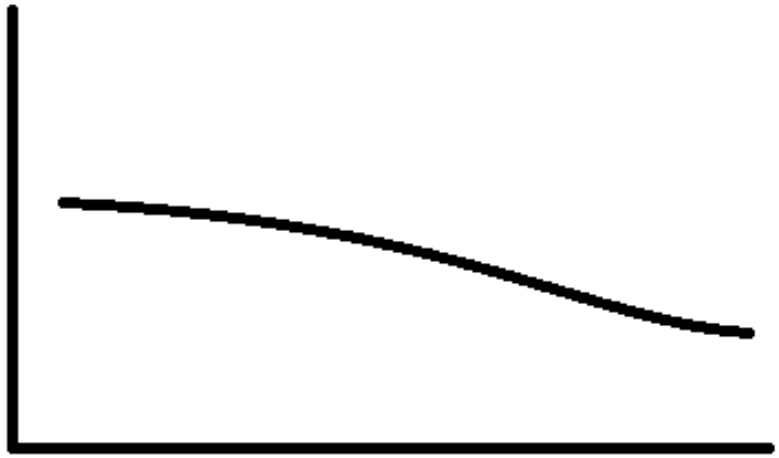
- Kooli projekt vs. „päris” projekt.

Olulised tegurid

- Erinõuded
- Erinevad tehnoloogiad
- Pärandkood
- Arendajate arv
- Arendajate vahetumine
- Arenduse prerioodi pikkus

Projekti kulgemisest



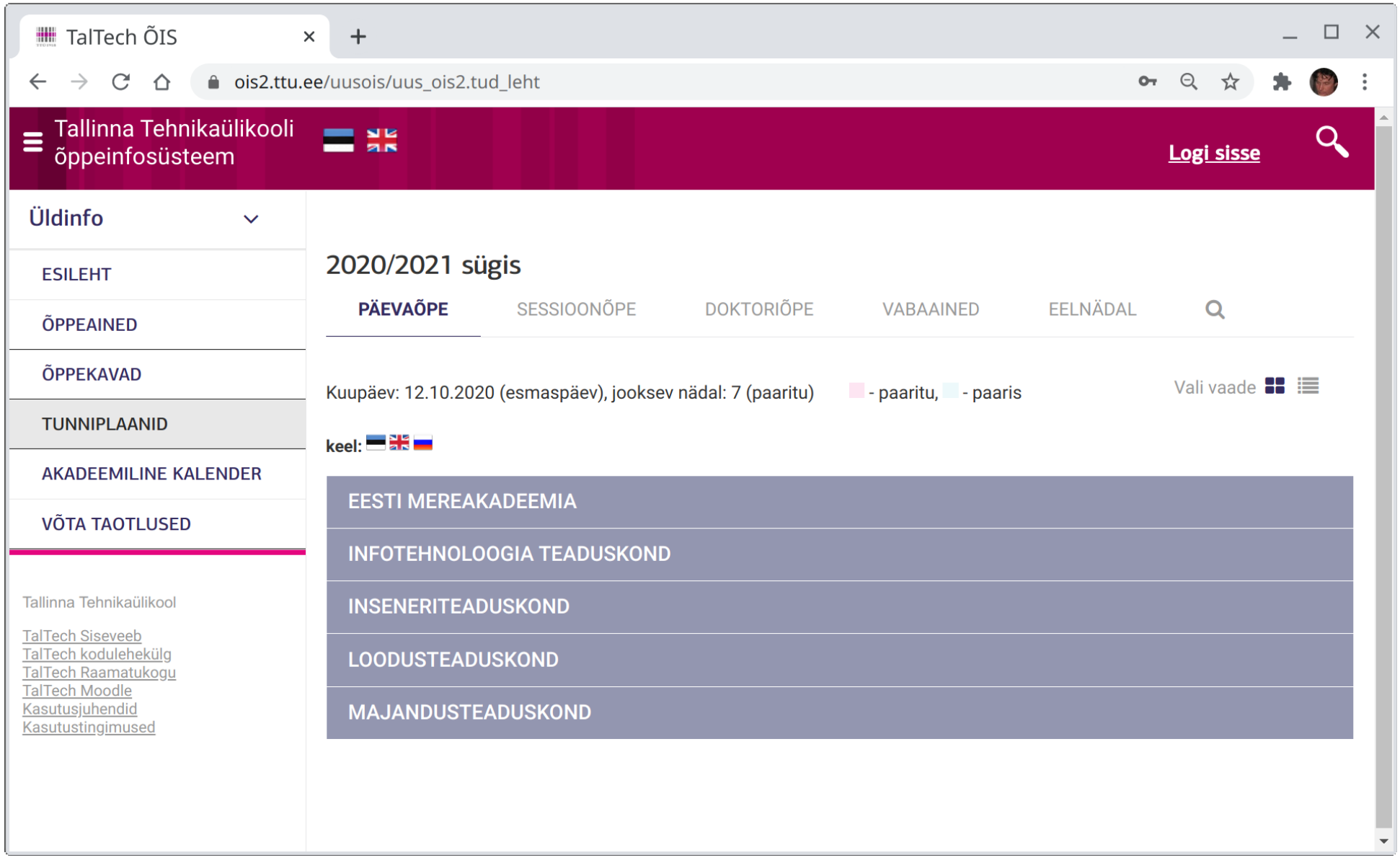


Vertikaal: funktsionaalsuse lisandumine
Horisontaal: aeg

Raamistike kasutamise mõju

- Raamistik annab arhitektuuri osaliselt ette
- Raamistik annab disaini osaliselt ette
- Google-st leiab näiteid

Projekti kulgemisest



Leevendavad tegurid

- Põhjalikumad teadmised
- Paindlik disain
- Automaattestimine

Paindlik disain

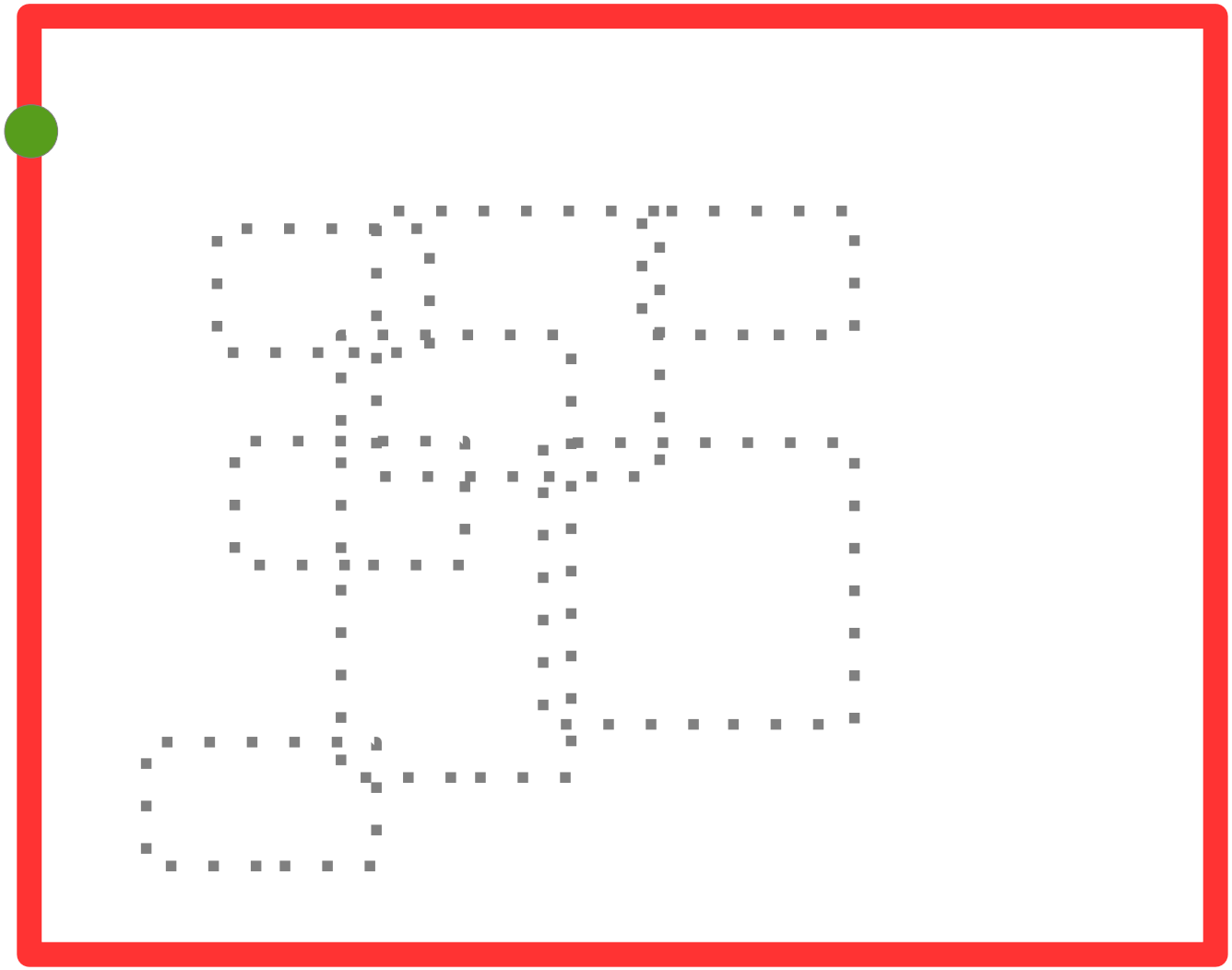
- Äriloogika on raamistikust lahus

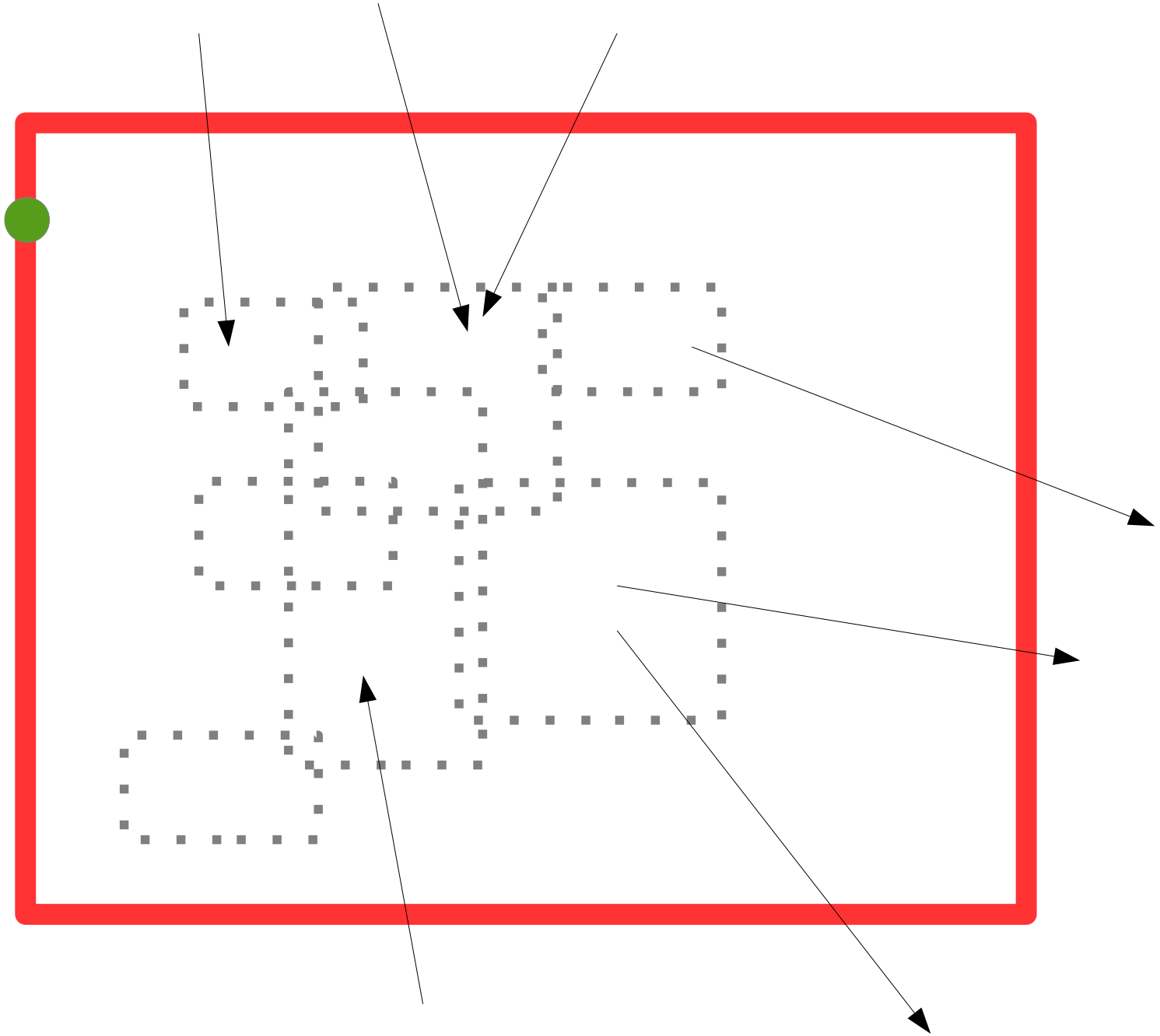
Arendus suure rakenduse kontekstis

- Teiste koodist arusaamine
- Koodi käivtamine sobivate parameetritega

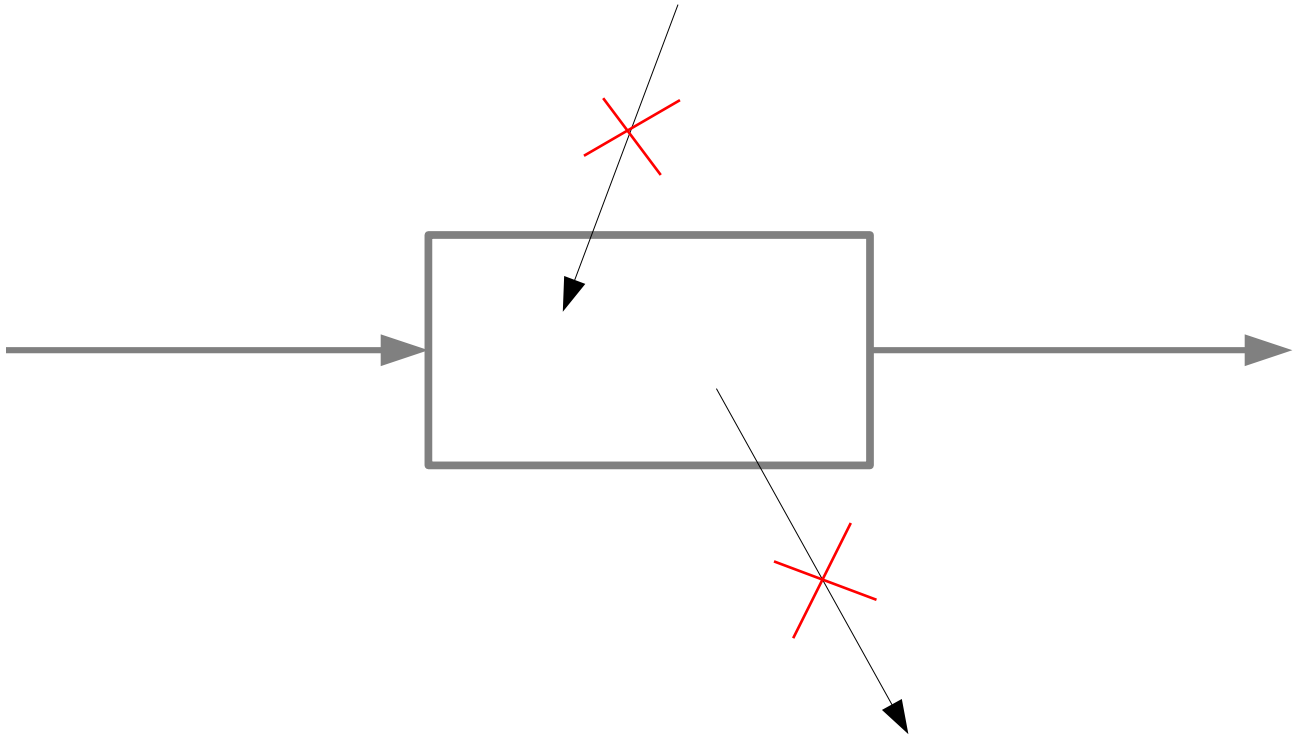
Koodi disainist

„Objektorienteeritud” disain

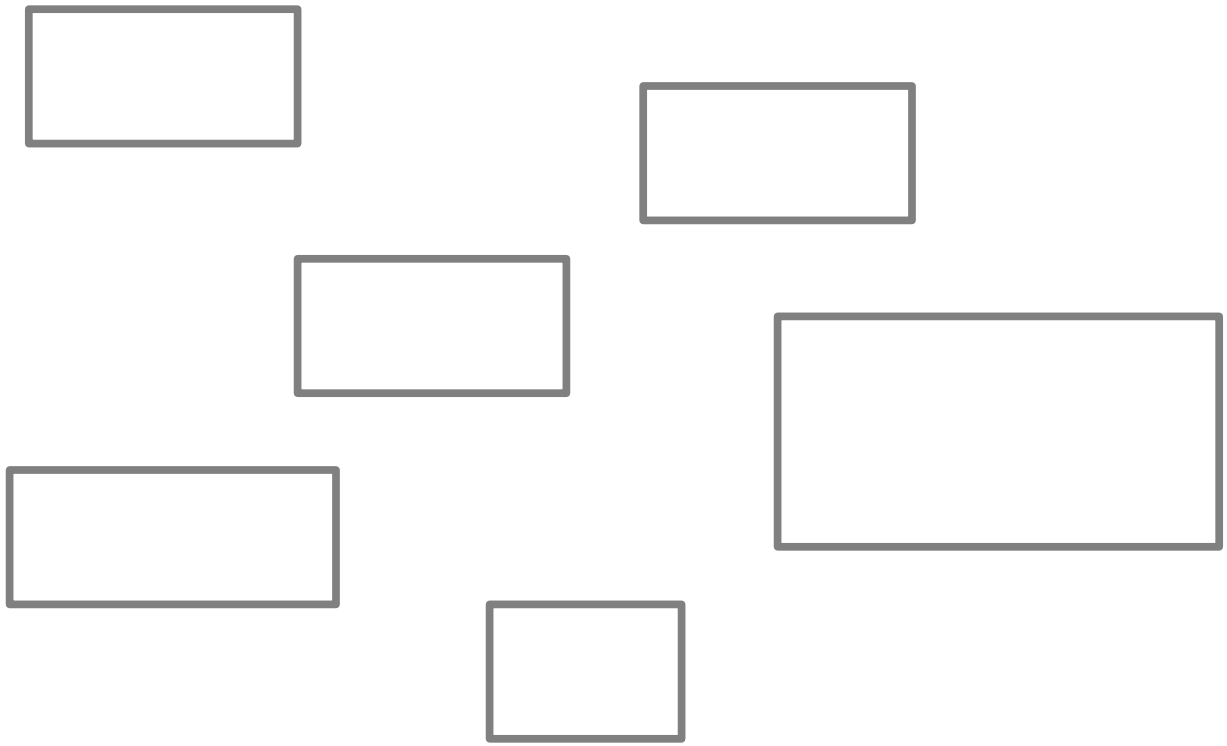




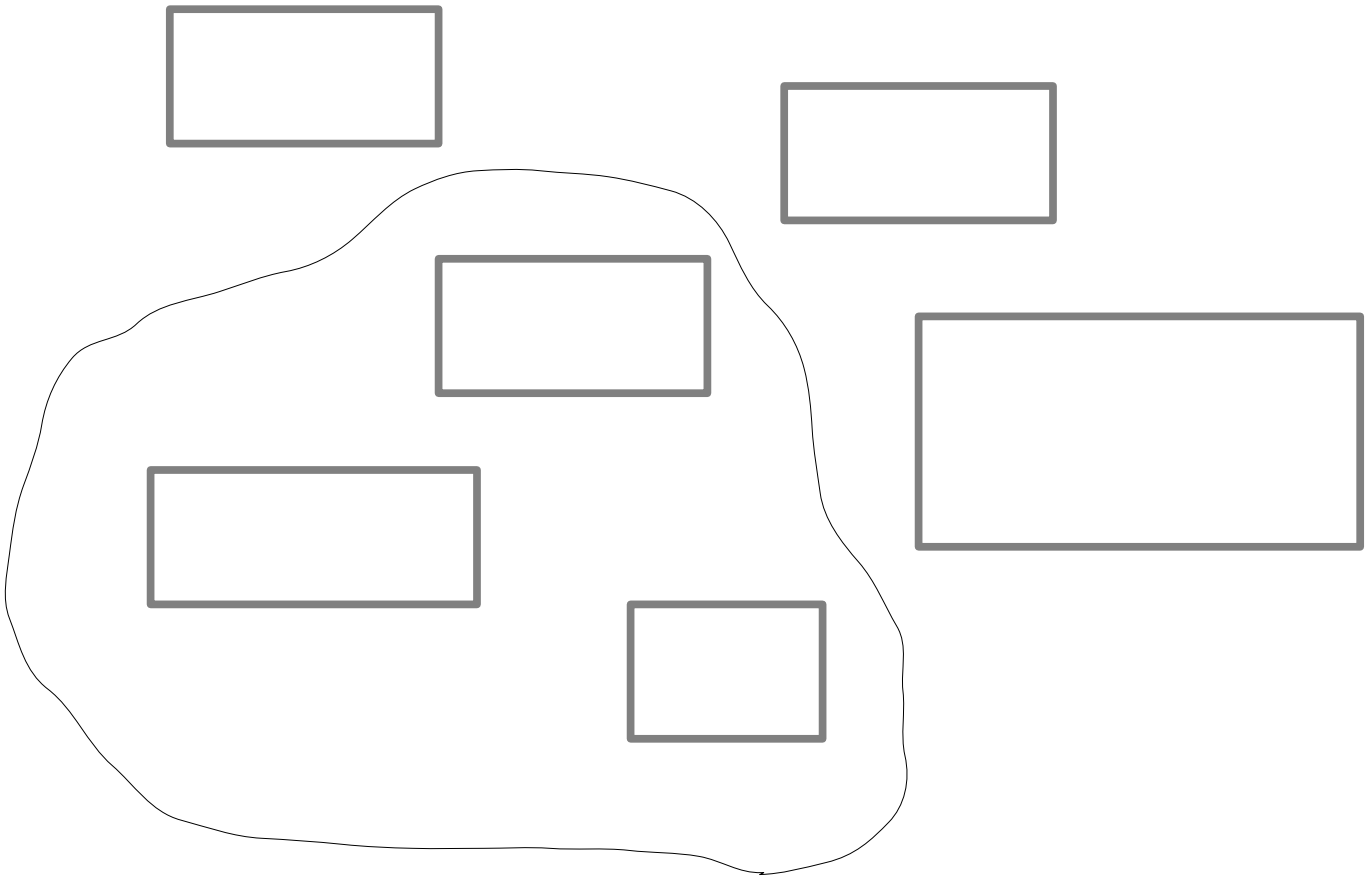
Funktsionaalne disain



Funktsionaalne disain

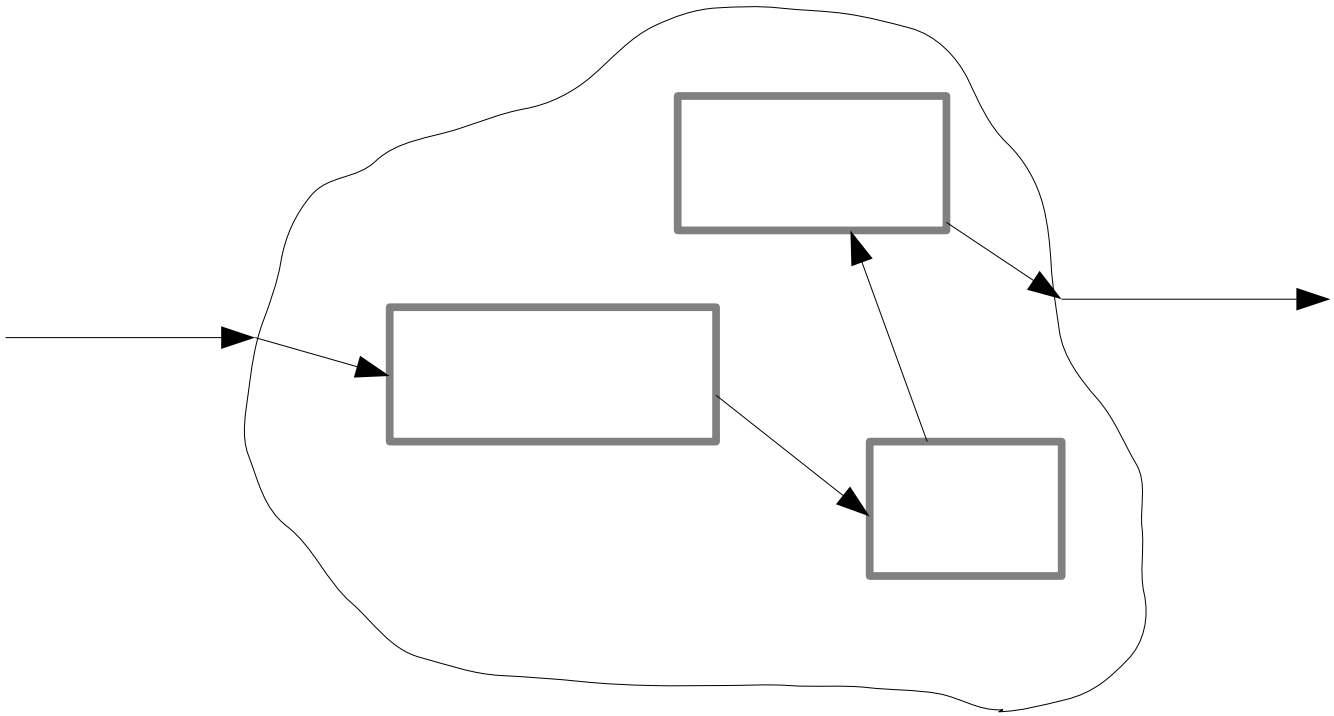


Funktsionaalne disain



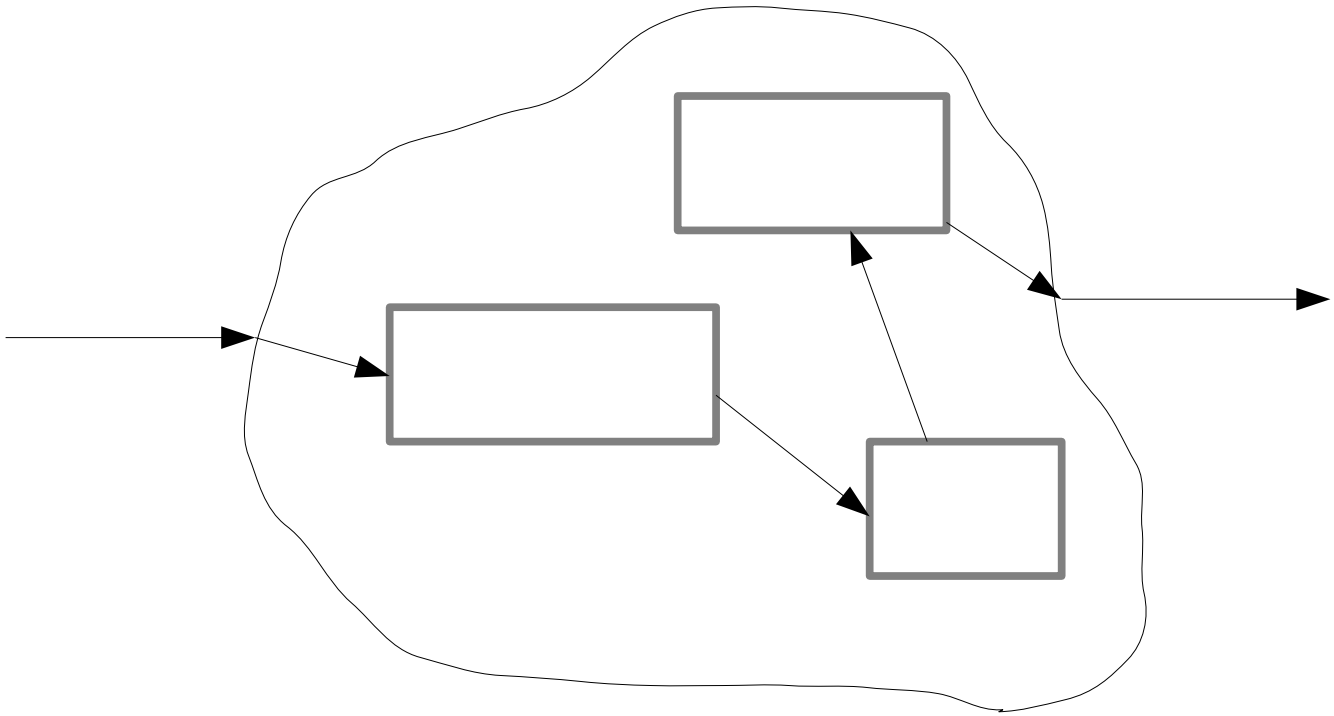
Siduv kood

- Ei oma keerulist loogikat
- Seob kokku selgelt eristatud alamosad



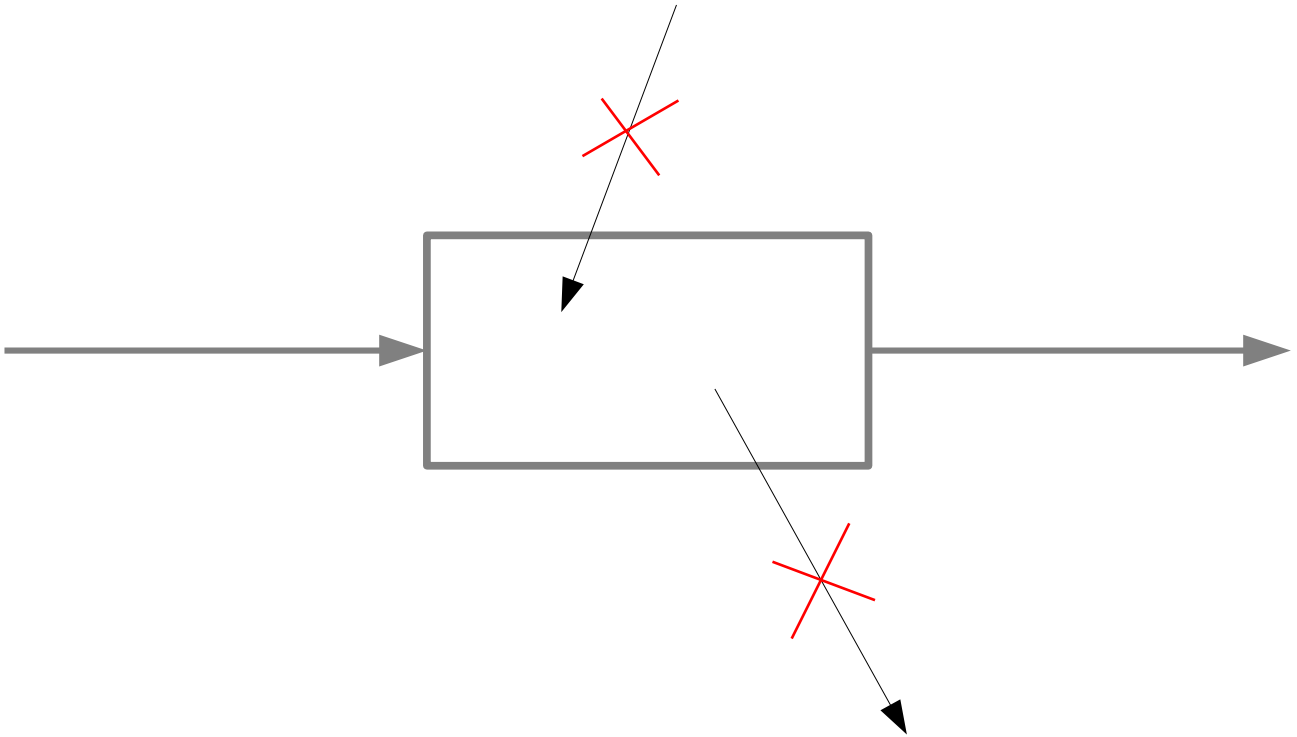
Koodil on vaid kaks võimalikku rolli

- Arvutada
- Siduda

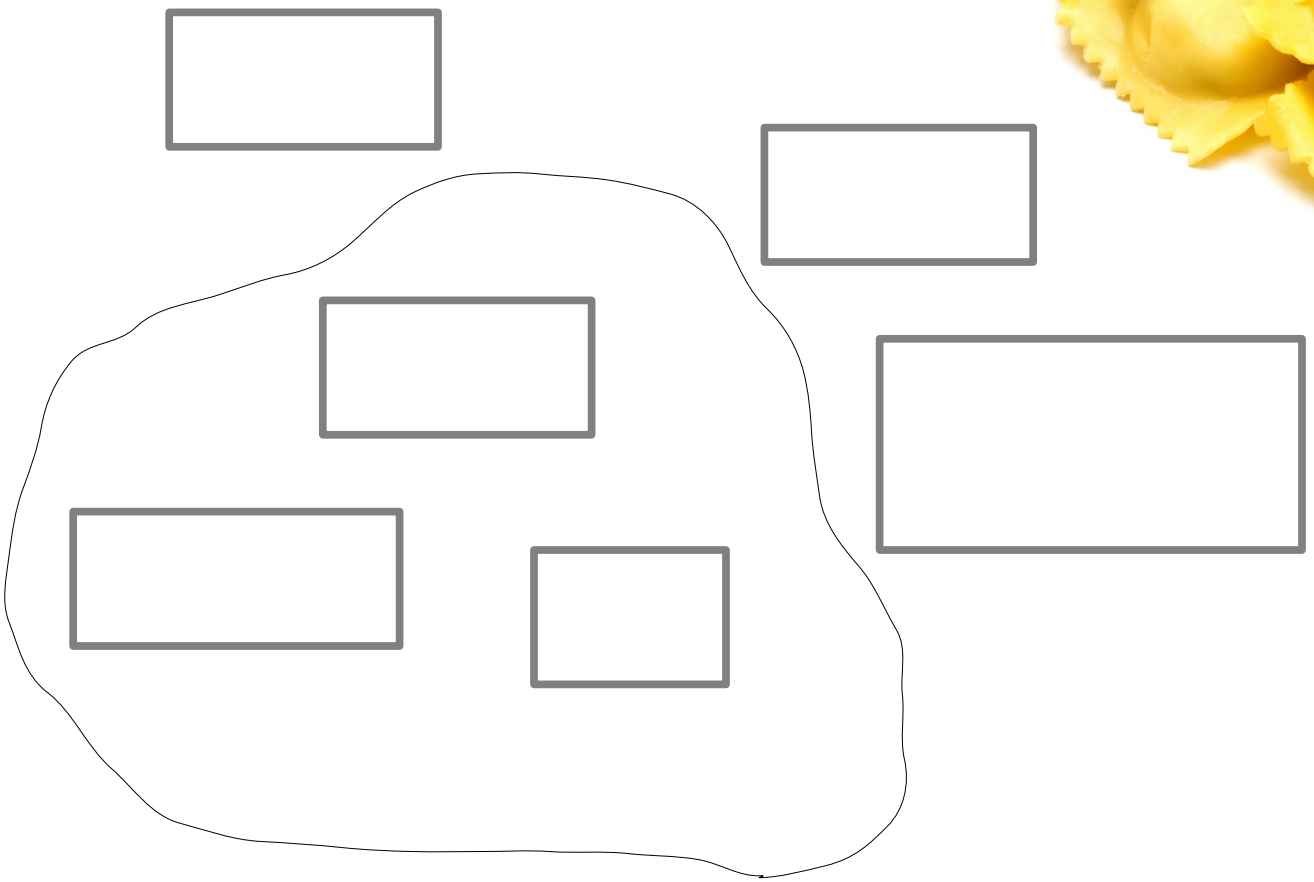


Arvutav kood

- Ei küsi andmeid
- Ei salvesta andmeid



Ravioli kood



Funktsionaalne vs oo

Automaattestimine

- Moodultestimine (Unit Testing)
- Terviku testimine (End to end)
- MockMvc

JSR 303
Bean validation

JSR

– Java Specification Request

JSR 303 teek

```
implementation group: 'org.hibernate',
               name: 'hibernate-validator',
               version: '6.1.6.Final'
```

Käsitsi valideerimine

```
private List<String> getValidationErrors(
        HttpServletRequest request) {

    List<String> errors = new ArrayList<String>();

    if ("".equals(request.getParameter("firstName"))) {
        errors.add("Sisesta eesnimi!");
    }

    if ("".equals(request.getParameter("name"))) {
        errors.add("Sisesta perekonnanimi!");
    }

    return errors;
}
```

Kasutamise näide (bean)

```
public class Person {

    @NotNull
    @Size(min = 2, max = 14)
    private String name;

    @NotNull
    @Min(0)
    private Integer age;

    @NotNull
    @Pattern(regexp = "[_0-9]+")
    private String code;
```

Demo (hibernate validator)

Alamobjektid

```
public class Order {

    @NotNull
    @Valid
    private List<OrderRow> orderRows;
```

JSR 303

- Defineerib annotatsioonid: @Digits, @Future, @Past, @Max, @Min, @Null, @NotNull, @Pattern, @Size …
- Võimalus ise juurde luua

Validaatori kasutamine Spring Mvc-s

```
@PostMapping("customers")
public void save(@RequestBody @Valid Customer customer) {
    ...
}
```

Vigade püüdmine (kontroller)

```
@ExceptionHandler
@ResponseStatus(HttpStatus.BAD_REQUEST)
public ValidationErrors handleValidationErrors(
            MethodArgumentNotValidException exception) {

    ValidationErrors errors = new ValidationErrors();

    ... = exception.getBindingResult().getFieldErrors();

    ...

    return errors;
}
```

Valideerimise vastus JSON

```
{
  "errors": [
    { "code": "NotNull.customer.code",
      "arguments": []
    },
    { "code": "Size.customer.firstName",
      "arguments": ["15", "2"]
    }
  ]
}
```

Valideerimise vastus JSON

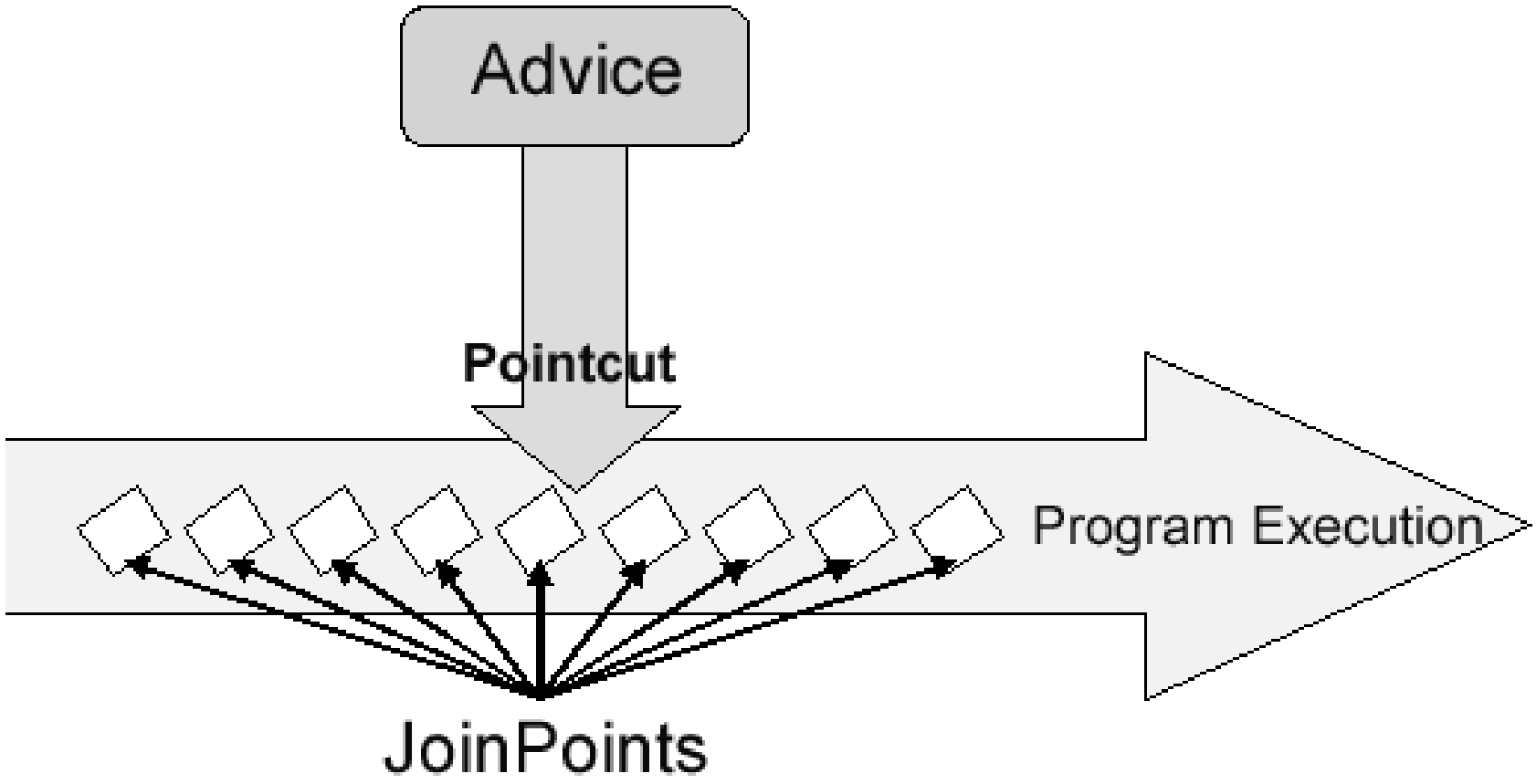
```
public class ValidationErrors {

    private List<ValidationError> errors = new ArrayList<>();

}
```

```
public class ValidationError {
    private String code;
    private List<String> arguments;
}
```

Kordamine

- Mis on AOP?

AOP



Kasutamise näide (advice)

```
@RestControllerAdvice
public class ValidationAdvice {

    @ExceptionHandler
    @ResponseStatus(HttpStatus.BAD_REQUEST)
    public ValidationErrors handleMethodArgumentNotValid( ...

        ...

}
```

Projekti 8. osa

- Kasutatavad tehnoloogiad: Spring Mvc, JSR 303, Hsql, Gradle

Projekti 8. osa

Päringu näide:

GET /api/orders/1/installments?start=2020-11-04&end=2021-01-01

Vastuse näide:

HTTP/1.1 200 OK
Content-Type: application/json


```
[
  {"amount":3,"date":"2020-11-04"},
  {"amount":3,"date":"2020-12-01"},
  {"amount":3,"date":"2021-01-01"}
]
```

Projekti 8. osa

- Osamakseteks jagamise reeglid
  - Osamaksed tulevad kuu 1. päevale.
  - Kui periood algab pärast 1. kuupäeva, siis on makse perioodi esimesel päeval.
  - Minimaalne ühik on 1 Euro. Kui summa ei jagu Euro täpsusega, siis jaotatakse vahe viimaste osamaksete vahel.
  - Miinimum osamakse summa on 3 Eurot.