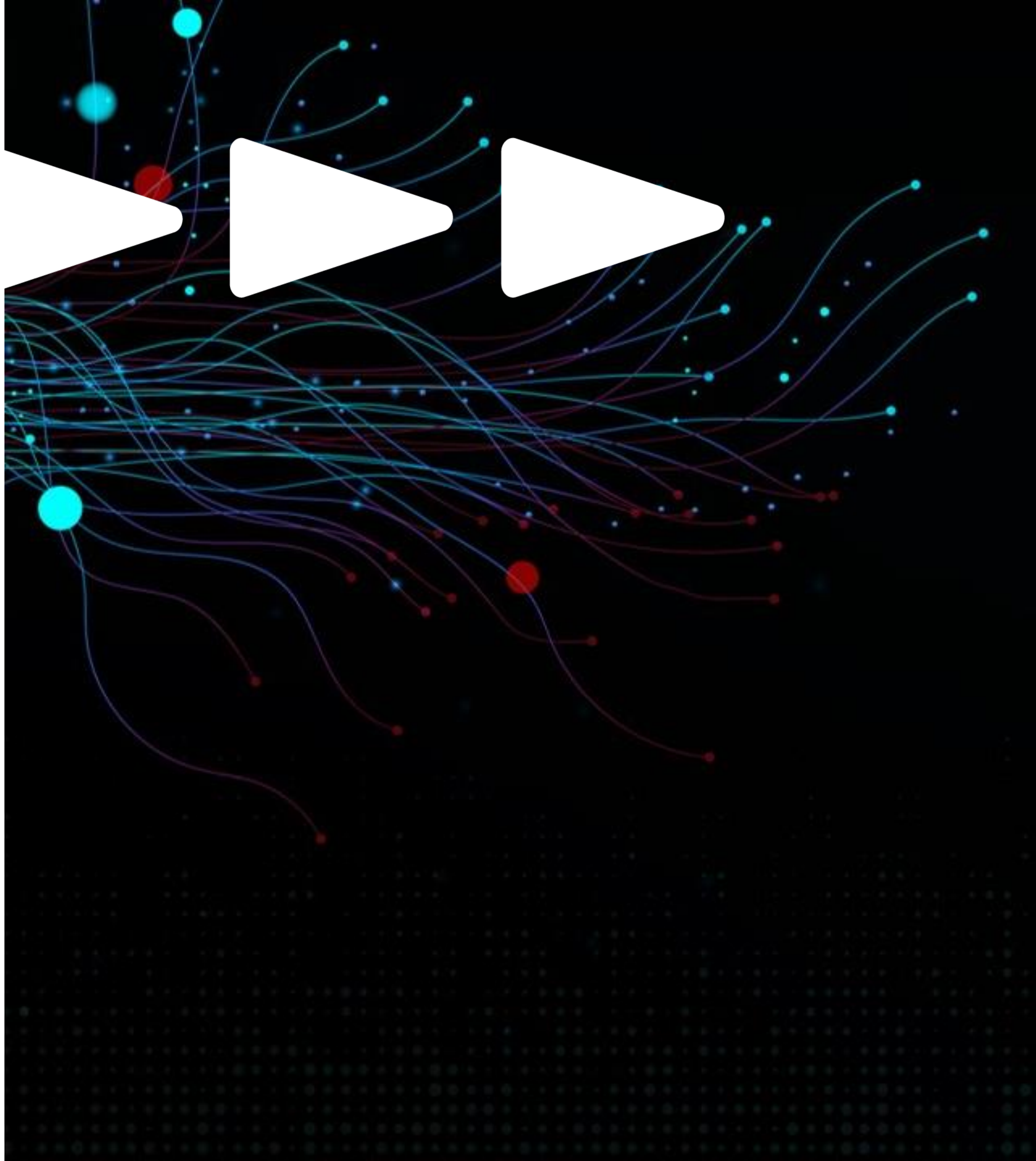
autor prezentacije

TOMISLAV PETRIČEVIĆ

DOBRE LOŠE RUZNE STRANE ANOTACIJA

KING ICT
INFORMATION & COMMUNICATION TECHNOLOGIES

JavaCro'22 Spring



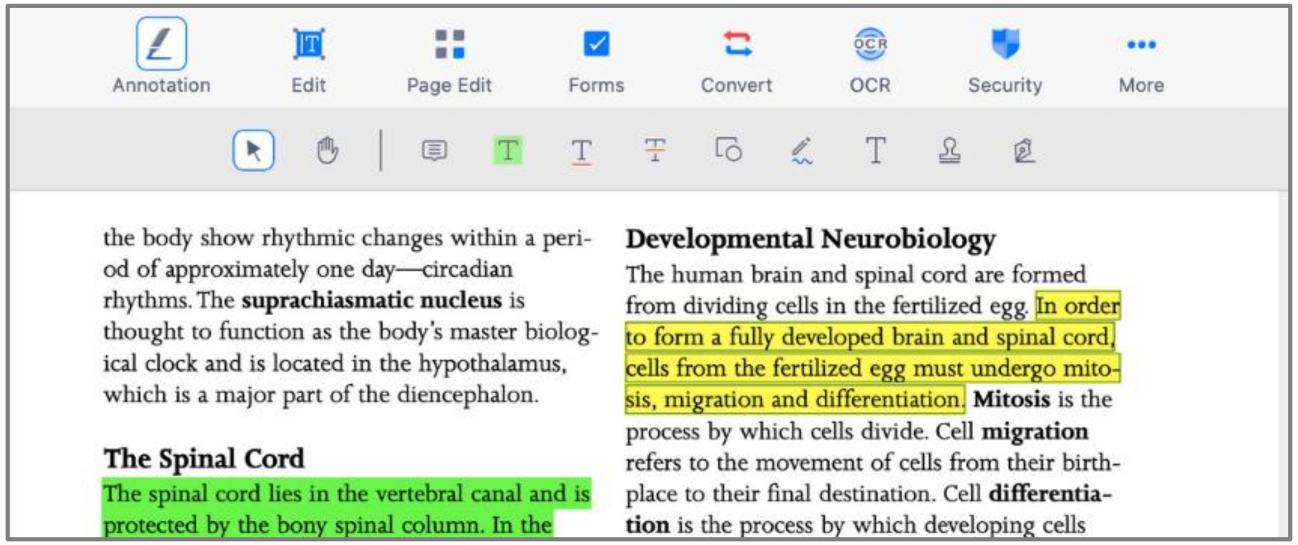
Agenda

- @ Uvod
- @ Primjena u Javi
- @ Dobra strana
- @ Loša strana
- @ Ružna strana
- @ Najbolje prakse

Uvod

Anotacije u bibliografiji

- anotacija (lat. annotatio) - bilješka
- Bilješka dodana naslovu, i/ili drugoj bibliografskoj obavijesti o dokumentu, kao objašnjenje bilo kojeg elementa bibliografskog opisa, ili pak elemenata sadržaja djela
- Anotacija može proširiti bibliografsku obavijest (npr. o prethodnim izdanjima, materijalnom obliku ili fiz. stanju dokumenta i sl.) ili opisati sadržaj, temu, predmet djela.



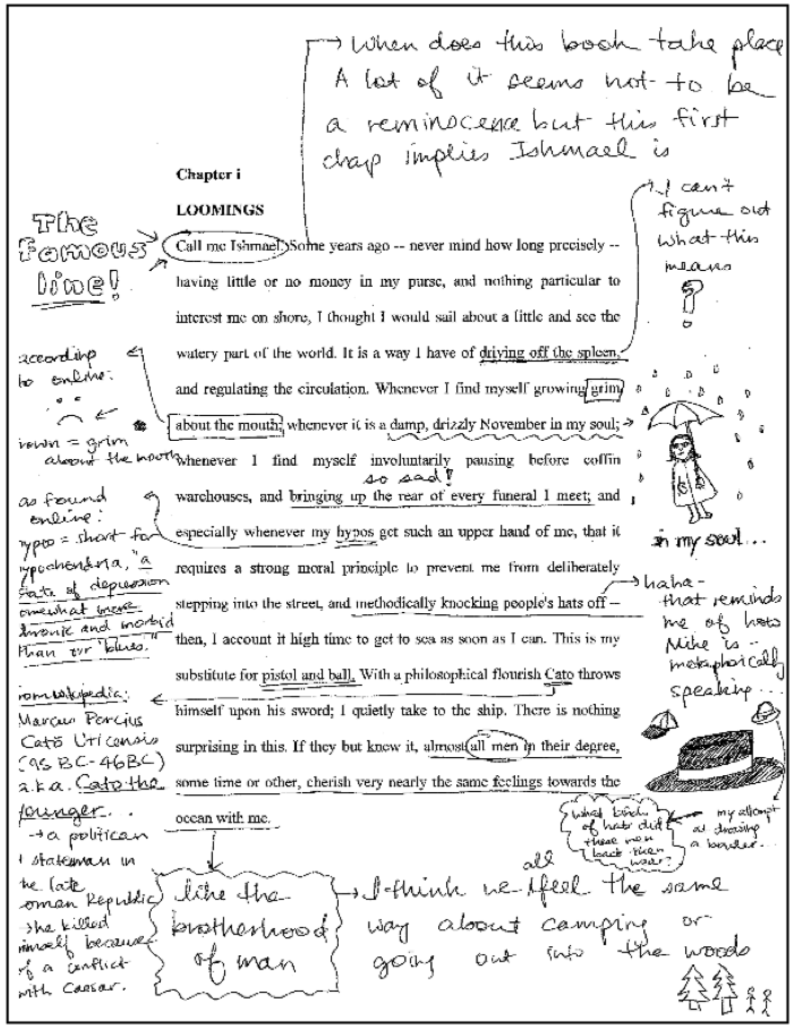
the body show rhythmic changes within a period of approximately one day—circadian rhythms. The **suprachiasmatic nucleus** is thought to function as the body's master biological clock and is located in the hypothalamus, which is a major part of the diencephalon.

The Spinal Cord

The spinal cord lies in the vertebral canal and is protected by the bony spinal column. In the

Developmental Neurobiology

The human brain and spinal cord are formed from dividing cells in the fertilized egg. In order to form a fully developed brain and spinal cord, cells from the fertilized egg must undergo mitosis, migration and differentiation. **Mitosis** is the process by which cells divide. Cell **migration** refers to the movement of cells from their birthplace to their final destination. Cell **differentiation** is the process by which developing cells



When does this book take place A lot of it seems not to be a reminocence but this first chap implies Ishmael is

Chapter i

LOOMINGS

The Famous line!

Call me Ishmael. Some years ago -- never mind how long precisely -- having little or no money in my purse, and nothing particular to interest me on shore, I thought I would sail about a little and see the watery part of the world. It is a way I have of driving off the spleen, and regulating the circulation. Whenever I find myself growing grim about the mouth; whenever it is a damp, drizzly November in my soul; whenever I find myself involuntarily pausing before coffin warehouses, and bringing up the rear of every funeral I meet; and especially whenever my hypos get such an upper hand of me, that it requires a strong moral principle to prevent me from deliberately stepping into the street, and methodically knocking people's hats off -- then, I account it high time to get to sea as soon as I can. This is my substitute for pistol and ball. With a philosophical flourish Cato throws himself upon his sword; I quietly take to the ship. There is nothing surprising in this. If they but knew it, almost all men in their degree, some time or other, cherish very nearly the same feelings towards the ocean with me.

I can't figure out what this means ?

according to online: ☹ rown = grim about the mouth

so sad!

in my soul...

as found online: hypo = short for hypochondria, "a state of depression omewhat more ironic and morbid than the 'blues.'"

haha - that reminds me of hats Mike is - metaphorically speaking...

rom Wikipedia: Marcus Porcius Cato Uticensis (95 BC - 46 BC) a.k.a. Cato the Younger... → a politican & statesman in the late Roman Republic → he killed himself because of a conflict with Caesar.

what kinds of hats did these men back then wear?

my attempt at drawing a bowler...

like the brotherhood of man

→ I think we all feel the same way about camping or going out into the woods

Anotacije u programskim jezicima

- @ slične anotacijama u lingvistici
- @ strukturni elementi koji sadrže dodatne informacije ili **meta-podatke**
- @ računalo ih u pravilu ignorira prilikom izvođenja
- @ jezici koji ih podržavaju: Java, C#, Python, Ruby and VB.NET
- @ agilne metodologije

Anotacije u programskim jezicima

Java

```java
/**
 * Manually shifts gears
 * @deprecated
 * This method is no longer acceptable.
 * <p> Use {@link Utils#automaticShift()} instead.
 */
@Deprecated
public void manualShift(){ }
```

C#

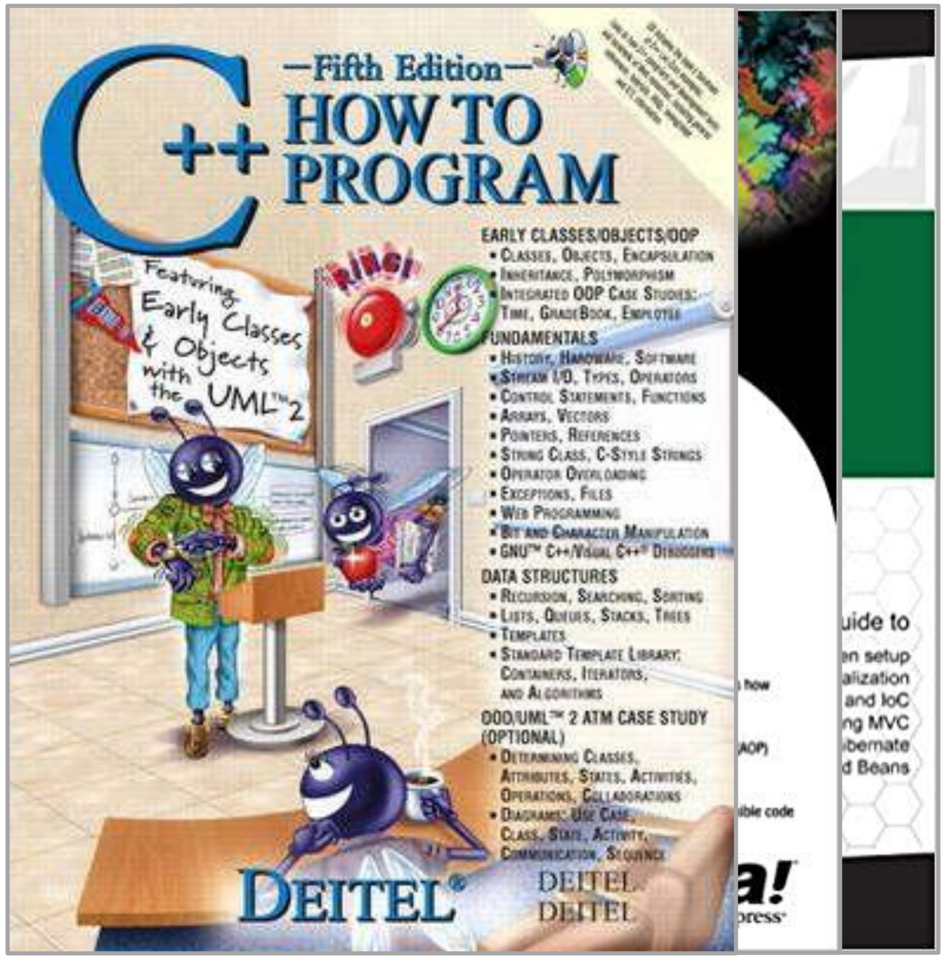
```csharp
[Obsolete("Use AutomaticShift() instead", true)]
public void ManualShift(){ }
```

VB

```vb
<Obsolete("Use AutomaticShift() instead")> _
Public Sub ManualShift()
End Sub
```

C++

```cpp
[[deprecated("Use AutomaticShift() instead")]]
void ManualShift() { }
```



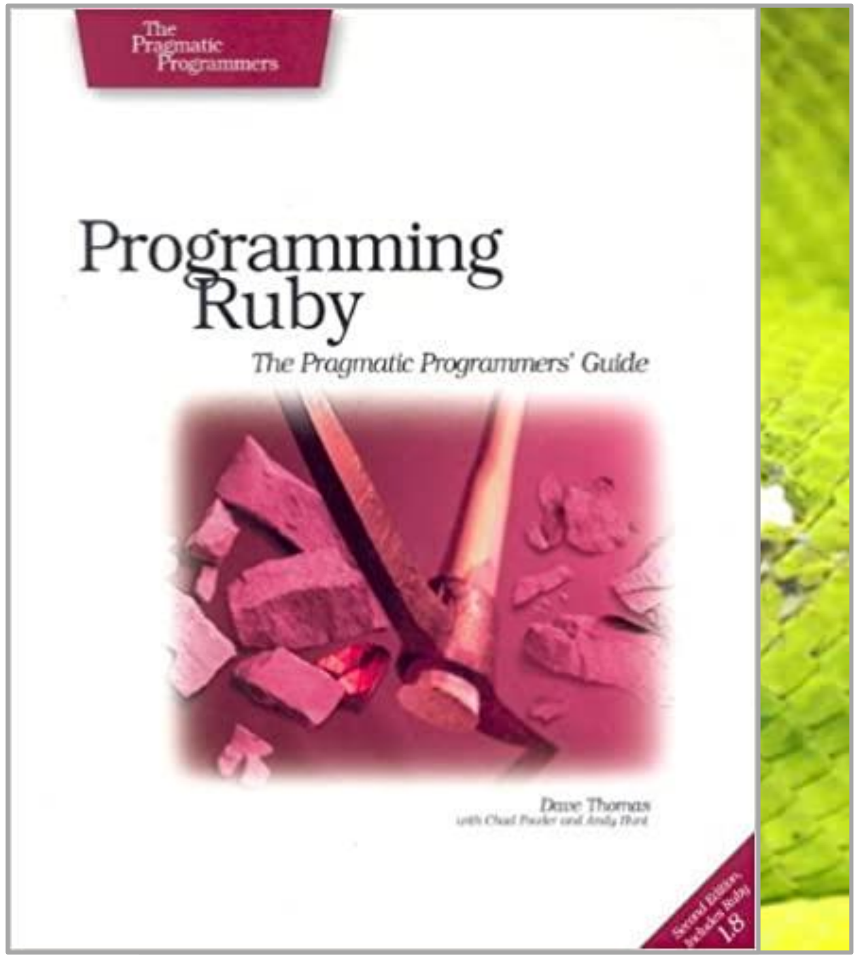
Anotacije u programskim jezicima

Python

```
def deprecated(message):
  def deprecated_decorator(func):
      def deprecated_func(*args, **kwargs):
           # some code…
      return deprecated_func
  return deprecated_decorator

from .utils import deprecated

@deprecated("Use method automatic_shift instead")
def manual_shift()"
 pass
```

Ruby

```
validate_range :@height, :with => 1..120
validate_range :@weight, :with => 1..1000
```


The Pragmatic Programmers
Programming Ruby
The Pragmatic Programmers' Guide
Dave Thomas
with Chad Fowler and Andy Hunt
Second Edition Includes Ruby 1.8

Meta-programiranje

- definicja = „writing code that writes code”
- uključuje:
  - Compile code generation or Runtime code generation (ili oboje)
  - aspektno orijentirano programiranje
  - DRY princip
- alati:
  - Anotacije, atributi, dekoratori
  - AOP
  - DSL, SpEL
  - Generics, refleksija

Anotacije u Javi

Anotacije u Javi

- Predstavljaju meta-podatak
- Pojavile su se sa verzijom 1.5
- Počinju karakterom **@**
- Mogu sadržavati elemente/atribute
- Moguće je koristiti više anotacija na istoj deklaraciji
- Korištenje istih tipova na jednoj deklaraciji (ponavljajuće anotacije)

Data

Metadata

name: Blondie
actor: Clint Eastwood
represents: the good
totem: cigarette

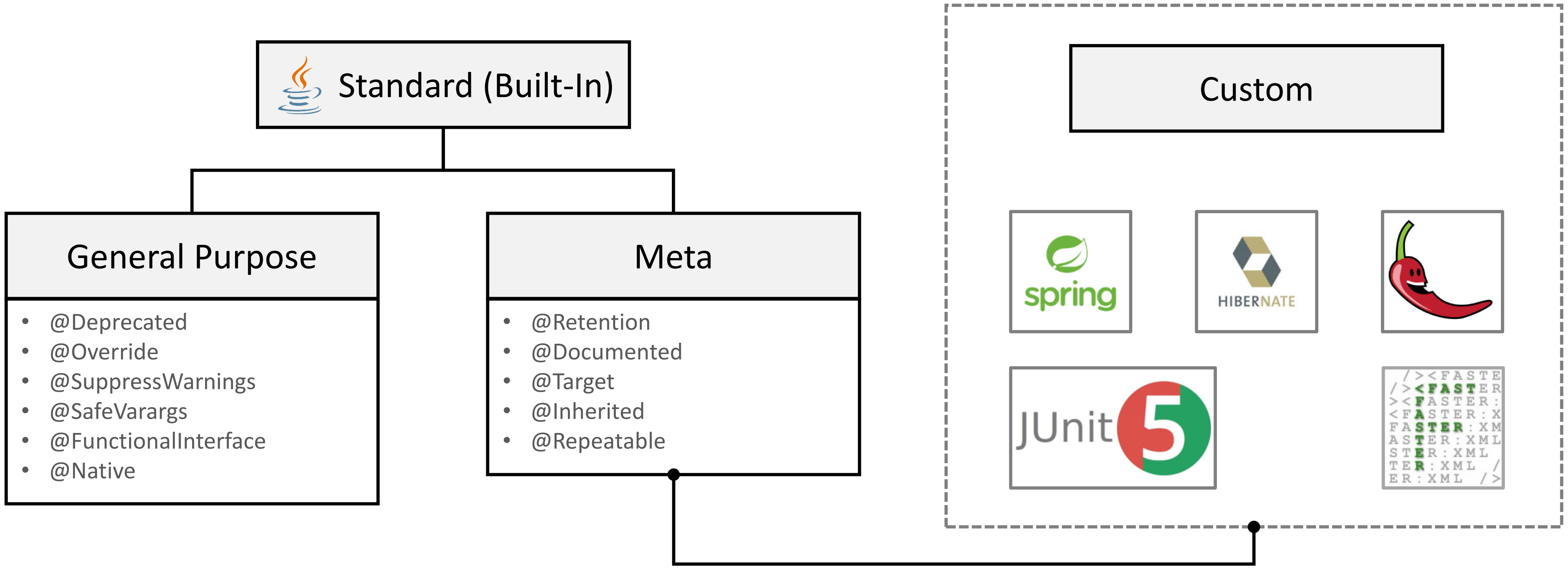
```
@Override
void mySuperMethod() { ... }
```

```
@Character(name = "Blondie", actor = "Clint Eastwood")
class MyClass { ... }

@Character("Blondie")
class MyClass { ... }
```

```
@Character(actor = "Clint Eastwood")
@Cowboy
class MyClass { ... }
```

```
@Schedule(dayOfMonth="last")
@Schedule(dayOfWeek="Fri", hour="23")
public void smokeCigarette() { ... }
```

Tipovi anotacija



Standard (Built-In)

General Purpose

- @Deprecated
- @Override
- @SuppressWarnings
- @SafeVarargs
- @FunctionalInterface
- @Native

Meta

- @Retention
- @Documented
- @Target
- @Inherited
- @Repeatable

Custom

Primjena anotacija - @Target

```java
@Component
public class NameProvider {

    @XMLJavaTypeAdapter(LocalDateAdapter.class)
    private LocalDate dob;
    private String name;

    @Autowired
    public NameProvider(@Value("${my.bla.name}") String name){
        this.name = name;
    }

    @JsonIgnore
    public String getName() {
        return this.name;
    }
}
```

```java
@Target(ElementType.TYPE)
@Retention(RetentionPolicy.RUNTIME)
public @interface Component {
    String value() default "";
}
```

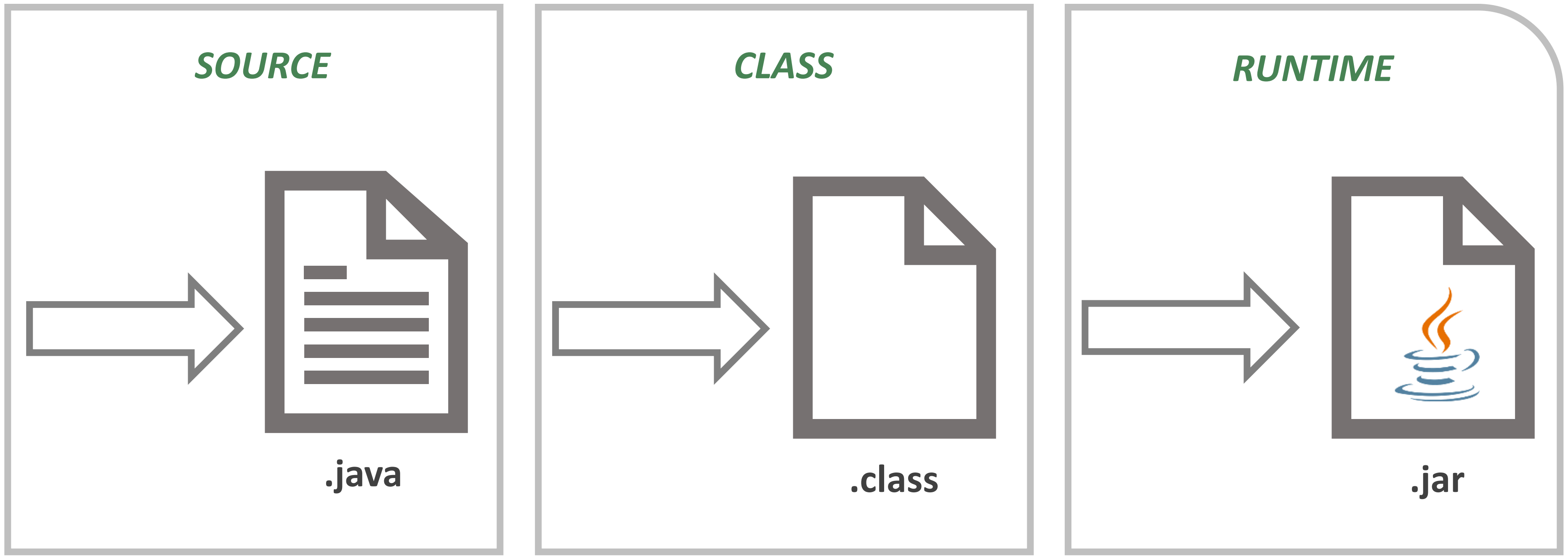
```java
@Target({ElementType.FIELD, ElementType.PARAMETER})
@Retention(RetentionPolicy.RUNTIME)
@Documented
public @interface Value {
    String value();
}
```

```java
@Target({ElementType.ANNOTATION_TYPE,
ElementType.METHOD, ElementType.CONSTRUCTOR,
ElementType.FIELD})
@Retention(RetentionPolicy.RUNTIME)
@JacksonAnnotation
public @interface JsonIgnore {
    boolean value() default true;
}
```

Primjenjuju se na deklaracije:

- klasa
- polja
- metoda

Politika izvršavanja - @Retention



- @Override
- @SuppressWarnings

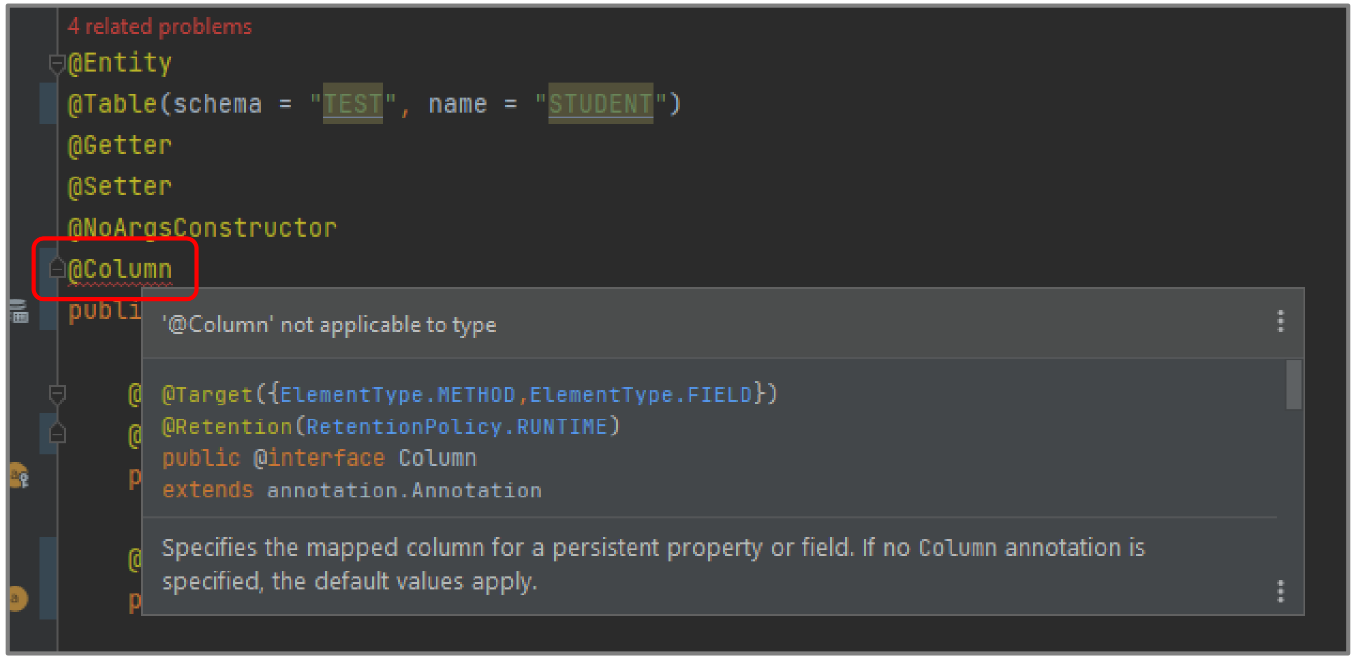
- @NonNull

Dobra strana

Zašto?

Static type checking
Smanjuje boilerplate code
Jasniji kod
Zamjenjuje XML konfiguraciju
the good

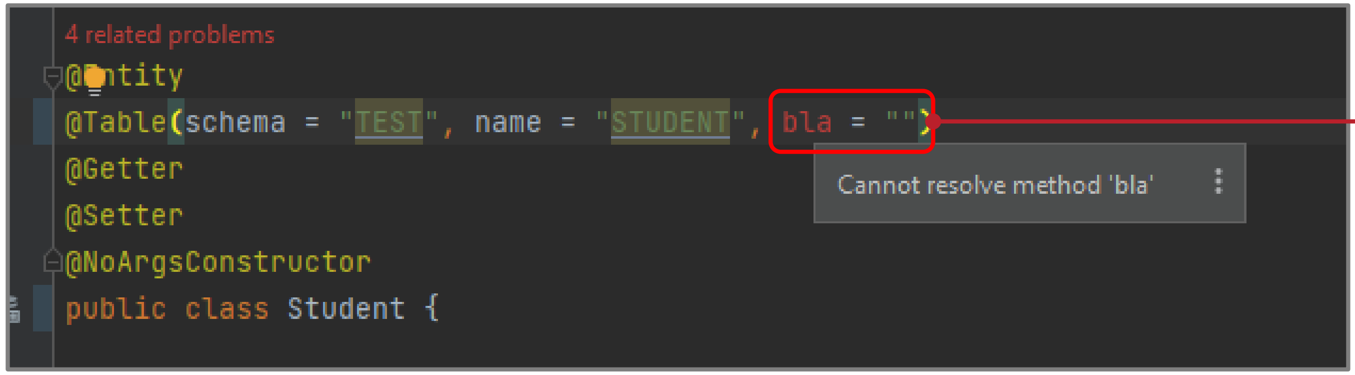
Static type checking



```
4 related problems
@Entity
@Table(schema = "TEST", name = "STUDENT")
@Getter
@Setter
@NoArgsConstructor
@Column
publi
```

```
'@Column' not applicable to type

@Target({ElementType.METHOD,ElementType.FIELD})
@Retention(RetentionPolicy.RUNTIME)
public @interface Column
extends annotation.Annotation
```

Specifies the mapped column for a persistent property or field. If no Column annotation is specified, the default values apply.

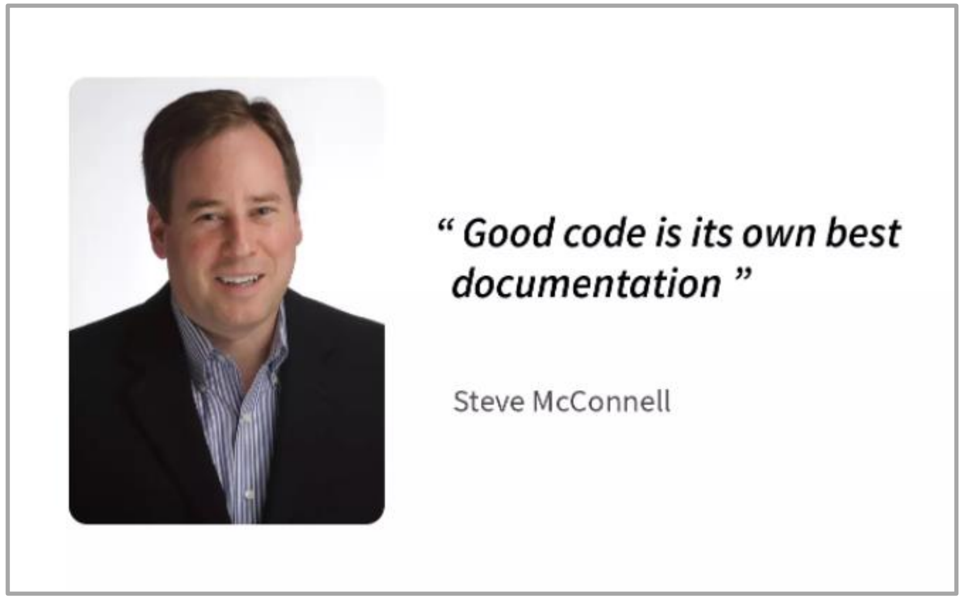


```
4 related problems
@Entity
@Table(schema = "TEST", name = "STUDENT", bla = "")
@Getter
@Setter
@NoArgsConstructor
public class Student {
```

```
Cannot resolve method 'bla'
```

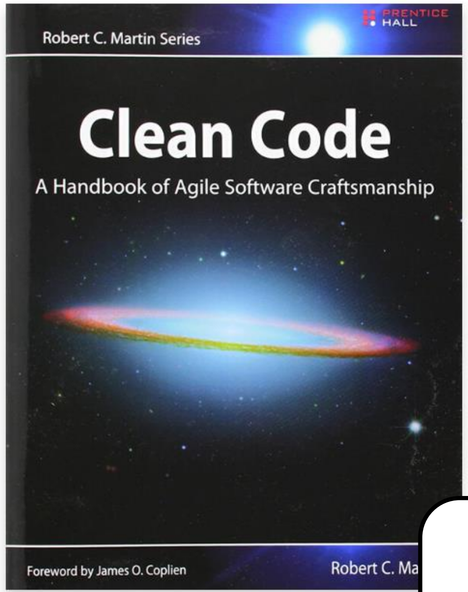
```
@Target({ElementType.TYPE})
@Retention(RetentionPolicy.RUNTIME)
public @interface Table {
    String name() default "";

    String catalog() default "";

    String schema() default "";

    UniqueConstraint[] uniqueConstraints()
default {};

    Index[] indexes() default {};
}
```

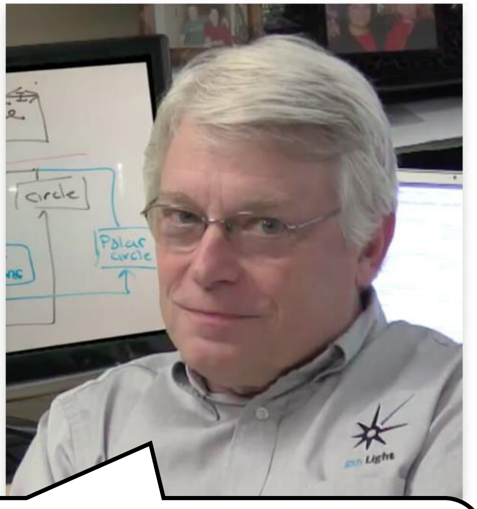
Jasniji kod



“ Good code is its own best documentation ”

Steve McConnell


Robert C. Martin Series
PRENTICE HALL
Clean Code
A Handbook of Agile Software Craftsmanship
Foreword by James O. Coplien
Robert C. Ma



„Clean code always looks like it was written by someone who cares.”

Karakteristike

- Jednostavnost (implementacije) 👍
- Lako održivo 👍
- Manje sklono greškama (testabilno)
- Čitljivo 👍

Principi

- SOLID
- DRY 👍
- KISS 👍

Jasniji kod - primjer

```java
public class Movie {

    @NotNull(message = "Name cannot be null")
    private String name;

    @AssertTrue
    private boolean released;

    @Size(min = 10, max = 200, message = "Description must be between 10 and 200 characters")
    private String description;

    @Min(value = 100, message = "Runtime should not be less than 100")
    @Max(value = 240, message = "Runtime should not be greater than 240")
    private int runtime;

    @PastOrPresent(message = "Release date should be in pas or present")
    private LocalDate releaseDate;

    // getters and setters
}
```

Boilerplate code - SpringBoot

Auto konfiguracija

```
@SpringBootApplication
public class EmployeeApplication {
    public static void main(String[] args) {}
}
```

Validacije

```
@NotNull(message = "Name cannot be null")
private String name;
```

Podrška za bazu

```
@Entity
public class City implements Serializable {
    @Id
    @GeneratedValue
    private Long id;

    @Column(nullable = false)
    private String name;

    // ... etc
}
```

Auto mapiranje kontrolera

```
@RestController
@RequestMapping("movie-rest")
public class SimpleMovieController {

    @GetMapping("/{id}"
    public Movie getMovie(@PathVariable int id){
        return findMovieById(id);
    }
}
```

Boilerplate code - Lombok

```java
@Getter
@RequiredArgsConstructor
@EqualsAndHashCode
@ToString
public class Movie {

    private final String name;
    private final String director;
    private final Integer year;
}
```

```java
@Getter
@Builder
public class Movie {
    private final String name;
    private final String director;
    private final Integer year;
}

Movie movie = Movie.builder()
  .name("The Good, the Bad and the Ugly")
  .director("Sergio Leone")
  .year(1966)
  .build();
```

Builder pattern

```java
public class Movie{
    private final String name;
    private final String director;
    private final Integer year;

    private Movie(MovieBuilder builder) {
        this.name = builder.name;
        this.director = builder.director;
        this.year = builder.year;
    }
    public String getName() {
        return name;
    }
    public String getDirector() {
        return director;
    }
    public Integer getYear() {
        return year;
    }
    @Override
    public String toString() {
        return "Movie: "+this.name+", "+this.director+", "+this.year;
    }
    public static class MovieBuilder {
        private final String name;
        private final String director;
        private Integer year;
        public MovieBuilder(String name) {
            this.name = name;
        }
        public MovieBuilder name(String name) {
            this.name = name;
            return this;
        }
        public MovieBuilder director(String director) {
            this.director = director;
            return this;
        }
        public MovieBuilder year(Integer year) {
            this.year = year;
            return this;
        }
        public Movie build() {
            return new Movie(this);
        }
    }
}
```

XML vs anotacije

XML vs anotacije - konfiguracija

XML

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
        xmlns:xsi=http://www.w3.org/2001/XMLSchema-instance
        xmlns:context=http://www.springframework.org/schema/context
    xsi:schemaLocation="http://www.springframework.org/schema/beans
    http://www.springframework.org/schema/beans/spring-beans.xsd
    http://www.springframework.org/schema/context
    http://www.springframework.org/schema/context/spring-context.xsd">

    <context:component-scan base-package="hr.anotacije.spring.basics"/>

    <bean id="theGoodBean" class="java.lang.String">
        <constructor-arg value="Blondie" />
    </bean>

    <bean id="theBadBean" class="java.lang.String">
        <constructor-arg value="Angel Eyes" />
    </bean>

    <bean id="theUglyBean" class="java.lang.String">
        <constructor-arg value="Tuco" />
    </bean>

</beans>
```

Anotacije

```
@Configuration
public class JavaConfiguration {
    @Bean
    public String theGoodBean() {
        return "Blondie";
    }
    @Bean
    public String theBadBean() {
        return "Angel Eyes";
    }
    @Bean
    public String theUglyBean() {
        return "Tuco";
    }
}
```

XML vs anotacije - security

XML

```
<http use-expressions="true">
  <intercept-url pattern="/**" access="authenticated"/>
  <logout
    logout-success-url="/login?logout"
    logout-url="/logout"
  />
  <form-login
    authentication-failure-url="/login?error"
    login-page="/login"
    login-processing-url="/login"
    password-parameter="password"
    username-parameter="username"
  />
</http>
<authentication-manager>
  <authentication-provider>
    <user-service>
      <user name="user"
          password="password"
          authorities="ROLE_USER"/>
    </user-service>
  </authentication-provider>
</authentication-manager>
```

Anotacije

```
@Configuration
@EnableWebSecurity
public class HelloWebSecurityConfiguration extends WebSecurityConfigurerAdapter {

  @Autowired
  public void configureGlobal(AuthenticationManagerBuilder auth) {
    auth
      .inMemoryAuthentication()
        .withUser("user").password("password").roles("USER");
  }
}
```

Loša strana

Zašto?

Zagađuju kod
Pisanje testova
Verifikacija parametara
Ograničeni tipovi i atributi
the bad

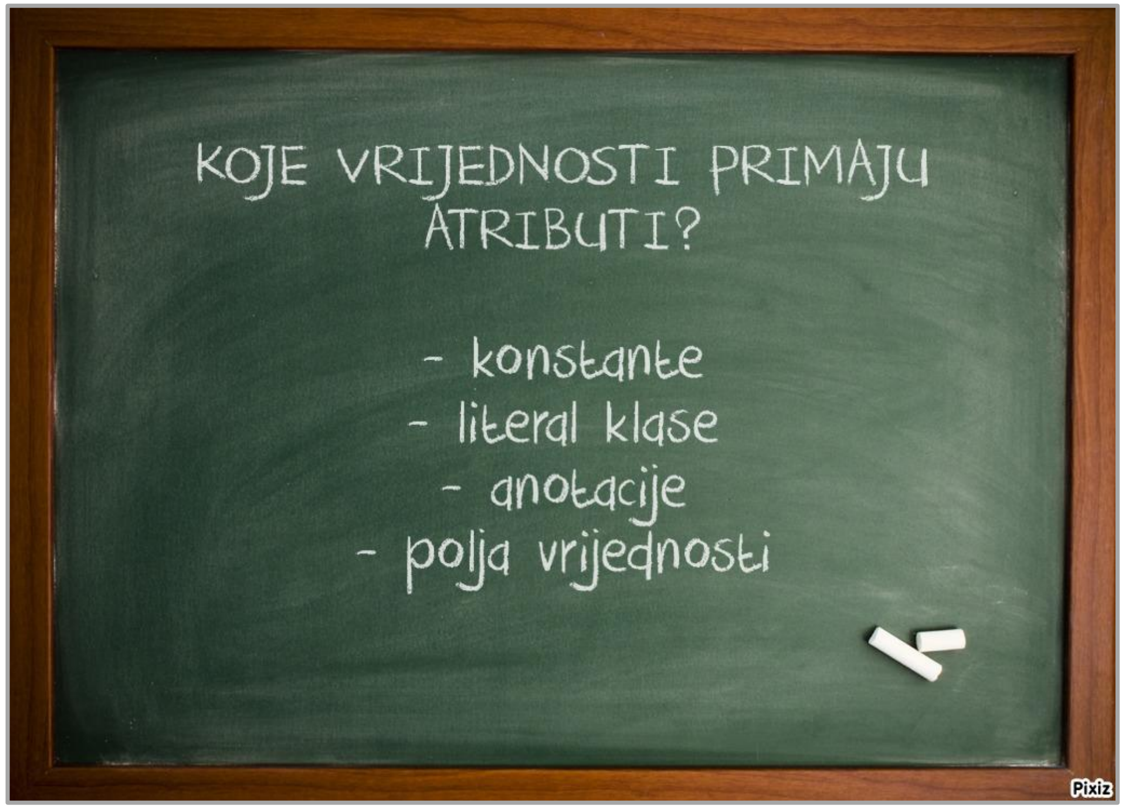
Ograničenje tipova

- Ne mogu sudjelovati u nasljeđivanju
- Metode ne mogu imati argumente
- Ne mogu biti generičke ili sadržavati **`throw`** klauzulu

- Mora vraćati jedan od tipova:
  - Primitivni tip
  - String
  - Klasa ili generička klasa
  - Enum
  - Anotacija
  - Polje (multidimenzionalni nisu dozvoljeni)

```java
@Target(ElementType.TYPE)
@Retention(RetentionPolicy.RUNTIME)
public @interface BugReport {
   enum Status { UNCONFIRMED, CONFIRMED, FIXED, NOTABUG };
   boolean showStopper() default false;
   String assignedTo() default "[none]";
   Class<?> testCase() default Void.class;
   Status status() default Status.UNCONFIRMED;
   Reference ref() default @Reference();
   String[] reportedBy();
}
```

Ograničenje atributa



Anotacija

```
@Target(ElementType.TYPE)
@Retention(RetentionPolicy.RUNTIME)
public @interface Marker {
    String value();
}
```

Ispravno

```
@Marker(Example.ATTRIBUTE_FOO + Example.ATTRIBUTE_BAR)
public class Example {
    static final String ATTRIBUTE_FOO = "foo";
    static final String ATTRIBUTE_BAR = "bar";

    // ...
}
```

Ograničenje atributa - primjeri

Static initializer

```
@Marker(Example.ATTRIBUTE_FOO)
public class Example {
    static final String[] ATTRIBUTES = {"foo", "Bar"};
    static final String ATTRIBUTE_FOO;

    static {
        ATTRIBUTE_FOO = ATTRIBUTES[0];
    }
}
```

```
@Marker(Example.ATTRIBUTE_FOO)
public class Example {
    static final String[] ATTRIBUTES = {"foo", "Bar"};
    static final String ATTRIBUTE_FOO = ATTRIBUTES[0];

    // ...
}
```

Array constant

```
@Marker(value = Example.ATTRIBUTES)
public class Example {
    static final String[] ATTRIBUTES = {"foo", "bar"};

    // ...
}
```

```
@Marker(Example.ATTRIBUTES[0])
public class Example {
    static final String[] ATTRIBUTES = {"Foo", "Bar"};
    // ...
}
```

```
@Marker(value = {"foo", "bar"})
public class Example {
    // ...
}
```

Verifikacija (String) parametara

Problemi

- Jezik unutar jezika koji koristi treći jezik za interpretaciju
- Validira se prilikom izvođenja

Reference na polja (SpEL)

```
@JournalDetails(subject = JournalSubEnum.VEHICLE, params = "#model")
public VehicleModel searchVehicle(VehicleSearchModel model) {

}
```

URL i parametri

```
@Path("vehicle/{vehicleId}")
public String getVehicle(@PathParam("vehicleId") vehicleId) {

}
```

SQL

```
@Query(
  value = "SELECT * FROM Vehicle ORDER BY id",
  countQuery = "SELECT count(*) FROM Vehicle",
  nativeQuery = true)
Page<User> findAllVehiclesWithPagination(Pageable pageable);
```

Zagađenje koda – nečitljivost #1

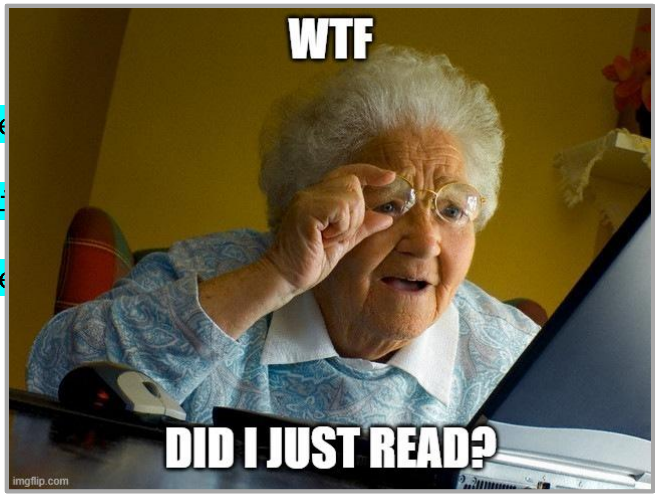
Stored procedures

```
@NamedStoredProcedureQuery(
    name = "searchUsers",
    procedureName = "SCHEMA.PKG_SEARCH.SEARCH_USER",
    resultSetMappings = "searchUsersMapping",
    parameters = {
        @StoredProcedureParameter(name = "P_USER", type = String.class, mode = ParameterMode.IN),
        @StoredProcedureParameter(name = "P_ORG_UNIT_ID", type = Long.class, mode = ParameterMode.IN),
        @StoredProcedureParameter(name = "P_STATUS", type = Long.class, mode = ParameterMode.OUT),
        @StoredProcedureParameter(name = "P_RESULT", type = void.class, mode = ParameterMode.REF_CURSOR)
    }
)
@SqlResultSetMapping(
    name = "searchUsersMapping",
    entities = @EntityResult(
        entityClass = UserResultSet.class,
        fields = {
            @FieldResult(name = "userName", column = "USERNAME"),
            @FieldResult(name = "firstName", column = "NAME"),
            @FieldResult(name = "lastName", column = "SURNAME"),
        }
    )
)
@Entity
@Getter
@ToString
public class UserResultSet {
    @Id
    private String userName;
    private String firstName;
    private String lastName;
}
```

Zagađenje koda – nečitljivost #2

REST api

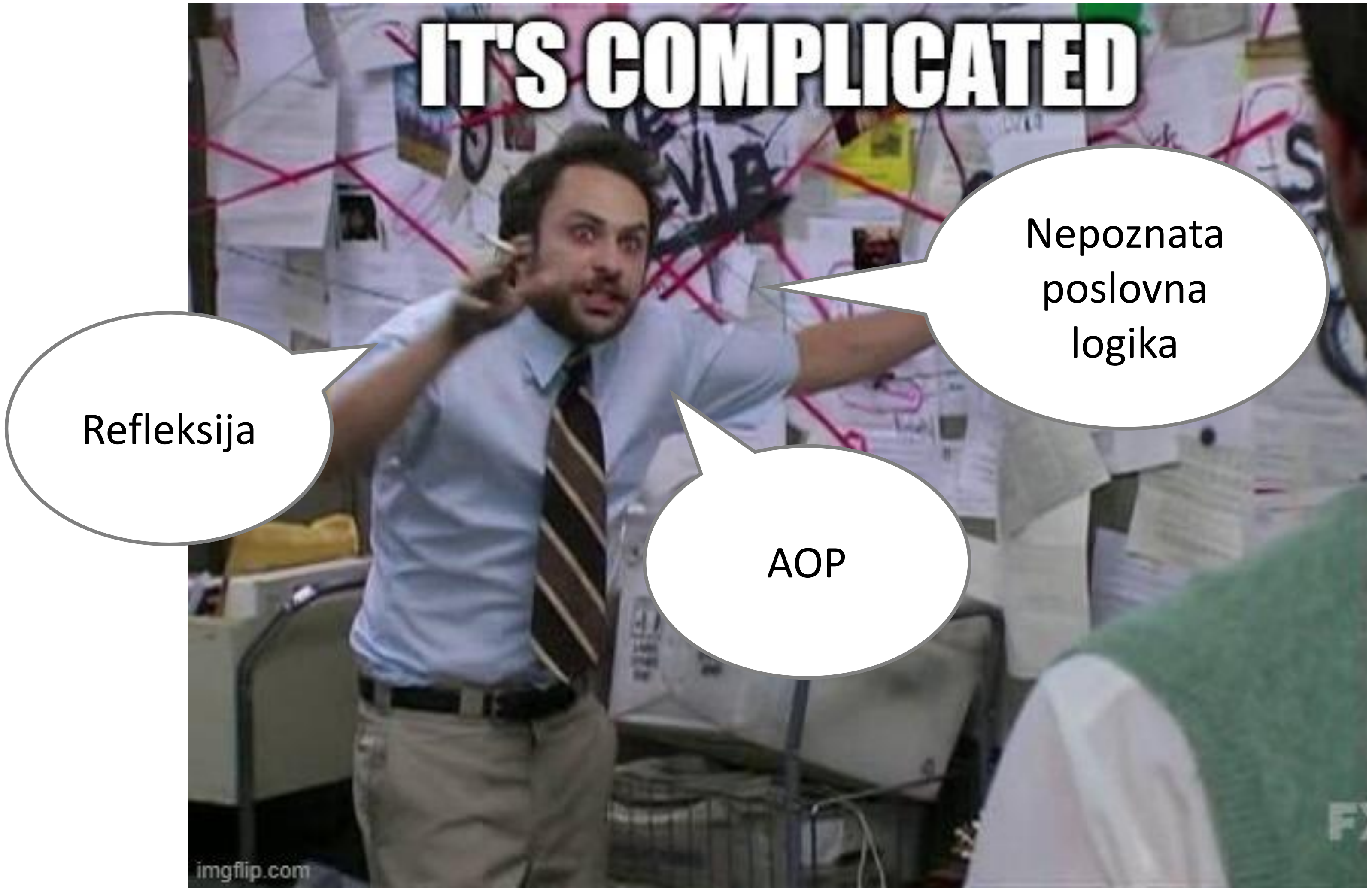
```
@ApiOperation(value = "Create a person object",
              notes = "Create a person object" +
                      "Return the newley created person object",
              response = Person.class)
@ApiResponses({
    @ApiResponse(code = HttpStatus.SC_INTERNAL_SERVER_ERROR, message = "Internal serve
    @ApiResponse(code = HttpStatus.SC_UNAUTHORIZED, message = "Unauthorized"),
    @ApiResponse(code = HttpStatus.SC_PRECONDITION_FAILED, message = "Precondition fa
    @ApiResponse(code = HttpStatus.SC_BAD_REQUEST, message = "Bad request"),
    @ApiResponse(code = HttpStatus.SC_UNPROCESSABLE_ENTITY, message = "Unprocessable e
})
@POST
@Path("rest/v1/persons")
@Consumes({MediaType.APPLICATION_JSON})
@Produces({MediaType.APPLICATION_JSON})
Person createPerson(
    @HeaderParam("SecurityToken") String token,
    @ApiParam(value = "person", defaultValue = "{ \"name\": = \"Bart Simpson\", \"age\": = 9 }") Person person);
```



Zagađenje koda - konfiguracija

```
@Configuration
@EnableIntegration
public class IntegrationConfig {
    @Bean
    ConcurrentMetadataStore metadataStore(DataSource dataSource,
                                          PlatformTransactionManager transactionManager,
                                          @Value("${app.spring-integration.table-prefix}") String tablePrefix,
                                          @Value("${app.spring-integration.region}") String region) {
        JdbcMetadataStore jdbcMetadataStore = new DuplicateKeyExceptionResistantJdbcMetadataStore(dataSource, transactionManager);
        jdbcMetadataStore.setTablePrefix(tablePrefix);
        jdbcMetadataStore.setRegion(region);
        return jdbcMetadataStore;
    }
    @Bean
    public JdbcChannelMessageStore channelMessageStore(DataSource dataSource,
                                                       @Value("${app.spring-integration.ta
                                                       @Value("${app.spring-integration.re
        JdbcChannelMessageStore jdbcChannelMessageStore = new JdbcChannelMessageStore();
        jdbcChannelMessageStore.setChannelMessageStoreQueryProvider(new OracleChannelMessa
        jdbcChannelMessageStore.setUsingIdCache(false);
        jdbcChannelMessageStore.setDataSource(dataSource);
        jdbcChannelMessageStore.setTablePrefix(tablePrefix);
        jdbcChannelMessageStore.setRegion(region);
        return jdbcChannelMessageStore;
    }
}
```

Pisanje testova i TDD

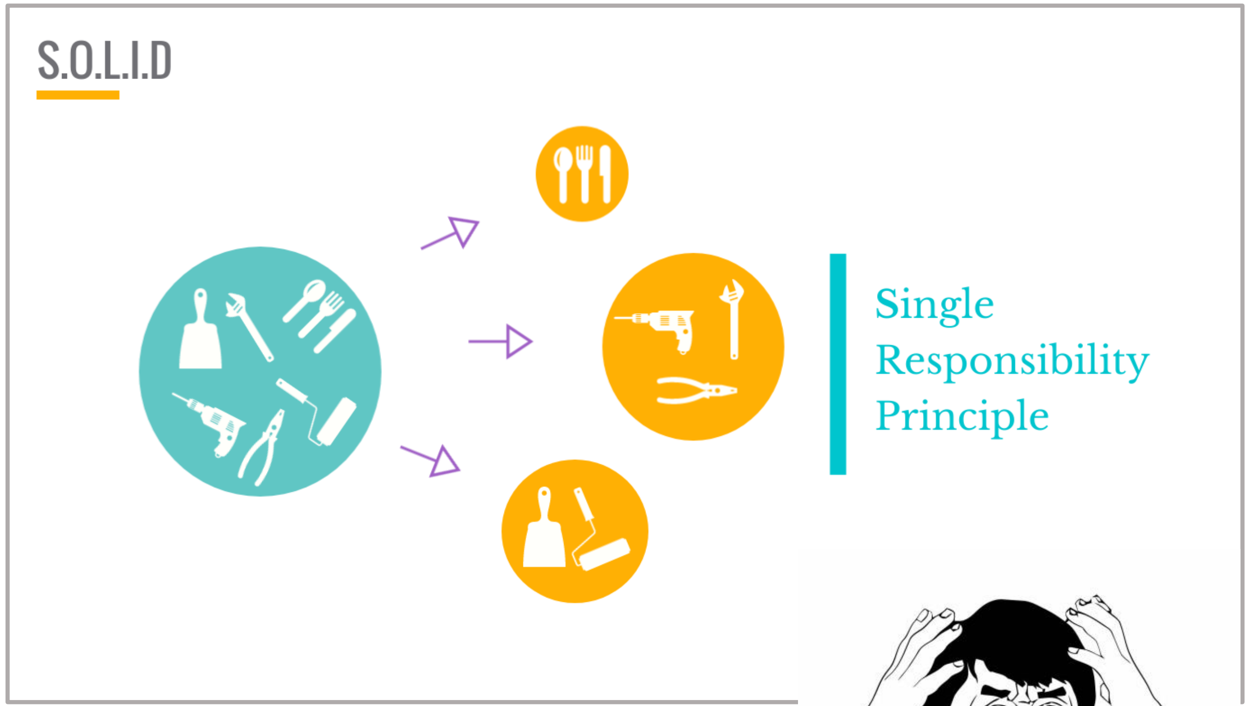


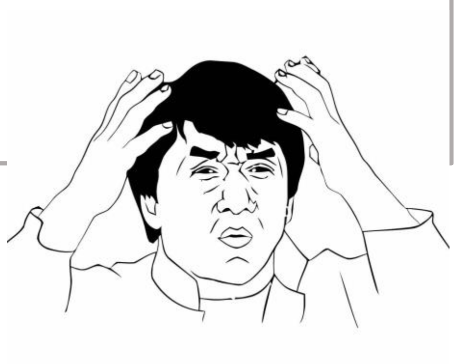
Ružna strana

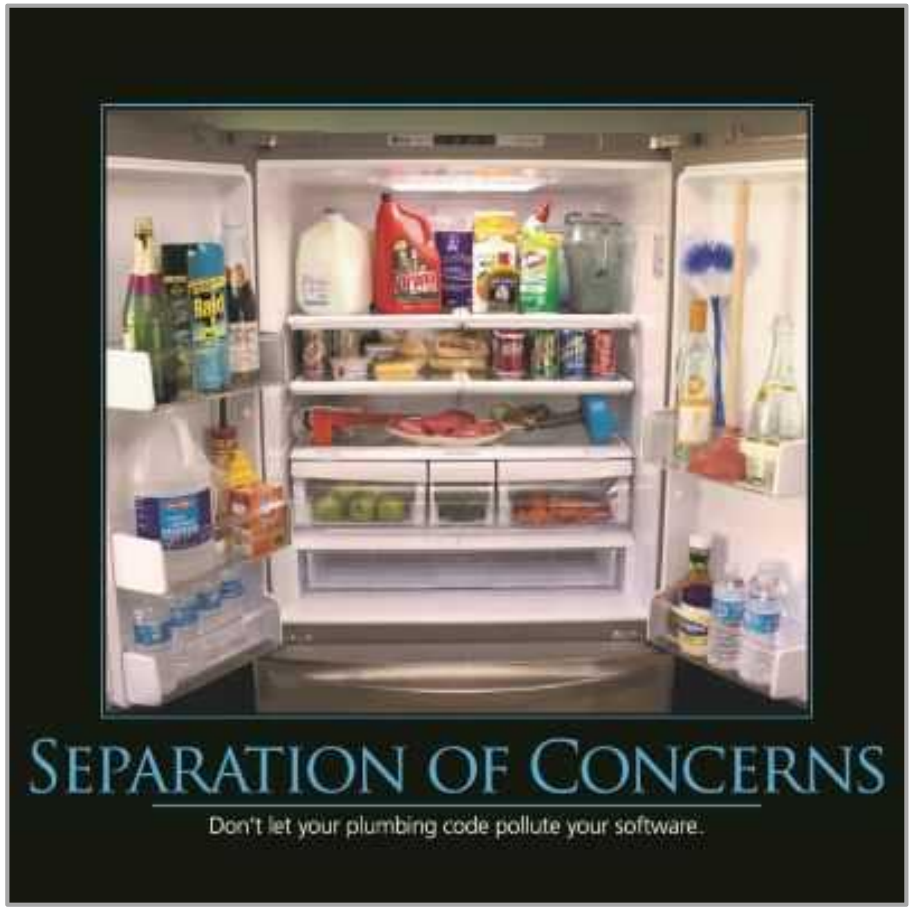
Zašto?

the ugly
Nepoznato djelovanje
Ne slijedi osnovne programske principe
Djeluju kao pošast

Ne slijedi (uvijek) osnovne programske principe







Prikriveni SOLID

```java
@Entity
public class Employee {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private long id;
    private String name;
    private int age;
    @ManyToOne
    private Department department;
    @OneToMany(mappedBy = "employee")
    private List<Phone> phones;

    // getters and setters...
}
```

- Ima ulogu DO-a/DTO-a čija je perzistentnost vezana za druge DTO-ove
- Ponaša se kao „black box”
- Loše surađuje s drugim anotacijama

```java
@Component
public class EmployeeFacade {

    @Autowired
    private EmployeeService service;
}
```

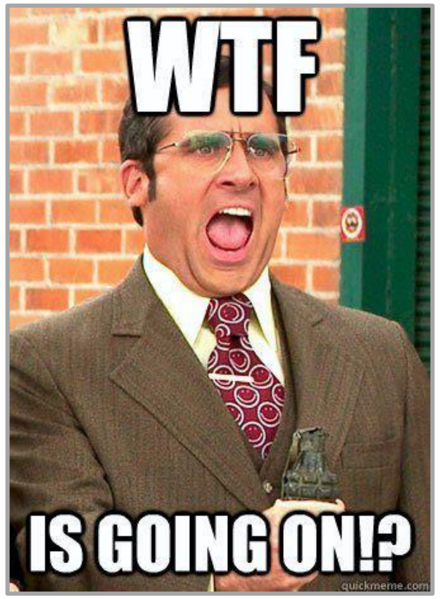
```java
@Component
public class EmployeeFacade {

    private final EmployeeService service;

    public EmployeeFacade(EmployeeService service) {
        this.service = service;
    }
}
```

- Prikriva „code smell”
- Otežana inicijalizacija klasa
- Stvara „tight coupling” sa DI kontejnerom

Nepoznato djelovanje

INPUT

OUTPUT



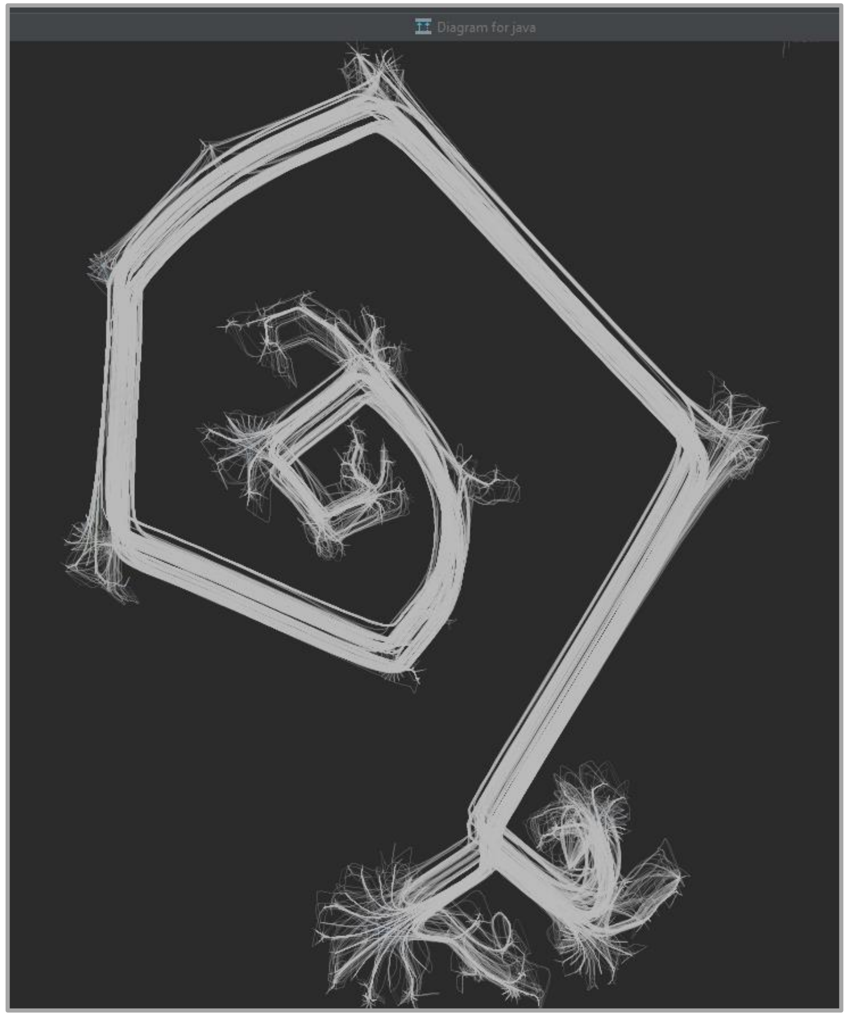
- Može mijenjati ponašanje tijekom izvođenja
- Skriva aspekte klase ili metode

Skriveni aspekti

```
@Aspect
@Configuration
public class JournalAspect {
    private final JournalService journalService;
    public JournalAspect(JournalService journalService) {
        this.journalService = journalService;
    }

    @AfterReturning(value = "@annotation(journalDetails))", returning = "retVal")
    public void journalDetailsAfterReturning(JoinPoint jp, JournalDetails journalDetails, Object retVal) {
        // extract params
        final JournalParams params = getJournalParams(jp, journalDetails.params());
        addParams(params, jp);
        this.journalService.logDetail(params, result);
    }
    private JournalParams getJournalParams(final JoinPoint jp, final String params) {
        return Optional.ofNullable(ReflectionUtils.getParams(jp, params))
                .map(JournalParamsDto.class::cast)
                .map(JournalParamsDto::toJournalParams)
                .orElseGet(JournalParams::new);
    }
    private void addParams(final JournalParams params, final JoinPoint jp) {
        Object[] methodArgs = jp.getArgs();
        int numArgs = methodArgs.length;
        MethodSignature methodSignature = (MethodSignature) jp.getSignature();
        Annotation[][] annotationMatrix = methodSignature.getMethod().getParameterAnnotations();
        for (int i = 0; i < numArgs; i++) {
            Annotation[] annotations = annotationMatrix[i];
            for (Annotation annotation : annotations) {
                if (annotation.annotationType() == JournalParam.class) {
                    params.put(((JournalParam) annotation).value(), methodArgs[i]);
                }
            }
        }
    }
}
```

Pošast #1 – kreiranje stabla ovisnosti

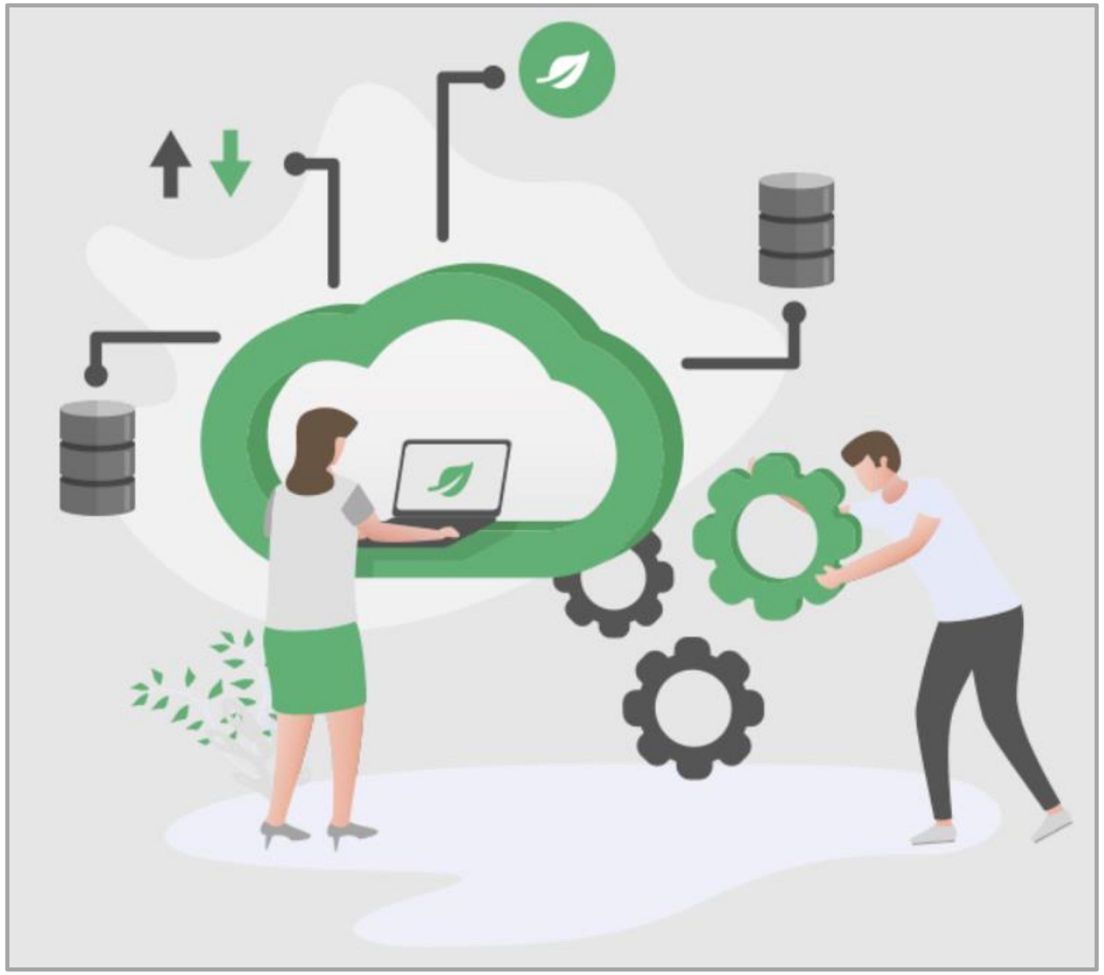
```
@Component
public class EmployeeFacade {
    @Autowired
    private EmployeeService service;
    @Autowired
    private CompanyService companyService;
}

@Service
public class EmployeeService {
    @Autowired
    private EmployeeRepository repository;
}

@Service
public class CompanyService {
    @Autowired
    private CompanyRepository repository;
}
```



- @Controller, @Service, @Component, @Repository
- Spring automatski detektira anotirane klase

Pošast #2 – Perzistiranje entiteta

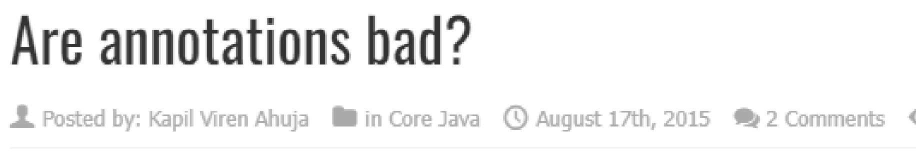
```
@Entity
@Table(name = "users")
public class User extends Worker {
    @OneToOne(cascade = CascadeType.ALL)
    @JoinColumn(name = "address_id", referencedColumnName = "id")
    private Address address;
}

@MappedSuperclass
public abstract class Worker { }

@Entity
@Table(name = "address")
public class Address {
    @OneToOne(mappedBy = "address")
    private User user;
}
```



Posljedice:

- Otežano korištenje „third party” objekata
- Strogo kontrolirano ili zatvoreno okruženje
- Otežava integracije

Dobre prakse


Are annotations bad?

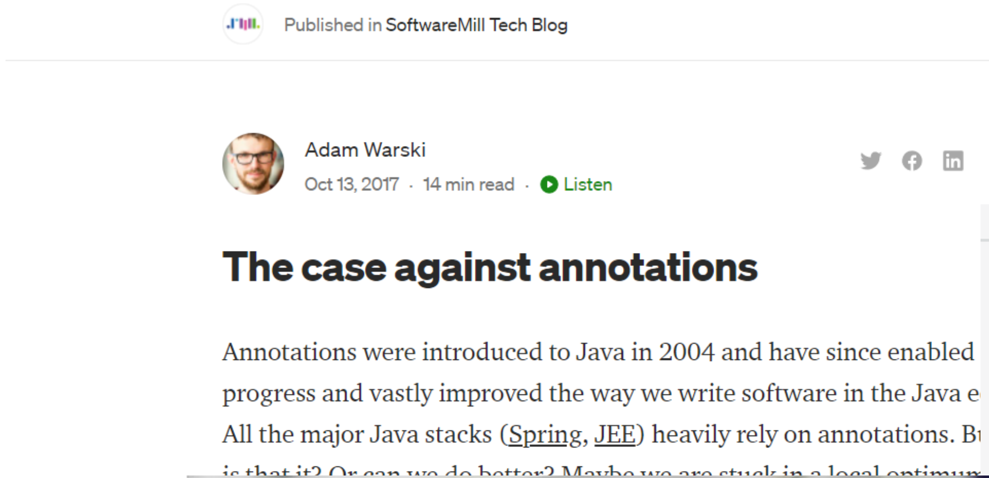
Posted by: Kapil Viren Ahuja in Core Java August 17th, 2015 2 Comments

Java Annotations Are a Big Mistake

12 April 2016 Seattle, WA 126 comments

java oop

Read more about this subject in

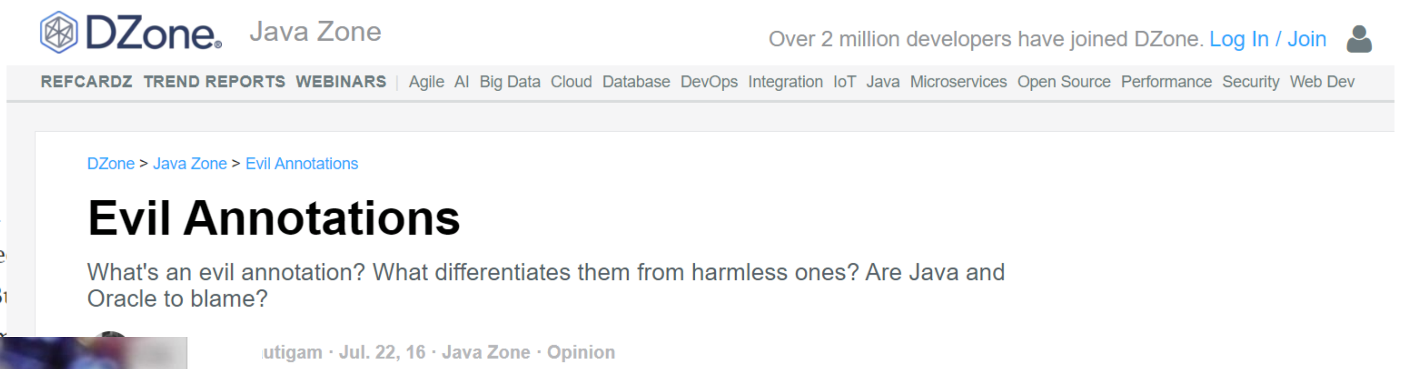

Published in SoftwareMill Tech Blog

Adam Warski

Oct 13, 2017 · 14 min read · Listen

The case against annotations

Annotations were introduced to Java in 2004 and have since enabled progress and vastly improved the way we write software in the Java e
All the major Java stacks (Spring, JEE) heavily rely on annotations. Bu
is that it? Or can we do better? Maybe we are stuck in a local optimum


DZone. Java Zone

Over 2 million developers have joined DZone. Log In / Join

REFCARDZ TREND REPORTS WEBINARS | Agile AI Big Data Cloud Database DevOps Integration IoT Java Microservices Open Source Performance Security Web Dev

DZone > Java Zone > Evil Annotations

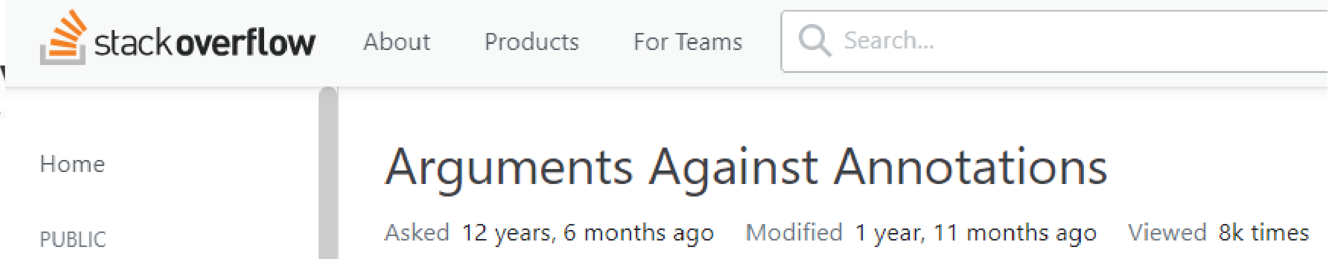
Evil Annotations

What's an evil annotation? What differentiates them from harmless ones? Are Java and Oracle to blame?

utigam · Jul. 22, 16 · Java Zone · Opinion

in Coding

Why using Spring's @
annotation is bad


stackoverflow About Products For Teams Search...

Home

PUBLIC

Arguments Against Annotations

Asked 12 years, 6 months ago Modified 1 year, 11 months ago Viewed 8k times

Preporuke za korištenje

1. Keep it simple, stupid
2. Don’t repeat yourself
3. Cross-cutting concerns
4. Be SOLID

Kada ne koristiti

Izmjene na podacima (baza)

Razmjena podataka (integracije)

Prikrivanje kompleksne poslovne logike

Mijenjanje ponašanja prilikom izvođenja

Zamjena za sučelje ili apstrakciju

Any fool can write code that a computer can understand. Good programmers write code that humans can understand.

— Martin Fowler —

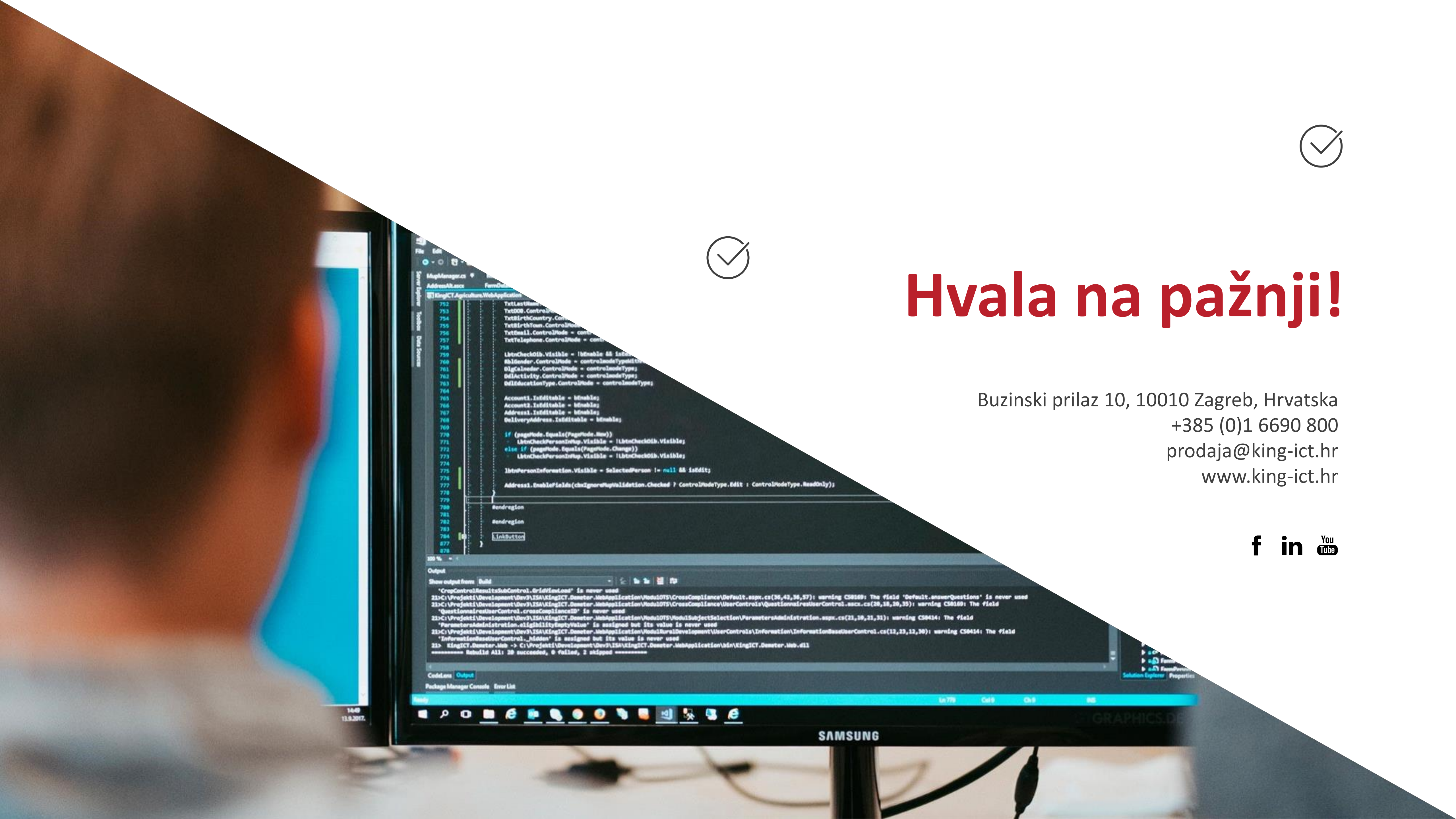
Literatura / Inspiracija

Knjige:

- Robert C. Martin - *Clean Code: A Handbook of Agile Software Craftsmanship*
- Martin Fowler - *Refactoring: Improving the Design of Existing Code*

Digitalni članci:

- baeldung - *Creating a Custom Annotation in Java* (Dostupno na: https://www.baeldung.com/java-custom-annotation [6.6.2021.])
- Yashwant Golecha - *How Do Annotations Work in Java?* (Dostupno na: https://dzone.com/articles/how-annotations-work-java [22.8.2019.])
- Adam Warski - *The case against annotations* (Dostupno na: https://blog.softwaremill.com/the-case-against-annotations-4b2fb170ed67 [13.10.2017.])
- Robert Brautigam - *Evil Annotations* (Dostupno na: https://dzone.com/articles/evil-annotations [22.7.2016.])
- Yegor Bugayenko - *Java Annotations Are a Big Mistake* (Dostupno na: https://www.yegor256.com/2016/04/12/java-annotations-are-evil.html [12.4.2016.])
- Kapil Viren Ahuja - *Are annotations bad?* (Dostupno na: https://www.javacodegeeks.com/2015/08/are-annotations-bad.html [17.8.2015.])



Hvala na pažnji!

Buzinski prilaz 10, 10010 Zagreb, Hrvatska
+385 (0)1 6690 800
prodaja@king-ict.hr
www.king-ict.hr