

**Akademia Górniczo-Hutnicza  
im. Stanisława Staszica w Krakowie**

---

Wydział Elektrotechniki, Automatyki, Informatyki i Elektroniki

KATEDRA AUTOMATYKI



**PRACA INŻYNIERSKA**

**KONRAD MALAWSKI**

**PROTODOC  
IMPLEMENTACJA ODPOWIEDNIKA NARZĘDZIA JAVADOC  
DLA JĘZKA DEFINICJI INTERFEJSÓW  
GOOGLE PROTOCOL BUFFERS**

PROMOTOR:

dr inż. Jacek Piwowarczyk

Kraków 2011

## **OŚWIADCZENIE AUTORA PRACY**

OŚWIADCZAM, ŚWIADOMY ODPOWIEDZIALNOŚCI KARNEJ ZA POŚWIADCZENIE NIEPRAWDY, ŻE NINIEJSZĄ PRACĘ DYPLOMOWĄ WYKONAŁEM OSOBIŚCIE I SAMODZIELNIE, I NIE KORZYSTAŁEM ZE ŹRÓDEŁ INNYCH NIŻ WYMIENIONE W PRACY.

.....

PODPIS

**AGH**  
**University of Science and Technology in Krakow**

---

Faculty of Electrical Engineering, Automatics, Computer Science and Electronics

DEPARTMENT OF AUTOMATICS



**BACHELOR OF SCIENCE THESIS**

**KONRAD MALAWSKI**

**PROTODOC**  
**DEVELOPMENT OF A JAVADOC TOOL EQUIVALENT FOR**  
**THE GOOGLE PROTOCOL BUFFERS**  
**INTERFACE DESCRIPTION LANGUAGE**

SUPERVISOR:

Jacek Piwowarczyk Ph.D

Krakow 2011

Chciałem serdecznie podziękować

## Spis treści

<b>1. Wprowadzenie .....</b>	<b>6</b>
1.1. Cele pracy .....	7
1.2. Zawartość pracy .....	7
1.3. Zrzuty ekranu wygenerowanej dokumentacji.....	7
<b>A. Podstawy języka Scala .....</b>	<b>8</b>
A.1. Krótka historia języka.....	8
A.2. Podstawy .....	8
A.3. Scala Parser Combinators .....	9
<b>B. Google Protocol Buffers .....</b>	<b>10</b>
B.1. Krótka historia języka.....	10
B.2. Przykładowe definicje wiadomości .....	10
B.3. Dostępne narzędzia.....	10



# 1. Wprowadzenie

## 1.1. Cele pracy

## 1.2. Zawartość pracy

## 1.3. Zrzuty ekranu wygenerowanej dokumentacji

### ProtoDoc

by Konrad Malawski  
konrad.malawski@java.pl

#### Table of Contents

Messages:

- [pl.project13.AmazingMessage](#)
- [pl.project13.AmazingMessage.E](#)
- [pl.project13.AmazingMessage.Ir](#)
- [pl.project13.AmazingMessage.S](#)
- [pl.project13.MessageWithInner](#)
- [pl.project13.MessageWithInner.I](#)
- [pl.project13.TopLevel](#)
- [pl.project13.TopLevel.MiddleLevel](#)
- [pl.project13.TopLevel.MiddleLevel](#)
- [pl.project13.WithEnum](#)
- [pl.project13.WithEnum.Message](#)

ProtoDoc (by Konrad Malawski) is Free Software, licensed under the Apache2 License. Want the sources? [Fork protodoc.github](#).

## M MessageWithInner

This is a simple Message which has some Inner Message defined Also note that it has a default value on the name property

defined in package: **pl.project13**

### Fields:

name
<p>This can be a name of your liking the default value is lorem ipsum etc</p> <p>Default value: loremipsum Modifier: required Tag: 2 Defined as: string Mapped to: java.lang.String</p>
<p>number</p> <p>A number is just a simple property</p> <p>Modifier: required Tag: 1 Defined as: int32 Mapped to: scala.Int</p>

### Inner Enums:

This message defines no enums.

### Inner messages:

[InnerMessage](#)

## 2. Budowa parsera

```
def messageTypeDef: Parser[ProtoMessageType] = opt(comment)
    ~ "message" ~ ID ~ "{"
    ~ rep(enumTypeDef | instanceField |
        messageTypeDef)
    ~ "}" ^^ {
case maybeDoc ~ m ~ id ~ p1 ~ allFields ~ p2 =>
    // utworzenie instancji ProtoMessageType
```



## A. Podstawy języka Scala

Celem tego dodatku jest przybliżenie czytelnikowi języka „Scala” aby w wystarczająco płynny sposób mógł czytać przykłady kodu używane w tym dokumencie.

### A.1. Krótka historia języka

Język Scala („Scalable Language”) najłatwiej jest przedstawić jako hybrydę dwóch znanych nurtów programowania: programowania obiektowego oraz funkcyjnego, wraz z powiązanymi z nimi językami programowania. Twórca języka Scala, Martin Oderski[Ode07]

Jako konkretnych „rodziców” można by wskazać:

- **Java** - jako reprezentant nurtu obiektowego
- oraz języki: **Haskell**, **SML** oraz pewne elementy języka **Erlang** (głównie *Actor model*).

### A.2. Podstawy

Ta sekcja służy przybliżeniu czytelnikowi języka *Scala* na poziomie wystarczającym aby swobodnie czytać przykłady kodu umieszczone w tej pracy. W niektórych przykładach pomijane są przypadki skrajne lub nietypowe, celem szybkiego oraz jasnego przedstawienia minimum wiedzy na temat języka aby móc swobodnie go „czytać”.

*Scala* jest językiem statycznie typowanym posiadającym lokalne „Type Inference”. Pozwala to kompilatorowi *scalac* na „odnajdywanie” typów wszystkich zmiennych oraz typów zwracanych przez metody podczas kompilacji, bez potrzeby definiowania ich wprost. System ten

Użycie nawiasów `()`, średnika `;` oraz kropki `.` jest analogiczne jak w przypadku Javy, jednak w wielu przypadkach opcjonalne gdyż kompilator jest w stanie wydedukować gdzie powinny się znaleźć.

```
val value = Option(42);  
val other = value.orElse(0);  
  
// może zostać zastąpione  
val value = Option(42)  
val other = value orElse 0
```

Jednym z ciekawych przykładów stosowania notacji bez nawiasów i kropek jest *ScalaTest*<sup>1</sup> (przy którego pomocy pisano testy w tym projekcie). Przykładowa *asercja* napisana w *DSL*u definiowanym przez tę bibliotekę wygląda następująco:

```
messages should (contain key ("Has") and not contain value ("NoSuchMsg"))
```

### A.3. Scala Parser Combinators

---

<sup>1</sup>ScalaTest - framework do testowania - <http://www.scalatest.org>

## B. Google Protocol Buffers

W tym dodatku zostanie omówiona idea oraz szczegóły implementacyjne stojące za Google Protocol Buffers.

### B.1. Krótka historia języka

### B.2. Przykładowe definicje wiadomości

### B.3. Dostępne narzędzia

```
message Person {  
  required int32 id = 1;  
  required string name = 2;  
  optional string email = 3;  
}
```

## **Bibliografia**

[Ode07] Martin Odersky. *Programming in Scala*. 2007.