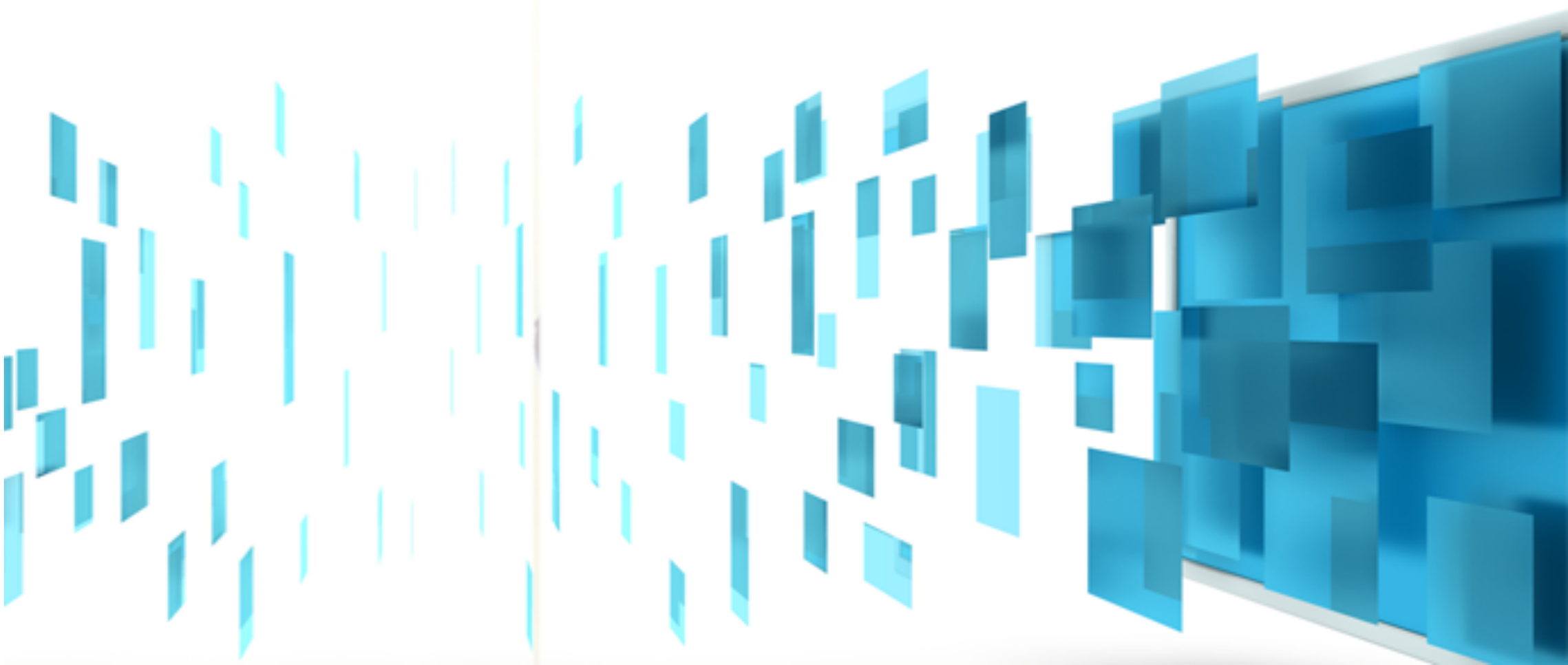
MINING & MODELING
UNSTRUCTURED DATA

Luca Ponzanelli, Andrea Mocci, Michele Lanza

REVEAL @ Faculty of Informatics
Università della Svizzera italiana



Luca **Ponzanelli**

Andrea **Mocci**

Michele **Lanza**

Tutorial
Speakers

Università
della
Svizzera
italiana

**Facoltà
di scienze
informatiche**

Luca **Ponzanelli**

Andrea **Mocci**

Michele **Lanza**

Tutorial
Speakers

Università
della
Svizzera
italiana

**Facoltà
di scienze
informatiche**

09:00 *Introduction*

09:45 *Foundations*

10:30 *Break*

11:00 *Hands On Session*

12:30 *Wrap-up and Q&A Session*

13:00 *End - Lunch Time*

Tutorial Outline

Who are **you**?

Tutorial
Audience


if(parameters.contains("name")){
hql += " and p.name = :name";
}
if(parameters.contains("age")){
hql += " and p.age = :age";
}
TypedQuery<person> query = em.createQuery(hql, p
if(parameters.contains("name")){
query.setParameter("name", values[0].toString
if(parameters.contains("age")){
query.setParameter("age", Integer.valueOf(v

Source
Code

Software Developer

Development Team

Developer
Community

Web
Community


THE OFFICIAL SEAL OF THE OCTOCAT
MELIUS SIMUL QUAM SOLUS
GITHUB
OF
MERITOCRACY
UNITED

GitHub


stackoverflow

Top Questions

interesting | 456 featured | hot | week | month

votes	answers	views		
0	0	2	visualize one class svm with multiple features in scikit	asked 13 secs ago AMisra 261
			python matplotlib svm	
0	0	2	I use react d3 basic bar chart	asked 13 secs ago Victoria 11
			d3.js reactjs bar-chart	
0	0	2	Excel VBA Evaluate with String in Forumla	asked 22 secs ago Reed Rawlings 113
			excel vba excel-vba double-quotes	
0	0	2	Lost in callback hell with Node, Express and google Analytics API	asked 31 secs ago Simon Breton 78
			javascript node.js asynchronous async-await	
0	0	2	Is it possible to create a `SQLReader` (like a `TextLineReader`) to read data from a database in TensorFlow?	asked 32 secs ago MiniQuark 12.7k
			database tensorflow reader	

Looking for a job?

(Junior) Javascript Entwickler (m/w) für den Standort Starnberg

estos GmbH — *Starnberg, Deutschland*

€42,000 - €55,000

css html

Top Questions

votes	answers	views	
0	0	2	visualize one cl… wi…
0	0	2	I use react… …hart…
0	0	2	Excel VBA F… …date wi…
0	0	2	Lost in callback hell with Node, Express and google Analytics API
0	0	2	Is it possible to create a `SQLReader` (like a `TextLineReader`) to read data from a database in TensorFlow?

python matplotli… svm

d3.js re…

excel vba excel-vba …ooth…

javascript node.js asynchronous async-await asked 31 secs ago Simon Breton 78

database tensorflow reader asked 32 secs ago MiniQuark 12.7k

…Read Rawlings 113

Android UI XML Layout

0

I've created textview layout into main.xml, i want to see effects onto emulator, but i did not get my changes, it shows old result when i run my app into emulator.....

eg:

```
<?xml version="1.0" encoding="utf-8"?>
<TextView xmlns:android="http://schemas.android.com/apk/res/android"
  android:id="@+id/textview"
  android:layout_width="fill_parent"
  android:layout_height="fill_parent"
  android:text="@string/hello"/>
```

strings:

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string name="hello">Hello, Android! I am a string resource!</string>
    <string name="app_name">Hello, Android</string>
</resources>
```

Please help if anyone know.....

android android-emulator

2 Answers

active | oldest | votes

0

You probaly need to set the contentView of your Activity using the setContentView() to point to the new layout

```
public class HelloAndroid extends Activity {
   /** Called when the activity is first created. */
   @Override
   public void onCreate(Bundle savedInstanceState) {
       super.onCreate(savedInstanceState);
       setContentView(R.layout.main);
   }
}
```

See more under "Upgrade the UI to an XML Layout"
http://developer.android.com/resources/tutorials/hello-world.html

share improve this answer

answered Oct 21 '10 at 7:51

Mads Lee Jensen
2,510 ● 3 ● 27 ● 44

hi folks, i'm new to android i've simple created a HelloAndroid application i want to try with xml layout. i've done it through developer guideline but after running the app it does not show the content that i've written into string.xml – abhay Oct 21 '10 at 8:08

add a comment

0

If you want dynamically change the text in your text view you need:

- set the content view for your activity:

```
setContentView(R.layout.main);
```

- get a reference to the TextView object:

```
TextView tv= (TextView)findViewById(R.id.textview);
```

- assing a new value:

```
tv.setText(R.string.app_name);
```

share improve this answer

edited Oct 21 '10 at 9:01

answered Oct 21 '10 at 8:19

Asahi

8,060 ● 7 ● 48 ● 70

Could you tell me the meaning of "v." in this line TextView tv= (TextView)v.findViewById(R.id.textview); ? – abhay Oct 21 '10 at 8:56

oops, missed it. meant to write just findViewById(). – Asahi Oct 21 '10 at 9:01

add a comment

Stack Overflow

Because of problems with "argoHome" location, I imported
java.net.URLDecoder and added the line
URLDecoder.decode(argoHome); just below this one :
public void loadModulesFromDir(String dir) in ModuleLoader.java.

Another problem in ModuleLoader :
since the cookbook explains how to make a PluggableDiagram (that's exactly what I am doing, so I extend this class), I have not found where the JMenuItem returned by method getDiagramMenuItem() in PluggableDiagram is attached in Argo menus. It seems this is not yet implemented, even though PluggableDiagrams implements Diagram!

So I have added those lines :
void append(PluggableDiagram aModule) {
ProjectBrowser.TheInstance
.appendPluggableDiagram((PluggableDiagram)aModule); }

Of course, such modifications must be reflected in ProjectBrowser.

Because of problems with "argoHome" location, I imported java.net.URLDecoder and added the line URLDecoder.decode(argoHome); just below this one : public void loadModulesFromDir(String dir) in ModuleLoader.java.

Another problem in ModuleLoader : since the cookbook explains how to make a PluggableDiagram (that's exactly what I am doing, so I extend this class), I have not found where the JMenuItem returned by method getDiagramMenuItem() in PluggableDiagram is attached in Argo menus. It seems this is not yet implemented, even though PluggableDiagrams implements Diagram!

So I have added those lines :

```
void append(PluggableDiagram aModule) {
ProjectBrowser.TheInstance
.appendPluggableDiagram((PluggableDiagram)aModule); }
```

Of course, such modifications must be reflected in ProjectBrowser.

Mining
Unstructured Data


State
of the Art

2016 IEEE/ACM 38th IEEE International Conference on Software Engineering

Augmenting API Documentation with Insights from Stack Overflow

Christoph Treude
School of Computer Science
University of Adelaide
Adelaide, SA, Australia
christoph.treude@adelaide.edu.au

Martin P. Robillard
School of Computer Science
McGill University
Montréal, QC, Canada
martin@cs.mcgill.ca

ABSTRACT

Software developers need access to different kinds of information which is often dispersed among different documentation sources, such as API documentation or Stack Overflow. We present an approach to automatically augment API documentation with "insight sentences" from Stack Overflow—sentences that are related to a particular API type and that provide insight not contained in the API documentation of that type. Based on a development set of 1,574 sentences, we compare the performance of two state-of-the-art summarization techniques as well as a pattern-based approach for insight sentence extraction. We then present SISE, a novel machine learning based approach that uses as features the sentences themselves, their formatting, their question, their answer, and their authors as well as part-of-speech tags and the similarity of a sentence to the corresponding API documentation. With SISE, we were able to achieve a precision of 0.64 and a coverage of 0.7 on the development set. In a comparative study with three state of the art developers, we found that SISE resulted in the highest number of sentences that

not obvious where a particular piece of information is stored. Different documentation formats, such as wikis or blogs, contain different kinds of information, written by different individuals and intended for different purposes [38]. For instance, API documentation captures information about functionality and structure, but lacks other types of information, such as concepts or purpose [18]. Some of the most severe obstacles faced by developers learning a new API are related to its documentation [32], in particular because of scarce information about the API's design, rationale [31], usage scenarios, and code examples [32].

On the other hand, "how-to" questions [35] (also referred to as "how-to-do-it" questions [10]) are the most frequent question type on the popular Question and Answer (Q&A) site Stack Overflow, and the answers to these questions have the potential to complement API documentation in terms of concepts, purpose, usage scenarios, and code examples. While a lot of research has focused on finding code examples for APIs (e.g., [17], [33]), less work has been conducted on improving or augmenting the natural language descriptions contained in API documentation.

Table 2: Regular Expressions for filtering Stack Overflow Threads

question part	pattern
body	`.*(^|[a-z]+ |[\.!?] |[\(<])TypeName([>\)\.,!?$]| [a-z]+).*` *(non-qualified API type surrounded by lower case words or punctuation marks)*
title or body	`(?i).*\bPackageName\.TypeName\b.*` *(fully-qualified API type, case-insensitive)*
body	`.*<code>.*\bTypeName\b.*</code>.*` *(non-qualified API type in code)*
body	`.*<a.*href.*PackageName/TypeName\.html.*>.*</a>.*` *(link to the official API documentation)*
title	`(?i).*\b(a |an )TypeName\b.*` *(non-qualified API type prefixed with "a" or "an", case-insensitive)*

State of the Art
Regular Expressions

Table 2: Regular Expressions for filtering Stack Overflow Threads

question part	pattern
body	`.*(^|[a-z]+ |[\.!?] |[\(<])TypeName([>\)\.,!?$]| [a-z]+).*` *(non-qualified API type surrounded by lower case words or punctuation marks)*
title or body	`(?i).*\bPackageName\.TypeName\b.*` *(fully-qualified API type, case-insensitive)*
body	`.*<code>.*\bTypeName\b.*</code>.*` *(non-qualified API type in code)*
body	`.*<a.*href.*PackageName/TypeName\.html.*>.*</a>.*` *(link to the official API documentation)*
title	`(?i).*\b(a |an )TypeName\b.*` *(non-qualified API type prefixed with "a" or "an", case-insensitive)*

State of the Art
Regular Expressions

srcML (sõrs em el), *n*. **1.** an infrastructure for the exploration, analysis, and manipulation of source code. **2.** an XML format for source code. **3.** a lightweight, highly scalable, robust, multi-language parsing tool to convert source code into srcML. **4.** a free software application licensed under GPL.

srcML is supported in part by a grant from the National Science Foundation (CNS 13-05292/05217) and is directed by Principal Investigators Dr. Michael L. Collard and Dr. Jonathan I. Maletic. The three year grant (July 2013 - June 2016) is for the enhancement and maintenance of srcML. The goal is to provide a more robust research infrastucture for the exploration, analysis, and manipulation of large scale software systems.

News:

- Latest versions of the srcML Tools srcSAX and srcSlice posted, June 30, 2015.
- New beta version 0.9.5 of srcML posted, May 20, 2015.
- srcML Technical Briefing at ICSE 2015 in Florence, Italy, May 18-19.
- srcSAX version posted Oct 2, 2014.
- One of the first papers on srcML published at IWPC 2003 and entitled "An XML-Based Lightweight C++ Fact Extractor" by M. L. Collard, H. Kagdi, and J. I. Maletic received the *Most Influential Paper Award* at ICPC 2013 in San Francisco, California, May 20-21.

State of the Art
<srcML>

<srcML>

Home | Download | About | Tools | Documentation | Tutorials | Support | Community

srcML (sõrs em el), *n*. **1.** an infrastructure for the exploration, analysis, and manipulation of source code. **2.** an XML format for source code. **3.** a lightweight, highly scalable, robust, multi-language parsing tool to convert source code into srcML. **4.** a free software application licensed under GPL.

srcML is supported in part by a grant from the National Science Foundation (CNS 13-05292/05217) and is directed by Principal Investigators Dr. Michael L. Collard and Dr. Jonathan I. Maletic. The three year grant (July 2013 - June 2016) is for the enhancement and maintenance of srcML. The goal is to provide a more robust research infrastucture for the exploration, analysis, and manipulation of large scale software systems.

Demo

News:

- Latest versions of the srcML Tools srcSAX and srcSlice posted, June 30, 2015.
- New beta version 0.9.5 of srcML posted, May 20, 2015.
- srcML Technical Briefing at ICSE 2015 in Florence, Italy, May 18-19.
- srcSAX version posted Oct 2, 2014.
- One of the first papers on srcML published at IWPC 2003 and entitled "An XML-Based Lightweight C++ Fact Extractor" by M. L. Collard, H. Kagdi, and J. I. Maletic received the *Most Influential Paper Award* at ICPC 2013 in San Francisco, California, May 20-21.

KENT STATE UNIVERSITY

The University of Akron

State of the Art
<srcML>

Island Grammars

Island Grammars

Generating Robust Parsers using Island Grammars*

Leon Moonen

CWI, P.O. Box 94079
1090 GB Amsterdam, The Netherlands
http://www.cwi.nl/~leon/
leon@cwi.nl

Abstract

Source model extraction—the automated extraction of information from system artifacts—is a common phase in reverse engineering tools. One of the major challenges of this phase is creating extractors that can deal with irregularities in the artifacts that are typical for the reverse engineering domain (for example, syntactic errors, incomplete source code, language dialects and embedded languages).

This paper proposes a solution in the form of island grammars, *a special kind of grammars that combine the detailed specification possibilities of grammars with the liberal behavior of lexical approaches. We show how island grammars can be used to generate robust parsers that combine the accuracy of syntactical analysis with the speed, flexibility and tolerance usually only found in lexical analysis. We conclude with a discussion of the development of* MANGROVE, *a generator for robust parsers based on island grammars and describe practical experience and case studies.*

Keywords and phrases: Island grammars, parser generation

These artifacts typically contain *irregularities* that make it hard (or even impossible) to parse the code using common parser based approaches. Our goal is to obtain *robust* parsers that can handle artifacts with such irregularities. Examples of the kind of irregularities we want to deal with include:

Syntax errors: In a program maintenance environment, we want to be able to deal with systems containing syntax errors (e.g., browse or query code to fix those errors). Most parser based techniques will fail when encountering syntactic errors.

Completeness: The source code of a system may be incomplete. A typical situation is that some of the header files (or copybooks) of a system are lost or mutilated over the years, making a full reconstruction impossible.

Dialects: A legacy language like COBOL (but also a language like C) has a large number of, slightly different, vendor-specific dialects. Ideally, we can support them all. However, a parser for one dialect may not accept code written in another.

Embedded languages: Several programming languages have been upgraded with embedded languages for database access,

```
IslandGrammar:
  ( Island | Water )*

Island:
  JavaFragment | JSONFragment | ...

Water:
    ANY+
```

Island Grammars

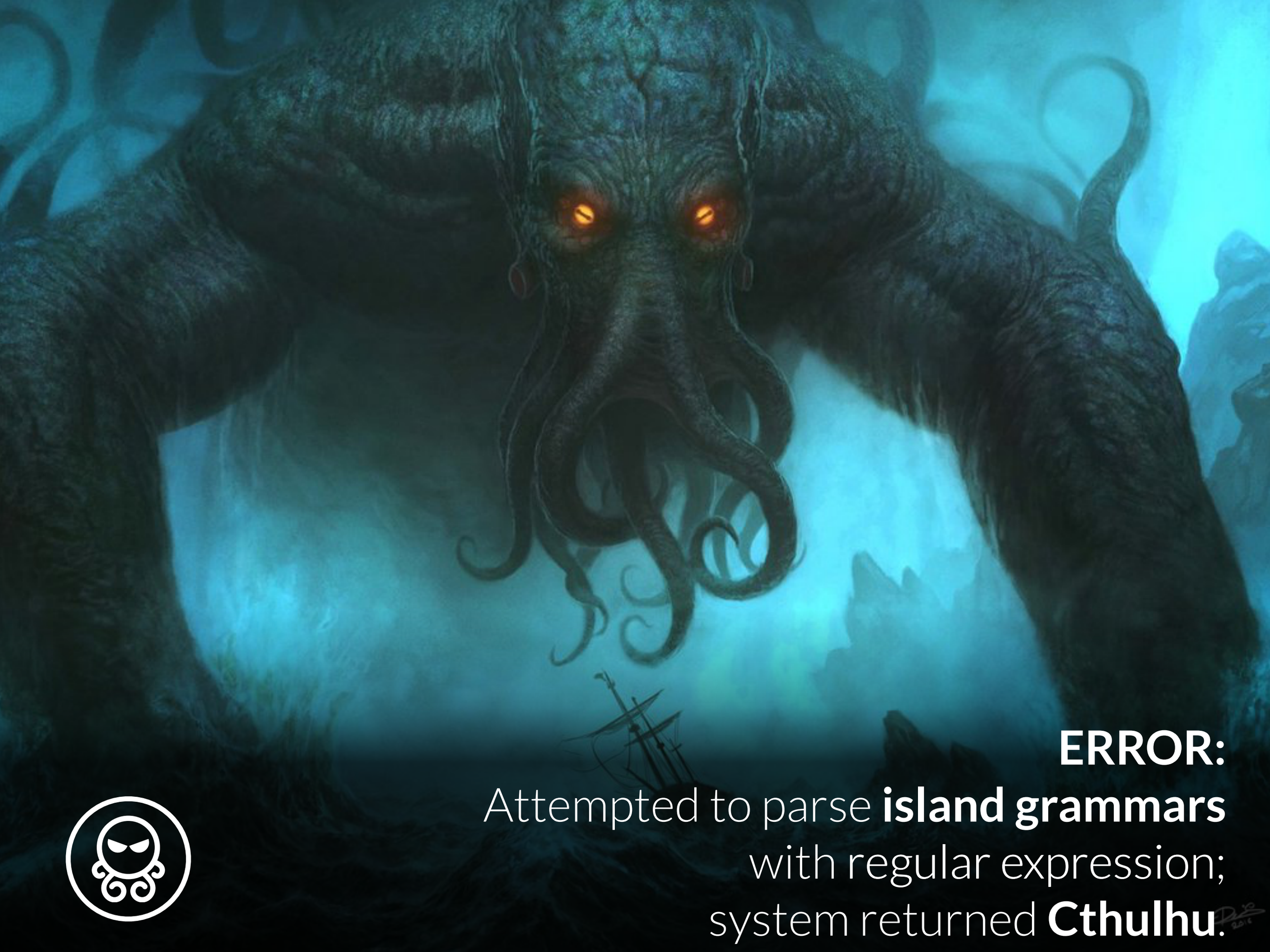
```
ReferenceType:
  ClassOrInterfaceType
  TypeVariable
  ArrayType

ClassOrInterfaceType:
  ClassType
  InterfaceType

ClassType:
  Identifier TypeArgumentsopt
  ClassOrInterfaceType.Identifier TypeArgumentsopt

InterfaceType:
  ClassType

TypeVariable:
  Identifier

ArrayType:
  PrimitiveType Dims
  ClassOrInterfaceType Dims
  TypeVariable Dims

Dims:
  ([ ])+
```

```
TypeParameter:
  TypeParameterModifier* Identifier TypeBoundopt

TypeParameterModifier:
  Annotation

TypeBound:
  extends TypeVariable
  extends ClassOrInterfaceType AdditionalBound*

AdditionalBound:
  & InterfaceType
```

Island Grammars
Recursion

```
ReferenceType:
  ClassType
  InterfaceType
  TypeVariable
  ArrayType

ClassType:
  Identifier TypeArguments_opt
  ClassOrInterfaceType.Identifier TypeArguments_opt

TypeArguments:
  < TypeArgumentList >

TypeArgumentList:
  TypeArgument (, TypeArgument )*

TypeArgument:
  ReferenceType
```

Island Grammars
Recursion



ERROR:
Attempted to parse **island grammars**
with regular expression;
system returned **Cthulhu**.

Extracting Structured Data from Natural Language Documents with Island Parsing

Alberto Bacchelli
Faculty of Informatics
Univ. of Lugano, Switzerland

Anthony Cleve
Faculty of Informatics
University of Namur, Belgium

Michele Lanza
Faculty of Informatics
Univ. of Lugano, Switzerland

Andrea Mocci
DEI
Politecnico di Milano, Italy

***Abstract*—The design and evolution of a software system leave traces in various kinds of artifacts. In software, produced by humans for humans, many artifacts are written in natural language by people involved in the project. Such entities contain structured information which constitute a valuable source of knowledge for analyzing and comprehending a system's design and evolution. However, the ambiguous and informal nature of narrative is a serious challenge in gathering such information, which is scattered throughout natural language text.**

We present an approach–based on island parsing–to recognize and enable the parsing of structured information that occur in natural language artifacts. We evaluate our approach by applying it to mailing lists pertaining to three software systems. We show that this approach allows us to extract structured data from emails with high precision and recall.

I. Introduction

Every system has a history: A history of human decisions during the design process, of changes during the development, and of successes and failures during the whole evolution.

Mining software repositories explores how a system's history can help to (1) support program comprehension, (2) predict future evolutions, and (3) plan various aspects of software projects. Most researchers investigated *structured*

Island Grammars

Being able to extract and parse the structured data enclosed in NL artifacts is the first step for recovering and exploring new facets of a system. For example, the fragments included in emails can be compared with the history of the implementation, to see how and when they start to diverge.

We propose an approach to *extract* the structured information to be found in documents written in NL that concern software development, to allow subsequent data parsing and modeling. The main contributions of this paper are: (1) A novel, robust and flexible technique that uses *island parsing* [3] for mining and extracting structured information of Java systems from textual artifacts (*e.g.*, stack traces, source code, file names); (2) An evaluation of the proposed technique for the extraction of the structured information, based on a significant sample of real-world complex textual documents (*i.e.*, development emails).

II. Source Fragment Extraction

Our purpose is to extract fragments of code and structured data within NL artifacts. Since these fragments are scattered within other information expressed in natural language, we need a robust parsing technique to deal with such information

StORMeD
Island Parser

http://stormed.inf.usi.ch

Parsing Expression Grammars: A Recognition-Based Syntactic Foundation

Bryan Ford
Massachusetts Institute of Technology
Cambridge, MA
baford@mit.edu

Abstract

For decades we have been using Chomsky's generative system of grammars, particularly context-free grammars (CFGs) and regular expressions (REs), to express the syntax of programming languages and protocols. The power of generative grammars to express ambiguity is crucial to their original purpose of modelling natural languages, but this very power makes it unnecessarily difficult both to express and to parse machine-oriented languages using CFGs. Parsing Expression Grammars (PEGs) provide an alternative, recognition-based formal foundation for describing machine-oriented syntax, which solves the ambiguity problem by not introducing ambiguity in the first place. Where CFGs express nondeterministic choice between alternatives, PEGs instead use *prioritized choice*. PEGs address frequently felt expressiveness limitations of CFGs and REs, simplifying syntax definitions and making it unnecessary to separate their lexical and hierarchical components. A linear-time parser can be built for any PEG, avoiding both the complexity and fragility of LR parsers and the inefficiency of generalized CFG parsing. While PEGs provide a rich set of operators for constructing grammars, they are reducible to two minimal recognition schemas developed around 1970, formalized as TS/TDPL and gTS/GTDPL, which are here proven equivalent in effective recognition power.

Island Parsing with PEGs

1 Introduction

Most language syntax theory and practice is based on *generative* systems, such as regular expressions and context-free grammars, in which a language is defined formally by a set of rules applied recursively to generate strings of the language. A *recognition-based* system, in contrast, defines a language in terms of rules or predicates that decide whether or not a given string is in the language. Simple languages can be expressed easily in either paradigm. For example, $\{s \in \mathtt{a}^* \mid s = (\mathtt{aa})^n\}$ is a generative definition of a trivial language over a unary character set, whose strings are "constructed" by concatenating pairs of a's. In contrast, $\{s \in \mathtt{a}^* \mid (|s| \ mod \ 2 = 0)\}$ is a recognition-based definition of the same language, in which a string of a's is "accepted" if its length is even.

While most language theory adopts the generative paradigm, most practical language applications in computer science involve the recognition and structural decomposition, or *parsing*, of strings. Bridging the gap from generative definitions to practical recognizers is the purpose of our ever-expanding library of parsing algorithms with diverse capabilities and trade-offs [9].

Chomsky's generative system of grammars, from which the ubiquitous context-free grammars (CFGs) and regular expressions (REs)

Input: "int"

IntFirst:✓
"int"✓| "integer"

IntegerFirst:✓
"integer"✗| "int"✓

Island Grammars with PEGs

Input: "int"

IntFirst:✓
"int"✓| "integer"

IntegerFirst:✓
"integer"✗| "int"✓

Input: "integer"

IntFirst:✗
"int"✓| "integer"

IntegerFirst:✓
"integer"✓| "int"

Priority Matters in PEG!

Island Grammars
with PEGs

Island Grammar
for Java 8

```
@Override
private void aMethod(
        Object param
        List<T> list,
        int[] array){
    ...
}
```

Fragments
and Sub-constructs

```
@CustomAnnotation(p1 = “foo”, p2 = “bar”)
private <T> void aMethod(
        final Function<T,Boolean> lambda,
        List<@NonNull ? super T> list,
        @NonNull T... params){
    …
    class LocalClass {
        …
    }
    …
}
```

Fragments
and Sub-constructs

```
Type:
    PrimitiveType
    ReferenceType

PrimitiveType:
    {Annotation} NumericType
    {Annotation} boolean

NumericType:
    IntegralType
    FloatingPointType

IntegralType:
    (one of)
    byte short int long char

FloatingPointType:
    (one of)
    float double

ReferenceType:
    ClassOrInterfaceType
    TypeVariable
    ArrayType
```

```
ClassOrInterfaceType:
    ClassType
    InterfaceType

ClassType:
    {Annotation} Identifier [TypeArguments]
    ClassOrInterfaceType . {Annotation} Identifier
[TypeArguments]

InterfaceType:
    ClassType

TypeVariable:
    {Annotation} Identifier

ArrayType:
    PrimitiveType Dims
    ClassOrInterfaceType Dims
    TypeVariable Dims

Dims:
    {Annotation} [ ] {{Annotation} [ ]}
```

```
UnannType:
    UnannPrimitiveType
    UnannReferenceType

UnannPrimitiveType:
    NumericType
    boolean

UnannReferenceType:
    UnannClassOrInterfaceType
    UnannTypeVariable
    UnannArrayType

UnannClassOrInterfaceType:
    UnannClassType
    UnannInterfaceType

UnannClassType:
    Identifier [TypeArguments]
    UnannClassOrInterfaceType . {Annotation}
Identifier [TypeArguments]

UnannInterfaceType:
    UnannClassType

UnannTypeVariable:
    Identifier

UnannArrayType:
    UnannPrimitiveType Dims
    UnannClassOrInterfaceType Dims
    UnannTypeVariable Dims
```

```
TypeParameter:
    {TypeParameterModifier} Identifier [TypeBound]

TypeParameterModifier:
    Annotation
TypeBound:
    extends TypeVariable
    extends ClassOrInterfaceType {AdditionalBound}

AdditionalBound:
    & InterfaceType
    TypeArguments:
    < TypeArgumentList >

TypeArgumentList:
    TypeArgument {, TypeArgument}

TypeArgument:
    ReferenceType
    Wildcard

Wildcard:
    {Annotation} ? [WildcardBounds]

WildcardBounds:
    extends ReferenceType
    super ReferenceType
```

```
MethodDeclaration:
    {MethodModifier} MethodHeader MethodBody

MethodModifier:
    (one of)
    Annotation public protected private
    abstract static final synchronized native
    strictfp

MethodHeader:
    Result MethodDeclarator [Throws]
    TypeParameters {Annotation} Result
    MethodDeclarator [Throws]

Result:
    UnannType
    void

MethodDeclarator:
    Identifier ( [FormalParameterList] ) [Dims]

FormalParameterList:
    ReceiverParameter
    FormalParameters , LastFormalParameter
    LastFormalParameter

FormalParameters:
    FormalParameter {, FormalParameter}
    ReceiverParameter {, FormalParameter}

FormalParameter:
    {VariableModifier} UnannType
    VariableDeclaratorId

VariableModifier:
    (one of)
    Annotation final

LastFormalParameter:
    {VariableModifier} UnannType {Annotation} ...
VariableDeclaratorId
    FormalParameter

ReceiverParameter:
    {Annotation} UnannType [Identifier .] this

Throws:
    throws ExceptionTypeList

ExceptionTypeList:
    ExceptionType {, ExceptionType}

ExceptionType:
    ClassType
    TypeVariable

MethodBody:
    Block
    ;
```

```
MethodDecl:
  Modifier* Type Identifier Args Body

ConstructorDecl:
  Modifier* Identifier Args Body
```

Fragments
and Sub-constructs

```
MethodDecl:
  Modifier* Type Identifier Args Body

ConstructorDecl:
  Modifier* Identifier Args Body
```

```
Island:
  ConstructorDecl | MethodDecl
```

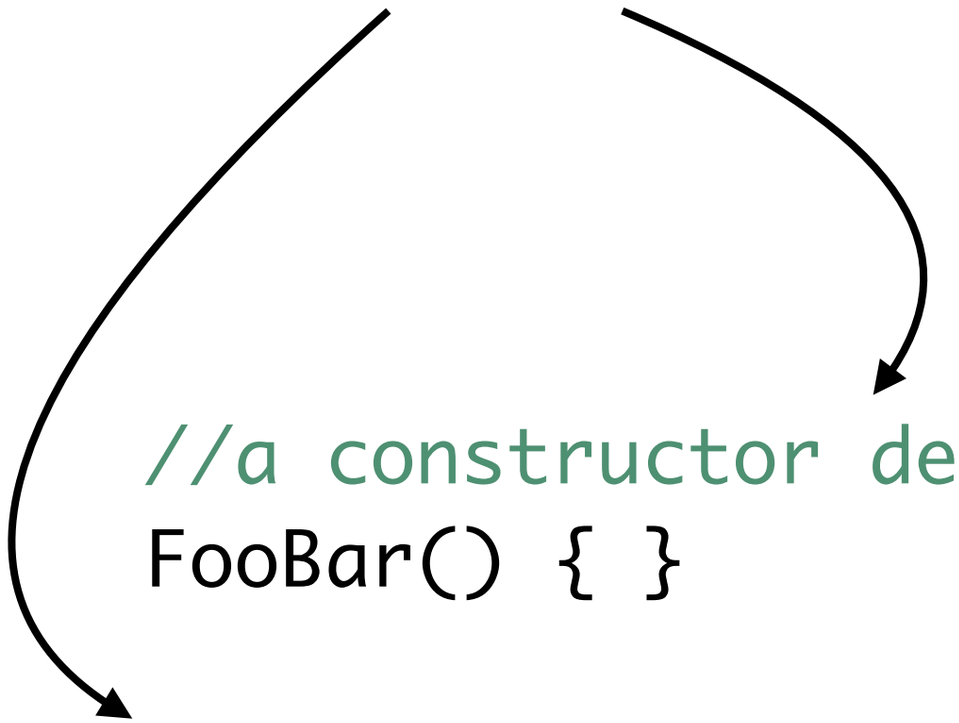
Fragments
and Sub-constructs

```
Island:
  ConstructorDecl | MethodDecl
```

```
//a constructor declarator
public FooBar() { }

//a method declarator
private void aMethod() { }
```

Fragments
and Sub-constructs

```
Island:
  ConstructorDecl | MethodDecl
```



```
  //a constructor declarator
  FooBar() { }

  //a method declarator
  void aMethod() { }
```

Fragments
and Sub-constructs

Priority Matters in PEG!

```
Island:
  MethodDecl | ConstructorDecl
```

≠

```
Island:
  ConstructorDecl | MethodDecl
```

Fragments
and Sub-constructs

```
Island:
  ConstructorDecl | MethodDecl
```

```
ConstructorDecl(FooBar() { })
Water(void)
ConstructorDecl(aMethod() { })
```

Fragments
and Sub-constructs

```
Island:
  MethodDecl | ConstructorDecl
```

```
ConstructorDecl(FooBar() { })
MethodDecl(void aMethod() { })
```

Fragments
and Sub-constructs

```
MethodDecl:
  Modifier* Type Identifier Args Body

ConstructorDecl:
  Modifier* Identifier Args Body
```

```
Island:
  ConstructorDecl | MethodDecl
```

Fragments
and Sub-constructs

```
MethodDecl:
  Modifier* Type Identifier Args Body

ConstructorDecl:
  Modifier* Identifier Args Body
```

```
Island:
  MethodDecl | ConstructorDecl
```

Fragments
and Sub-constructs

Is this enough to deal with **natural language**?

Fragments
and Sub-constructs

Consider the constructor `public FooBar() { }` and the method `private void aMethod() { }` immersed in narrative

Fragments
and Natural Language

Consider the constructor `public FooBar() { }` and the method `private void aMethod() { }` immersed in narrative

`Water`(Consider the constructor)

Fragments
and Natural Language

Consider the constructor `public FooBar() { }` and the method `private void aMethod() { }` immersed in narrative

```
Water(Consider the constructor)
ConstructorDecl(public FooBar() { })
```

Fragments
and Natural Language

Consider the constructor `public FooBar() { }` and the method `private void aMethod() { }` immersed in narrative

`Water(`Consider the constructor`)`

`ConstructorDecl(public FooBar() { })`

`Water(`and the method`)`

Fragments
and Natural Language

Consider the constructor `public FooBar() { }` and the method `private void aMethod() { }` immersed in narrative

```
Water(Consider the constructor)
ConstructorDecl(public FooBar() { })
Water(and the method)
MethodDecl(private void aMethod(){ })
```

Fragments
and Natural Language

Consider the constructor `public FooBar() { }` and the method `private void aMethod() { }` immersed in narrative

```
Water(Consider the constructor)
ConstructorDecl(public FooBar() { })
Water(and the method)
MethodDecl(private void aMethod(){ })
Water(immersed in narrative)
```

Fragments
and Natural Language

Consider the constructor `public FooBar() { }` and the method `private void aMethod() { }` immersed in narrative

```
Water(Consider the constructor)
ConstructorDecl(public FooBar() { })
Water(and the method)
MethodDecl(private void aMethod() { })
Water(immersed in narrative)
```

Fragments
and Natural Language

Consider the constructor `FooBar() { }` and the method `void aMethod() { }` immersed in narrative

Fragments
and Natural Language

Consider the constructor `FooBar() { }` and the method `void aMethod() { }` immersed in narrative

```
Water(Consider the)
MethodDecl(constructor FooBar() { })
Water(and the method)
MethodDecl(void aMethod() { })
Water(immersed in narrative)
```

Fragments
and Natural Language

Consider the constructor `FooBar() { }` and the method `void aMethod() { }` immersed in narrative

```
Water(Consider the)
MethodDecl(constructor FooBar() { })
Water(and the method)
MethodDecl(void aMethod() { })
Water(immersed in narrative)
```

Fragments
and Natural Language

`MethodDecl`(constructor `FooBar() { }`)

```
MethodDecl:
  Modifier* Type Identifier Args Body
```

Fragments
and Natural Language

`MethodDecl`(constructor `FooBar() { }`)

MethodDecl:	
`Modifier*`	-
`Type`	constructor
`Identifier`	`FooBar`
`Args`	`()`
`Body`	`{ }`

Fragments
and Natural Language

`MethodDecl(constructor FooBar() { })`

```
MethodDecl:
    Modifier*       -
    Type            constructor
    Identifier      FooBar
    Args            ()
    Body            { }
```

Fragments
and Natural Language

Java **syntax** does not handle
ambiguity with natural language

Fragments
and Natural Language

Java **syntax** does not handle
ambiguity with natural language

```
//valid constructor
fooBar() { }

//valid method
void Bar() { }
```

Fragments
and Natural Language

How can we **improve disambiguation**?

Fragments
and Natural Language

Naming conventions to the rescue!

METHOD NAMES

Begin with lower letter (*e.g.*, aMethod)

Might implement camel case

CLASS NAMES

Begin with capital letter (*e.g.*, SomeType)

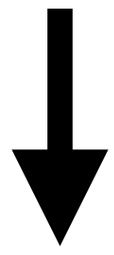
Might implement camel case

Fragments
and Natural Language

```
ConstructorDecl:
  Modifier* Identifier Args Body
```

Fragments
and Natural Language

```
ConstructorDecl:
  Modifier* Identifier Args Body
```



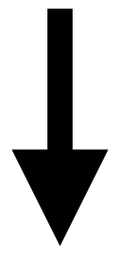
```
ConstructorDecl:
  Modifier+ Identifier Args Body |
  ClassIdentifier Args Body
```

Fragments
and Natural Language

```
MethodDecl:
  Modifier* Type Identifier Args Body
```

Fragments
and Natural Language

```
MethodDecl:
  Modifier* Type Identifier Args Body
```



```
MethodDecl:
  Modifier+ Type Identifier Args Body |
  StrictType MethodIdentifier Args Body
```

Fragments
and Natural Language

```
MethodDecl:
  Modifier+ Type Identifier Args Body |
  StrictType MethodIdentifier Args Body

ConstructorDecl:
  Modifier+ Identifier Args Body |
  ClassIdentifier Args Body
```

Fragments
and Natural Language

Consider the constructor `FooBar() { }` and the method `void aMethod() { }` immersed in narrative

```
Water(Consider the)
MethodDecl(constructor FooBar() { })
Water(and the method)
MethodDecl(void aMethod() { })
Water(immersed in narrative)
```

Fragments
and Natural Language

Consider the constructor `FooBar() { }` and the method `void aMethod() { }` immersed in narrative

```
Water(Consider the constructor)
ConstructorDecl(FooBar() { })
Water(and the method)
MethodDecl(void aMethod(){ })
Water(immersed in narrative)
```

Fragments
and Natural Language

Consider the constructor `FooBar()` and the method `void aMethod()` immersed in narrative

```
Water(Consider the constructor)
XXX(FooBar())
Water(and the method)
MethodDecl(void aMethod())
Water(immersed in narrative)
```

Fragments
and Natural Language

XXX(FooBar())

```
//valid constructor declarator
public FooBar() {
  …
}
```

```
//valid method invocation
new FooBar();
```

Fragments
and Natural Language

```
//valid constructor
declarator
public FooBar() {
  …
}
```

```
//valid method
invocation
new FooBar();
```

`FooBar()` is an **incomplete production**

Fragments
and Natural Language

An Incomplete Production:

Matches a **relaxed** grammar rule

Could lead to **hardly solvable** ambiguity

Incomplete
Productions

Why are incomplete productions needed?

Incomplete
Productions

stackoverflow

Problem with extending JPanel

0

I have an abstract entity:

```
public abstract class Entity extends JPanel implements FocusListener
```

And I have a TextEntity:

```
public class TextEntity extends Entity
```

Inside TextEntity's constructor I want to put a JTextArea that will cover the panel:

```
textArea = new JTextArea();
textArea.setSize(getWidth(),getHeight());
add(textArea);
```

But `getWidth()` and `getHeight()` returns 0. Is it a problem with the inheritance or the constructor?

java inheritance

share edit

asked Apr 7 '10 at 12:40

Halo

756 ● 12 ● 32

```
public abstract class SomeClass
  extends SuperClass implements SomeInterface
```

```
public class SomeClass extends SuperClass
```

```
class SomeClass extends SuperClass
```

```
class SomeClass
```

Incomplete
Productions

```
IncompleteClassDecl:
  …
  Modifier* class Identifier extends Type  |
  Modifier* class Identifier implements Type  |
  Modifier* class Identifier
```

Incomplete
Productions

The higher the **relaxation**
The higher the **ambiguity**

Incomplete
Productions

The higher the **relaxation**
The higher the **ambiguity**

"I missed a class yesterday"

Matches IncompleteClassDecl!

Incomplete
Productions

```
IncompleteClassDecl:
  …
  Modifier* class Identifier extends Type  |
  Modifier* class Identifier implements Type  |
  Modifier* class Identifier
```

Incomplete
Productions

```
IncompleteClassDecl:
  …
  Modifier* class Identifier extends Type  |
  Modifier* class Identifier implements Type  |
  Modifier+ class Identifier |
  class ClassIdentifier

ClassIdentifier:
  Identifier respecting naming conventions
```

Incomplete
Productions

Consider the constructor `FooBar()` and the method `void aMethod()` immersed in narrative

```
Water(Consider the constructor)
XXX(FooBar())
Water(and the method)
MethodDecl(void aMethod())
Water(immersed in narrative)
```

```
ConstructorDecl:
  Modifier+ Identifier Args Body   |
  ClassIdentifier Args Body
```

Incomplete
Productions

```
IncompleteConstructorDecl:
  Modifier+ Identifier Args Bodyopt |
  ClassIdentifier Args Bodyopt
```

Incomplete
Productions

```
IncompleteConstructorDecl:
  Modifier+ Identifier Args Bodyopt |
  ClassIdentifier Args Bodyopt

StrictMethodInvocation:
  MethodIdentifier Args |
  …
```

Incomplete
Productions

FooBar() **matches both!**

```
ConstructorDecl(FooBar())
MethodInv(FooBar())
```

Incomplete
Productions

Unsolvable Ambiguity

Incomplete
Productions

```
IncompleteConstructorDecl:
  Modifier+ Identifier Args Bodyopt |
  ClassIdentifier Args Bodyopt
```

```
StrictMethodInvocation:
  MethodIdentifier Args |
  …
```

Incomplete
Productions

Code elements
with a **role** in the narrative

In-Paragraph
Fragments

stackoverflow

Problem with extending JPanel

0

I have an abstract entity:

```
public abstract class Entity extends JPanel implements FocusListener
```

And I have a TextEntity:

```
public class TextEntity extends Entity
```

Inside TextEntity's constructor I want to put a JTextArea that will cover the panel:

```
textArea = new JTextArea();
textArea.setSize(getWidth(),getHeight());
add(textArea);
```

But `getWidth()` and `getHeight()` returns 0. Is it a problem with the inheritance or the constructor?

java inheritance

share edit

asked Apr 7 '10 at 12:40

Halo

756 ● 12 ● 32

Qualified Identifiers

Method and Class Names

Method Invocations

In-Paragraph
Fragments

Qualified Identifiers

Method and Class Names

Method Invocations

In-Paragraph
Fragments

```
QualifiedIdentifier:
  Identifier (. Identifier)+
```

In-Paragraph Fragments

```
QualifiedIdentifier:
  Identifier (. Identifier)+
```

“java.util.Date vs java.sql.Date: when to use which and why?”

```
QualifiedIdentifier(java.util.Date)
Water(vs )
QualifiedIdentifier(java.sql.Date)
Water(: when to use which and why?)
```

In-Paragraph Fragments

```
QualifiedIdentifier:
  Identifier (. Identifier)+
```

“Hi Mario. How are you?”

```
Water(Hi )
QualifiedIdentifier(Mario. How)
Water(are you?)
```

In-Paragraph Fragments

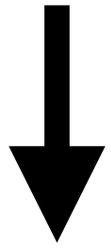
```
QualifiedIdentifier:
  Identifier (. Identifier)+
```

“Hi Mario. How are you?”

Spaces are allowed in Java

In-Paragraph
Fragments

```
QualifiedIdentifier:
  Identifier (. Identifier)+
```



```
QualifiedIdentifier:
  Identifier (. !(Space+) Identifier)+
```

In-Paragraph
Fragments

Qualified Identifiers

Method and Class Names

Method Invocations

In-Paragraph
Fragments

Qualified Identifiers

Method and Class Names

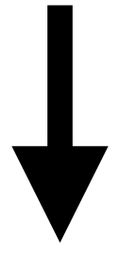
Method Invocations

In-Paragraph
Fragments

```
ClassName:
  JavaLetter (JavaLetterOrDigit)*
```

In-Paragraph Fragments

```
ClassName:
  JavaLetter (JavaLetterOrDigit)*
```



```
ClassName:
  CapitalLetter (JavaLetterOrDigit)+
```

In-Paragraph
Fragments

```
ClassName:
  CapitalLetter (JavaLetterOrDigit)+
```

“Inside TextEntity’s constructor I want to put a JTextArea that will cover the panel”

`ClassName`(Inside)

`ClassName`(TextEntity)

`Water`(’s constructor I want to put a)

`ClassName`(JTextArea)

`Water`(that will cover the panel)

In-Paragraph
Fragments

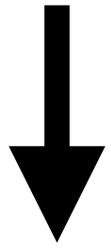
```
ClassName:
  CapitalLetter (JavaLetterOrDigit)+
```

"Inside TextEntity's constructor I want to put a JTextArea that will cover the panel"

Respects the naming convention!

In-Paragraph
Fragments

```
ClassName:
  CapitalLetter (JavaLetterOrDigit)+
```



```
ClassName:
  CapitalLetter IdentifierWithCaseChanges
```

In-Paragraph
Fragments

```
ClassName:
  CapitalLetter IdentifierWithCaseChanges
```

“Inside TextEntity’s constructor I want to put a JTextArea that will cover the panel”

```
Water(Inside )
ClassName(TextEntity)
Water(’s constructor I want to put a )
ClassName(JTextArea)
Water(that will cover the panel)
```

In-Paragraph
Fragments

```
ClassName:
  CapitalLetter IdentifierWithCaseChanges

MethodName:
  LowerCaseLetter IdentifierWithCaseChanges
```

In-Paragraph
Fragments

```
MethodName:
  LowerCaseLetter IdentifierWithCaseChanges
```

`aMethodName`

Java Style

`a_method_name`

C Style

In-Paragraph
Fragments

```
MethodName:
  LowerCaseLetter IdentifierWithCaseChangeOrUnderscore
```

`aMethodName`

Java Style

`a_method_name`

C Style

In-Paragraph
Fragments

Qualified Identifiers

Method and Class Names

Method Invocations

In-Paragraph
Fragments

Qualified Identifiers

Method and Class Names

Method Invocations

In-Paragraph
Fragments

```
object.aMethod<SomeType>(p1,p2,p3)
```

In-Paragraph Fragments

```
MethodInvocation:
    QualifiedIdentifier      object.aMethod
    TypeArgument_opt         <SomeType>
    Args                     (p1,p2,p3)
```

In-Paragraph Fragments

MethodInvocation:

QualifiedIdentifier TypeArgument$_{opt}$ Args

```
object.aMethod<SomeType>()
aMethod<SomeType>()
method<SomeType>()
```

In-Paragraph
Fragments

MethodInvocation:

    QualifiedIdentifier TypeArgument$_{opt}$ Args

```
object.aMethod()
aMethod()
method()
```

In-Paragraph
Fragments

```
MethodInvocation:
    QualifiedIdentifier TypeArgument_opt Args
```

```
object.aMethod()
aMethod()
method()
```

In-Paragraph
Fragments

MethodInvocation:

QualifiedIdentifier TypeArgument$_{opt}$ Args

```
object.aMethod(p1, p2, p3)
aMethod(p1, p2, p3)
method(p1, p2, p3)
```

In-Paragraph
Fragments

MethodInvocation:

QualifiedIdentifier $\text{TypeArgument}_{opt}$ Args

```
object.aMethod(p1, p2, p3)
aMethod(p1, p2, p3)
method(p1, p2, p3)
```

MethodInvocation:

    QualifiedIdentifier TypeArgument$_{opt}$ Args

```
object.aMethod(p1)
aMethod(p1)
method(p1)
```

In-Paragraph
Fragments

```
MethodInvocation:
    QualifiedIdentifier TypeArgument_opt Args
```

```
object.aMethod(p1)
aMethod(p1)
method(p1)
```

In-Paragraph
Fragments

MethodInvocation:

    QualifiedIdentifier TypeArgument$_{opt}$ Args

```
object.aMethod(p1)
aMethod(p1)
method(p1)
```

In-Paragraph
Fragments

method invocations with one argument introduce **false positives**

In-Paragraph
Fragments

“the color of the apples (red) is not yellow”

```
Water(the color of the )
MethodInvocation(apples (red))
Water(is not yellow)
```

In-Paragraph
Fragments

Should we ignore all method invocations with **one** argument?

In-Paragraph
Fragments

MethodInvocation:

QualifiedIdentifier TypeArgument$_{opt}$ Args

```
object.aMethod(p1)
aMethod(p1)
method(p1)
method(obj.value)
method(“text”)
method(obj.inv())
method(2 * var)
```

In-Paragraph
Fragments

Valid constructs **containing** invalid
or incomplete constructs

Island
With Lakes

IncompleteClassDeclaration

```
@Entity
@Table(name = “shops”)
public class Shop {
  ...
 @ManyToOne(fetch=FetchType.EAGER)
 @Table(name = “shop_type_id”)
 private TypeShop typeShop;
  ...
}
```

WaterMethod

FieldDeclaration

Island
with Lakes

```
ClassDeclaration:
  Modifier* class Identifier (extends Type)opt
    (implements InterfaceType)opt Body
```

Island
with Lakes

```
ClassDeclaration:
  Modifier* class Identifier (extends Type)opt
    (implements InterfaceType)opt BodyWithIslands
```

Island
with Lakes

```
ClassDeclaration:
  Modifier* class Identifier (extends Type)opt
    (implements InterfaceType)opt    BodyWithIslands


MethodDeclaration:
  Modifier+ Type Identifier Args BodyWithIslands |
  StrictType MethodIdentifier Args BodyWithIslands
```

Island
with Lakes

IncompleteClassDeclaration

```
@Entity
@Table(name = “shops”)
public class Shop {
  ...
 @ManyToOne(fetch=FetchType.EAGER)
 @Table(name = “shop_type_id”)
 private TypeShop typeShop;
  ...
}
```

WaterMethod

FieldDeclaration

Island
with Lakes

ClassDeclaration

```
@Entity
@Table(name = “shops”)
public class Shop {
  ...
 @ManyToOne(fetch=FetchType.EAGER)
 @Table(name = “shop_type_id”)
 private TypeShop typeShop;
  ...
}
```

Island
with Lakes

Are lakes limited to the **body**?

Island
with Lakes

stackoverflow

Java generics to enforce return type of abstract method

11

I have the following situation :

```
abstract class X { abstract X someMethod (...) {...} }.
```

Now I want to constrain any implementation of X to have its 'someMethod' method return that particular implementation type, not just X :

```
class X1 extends X { X1 someMethod (...) {...} }.
class X1 extends X { X someMethod (...) {...} }.  //want this to be flagged as an error
class X2 extends X { X1 someMethod (...) {...} }.  //want this to be flagged as an error too
```

Is it possible to achieve this using Java generics ?

EDIT

Okay. I only asked the yes/no question and got a "yes". My fault. What I was actually interested in is "how do I write the declarations".

java

share edit

edited Oct 26 '09 at 22:12

asked Oct 26 '09 at 22:01

Erwin Smout

9,857 ● 1 ● 12 ● 33

```
abstract class SomeClass {
 abstract void someMethod(...) {...}
}
```

Elision

```
abstract class SomeClass {
 abstract void someMethod(...) {...}
}
```

```
MethodDeclaration:
  Modifier+ Type Identifier Args BodyWithIslands |
  StrictType MethodIdentifier Args BodyWithIslands
```

Island
with Lakes

```
MethodDeclaration:
  Modifier+ Type Identifier Args BodyWithIslands |
  StrictType MethodIdentifier Args BodyWithIslands


Args:
( FormalParam* ) |
( Ellipsis )
```

Island
with Lakes

How do we **model** the results of the parser?

Modeling
Results

Heterogenous
Abstract Syntax Tree (**H-AST**)

Modeling
Results

```
case class ClassDeclarationNode (
 val modifiers: Seq[ModifierNode],
 val identifier: IdentifierNode,
 val typeParams: Option[TypeParamsNode],
 val superType: Option[TypeNode],
 val interfaces: Option[TypeListNode],
 val body: ClassBodyNode
)
```

Modeling
Results

What about **incompleteness**?

Modeling
Results

```
case class ClassDeclarationNode (
 val modifiers: Seq[ModifierNode],
 val identifier: IdentifierNode,
 val typeParams: Option[TypeParamsNode],
 val superType: Option[TypeNode],
 val interfaces: Option[TypeListNode],
 val body: ClassBodyNode
)
```

Modeling
Results

```
case class ClassDeclarationNode (
 val modifiers: Seq[ModifierNode],
 val identifier: IdentifierNode,
 val typeParams: Option[TypeParamsNode],
 val superType: Option[TypeNode],
 val interfaces: Option[TypeListNode],
 val body: Option[ClassBodyNode]
)
```

Modeling
Results

…and **islands** with lakes?

Modeling
Results

```
case class ClassDeclarationNode (
 val modifiers: Seq[ModifierNode],
 val identifier: IdentifierNode,
 val typeParams: Option[TypeParamsNode],
 val superType: Option[TypeNode],
 val interfaces: Option[TypeListNode],
 val body: Option[ClassBodyNode]
)

case class ClassBodyNode (
 val members: Seq[MemberDeclarationNode]
)
```

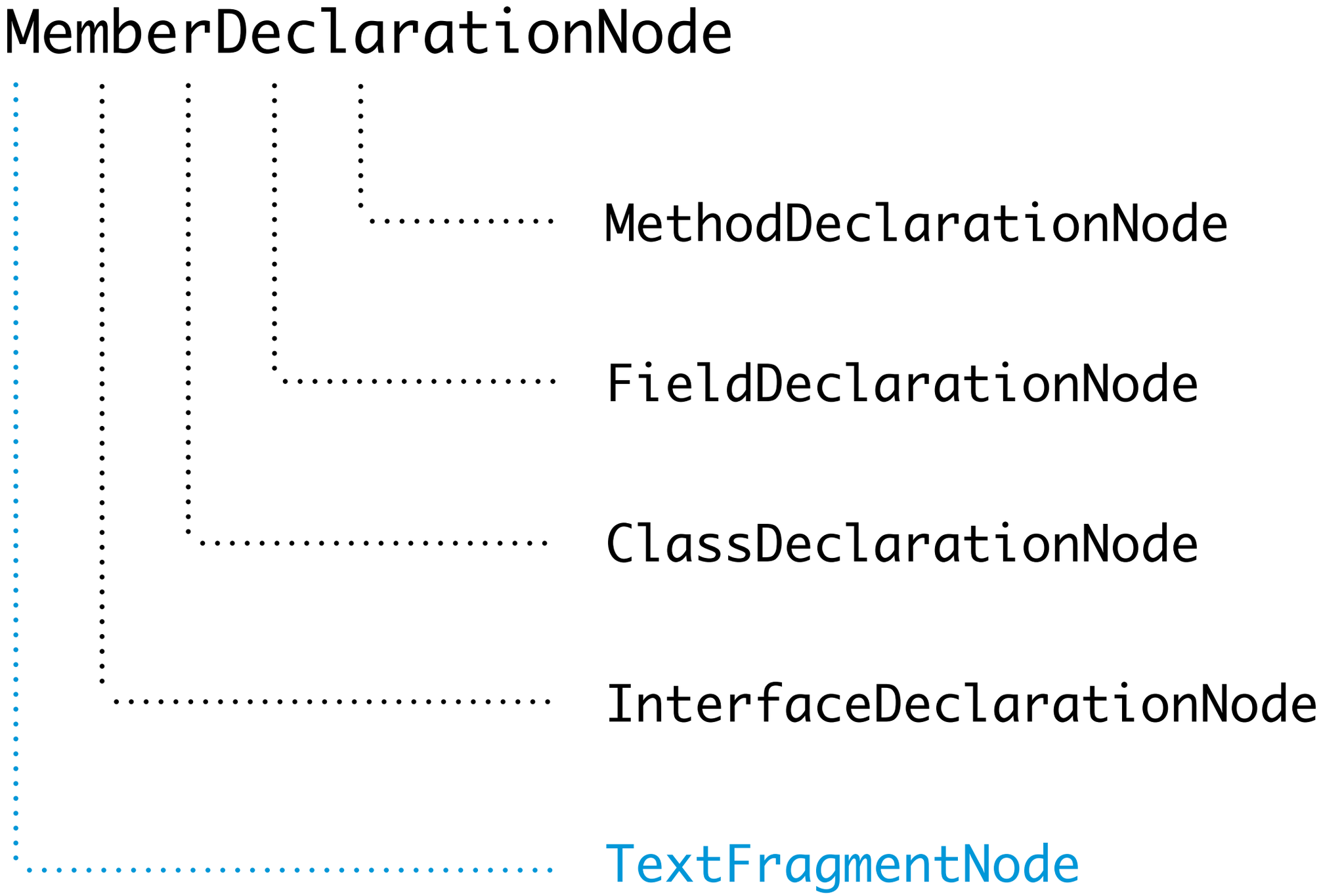
Modeling
Results

```
case class ClassDeclarationNode (
 val modifiers: Seq[ModifierNode],
 val identifier: IdentifierNode,
 val typeParams: Option[TypeParamsNode],
 val superType: Option[TypeNode],
 val interfaces: Option[TypeListNode],
 val body: Option[ClassBodyNode]
)

case class ClassBodyNode (
 val members: Seq[MemberDeclarationNode]
)
```

Modeling
Results

a H-AST allows **extensibility**

Modeling
Results



Modeling
Results

a H-AST allows analyses **beyond text**

Modeling
Results

```
@Entity
@Table(name = “shops”)
public class Shop {
  ...
  I dunno if this is correct...it keeps failing

  @ManyToOne(fetch=FetchType.EAGER)
  @Table(name = “shop_type_id”)
  private TypeShop typeShop;
  ...
}
```

Modeling
Results

```
@Entity
@Table(name = “shops”)
public class Shop {
  ...
  I dunno if this is correct...it keeps failing

  @ManyToOne(fetch=FetchType.EAGER)
  @Table(name = “shop_type_id”)
  private TypeShop typeShop;
  ...
}
```

Exclude Textual Fragments

Modeling
Results

```
@Entity
@Table(name = “shops”)
public class Shop {
  ...
  I dunno if this is correct...it keeps failing

  @ManyToOne(fetch=FetchType.EAGER)
  @Table(name = “shop_type_id”)
  private TypeShop typeShop;
  ...
}
```

Field Declarations Only

Modeling
Results

```
@Entity
@Table(name = “shops”)
public class Shop {
  ...
  I dunno if this is correct...it keeps failing

  @ManyToOne(fetch=FetchType.EAGER)
  @Table(name = “shop_type_id”)
  private TypeShop typeShop;
  ...
}
```

Identifiers Only

Modeling
Results

Can we extend the model to **additional languages**?

Multilingual
Support

Migrating from spring xml configuration to @Configuration (servlet 3.0) with bean references causes BeanNotOfRequiredTypeException

0

1

I am migrating from xml based spring configuration to "class" based configuration using the corresponding @Configuration annotation.

I came across the following problem: I want to create a new bean, which has a reference to another (service) bean. Therefore I autowired this class to set this reference during bean creation. My configuration class looks as follows:

```
@Configuration
@ComponentScan(basePackages = {"com.akme"})
public class ApplicationContext {

    @Resource
    private StorageManagerBean storageManagerBean;

    @Bean(name = "/storageManager")
    public HessianServiceExporter storageManager() {
        HessianServiceExporter hessianServiceExporter = new HessianServiceExporter();
        hessianServiceExporter.setServiceInterface(StorageManager.class);
        hessianServiceExporter.setService(storageManagerBean);
        return hessianServiceExporter;
    }
}
```

But this doesn't work, because the causes a BeanNotOfRequiredTypeException exception during startup.

```
Bean named 'storageManagerBean' must be of type [com.akme.StorageManagerBean], but was actua
```

The StorageManagerBean is annotated with an @Service annotation. And the xml based configuration worked as expected:

```
<bean name="/storageManager" class="org.springframework.remoting.caucho.HessianServiceExport
    <property name="service" ref="storageManagerBean"/>
    <property name="serviceInterface" value="com.akme.StorageManager"/>
</bean>
```

How to parse JSON in Java

379

109

I have the following JSON text that I need to parse to get `pageName`, `pagePic`, `post_id`, etc.

What is the required code?

```
{
   "pageInfo": {
         "pageName": "abc",
         "pagePic": "http://example.com/content.jpg"
    }
    "posts": [
         {
              "post_id": "123456789012_123456789012",
              "actor_id": "1234567890",
              "picOfPersonWhoPosted": "http://example.com/photo.jpg",
              "nameOfPersonWhoPosted": "Jane Doe",
              "message": "Sounds cool. Can't wait to see it!",
              "likesCount": "2",
              "comments": [],
              "timeOfPost": "1234567890"
         }
    ]
}
```

java json parsing

share improve this question

edited Oct 30 '15 at 6:24

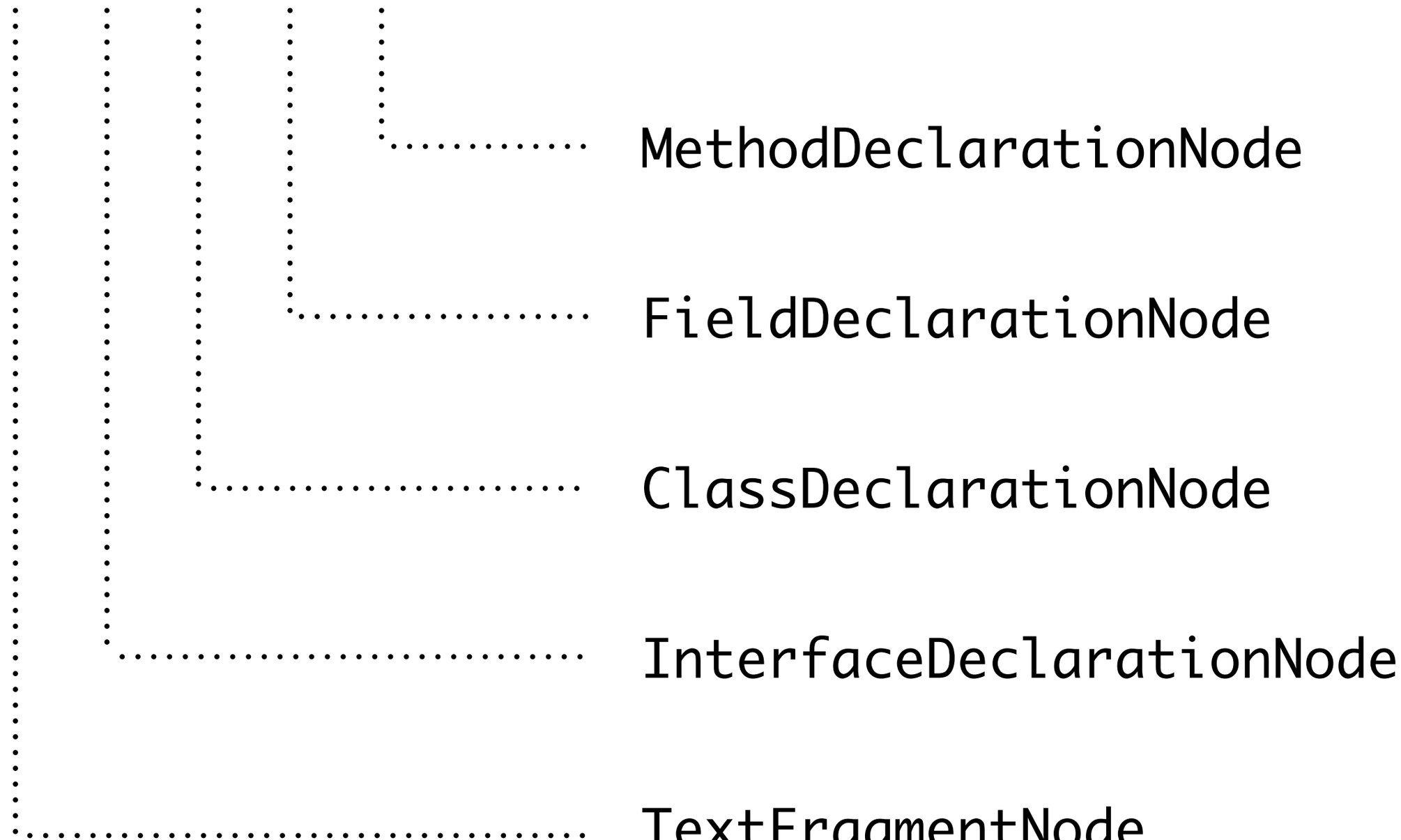
hjpotter92

42.4k ● 11 ● 58 ● 88

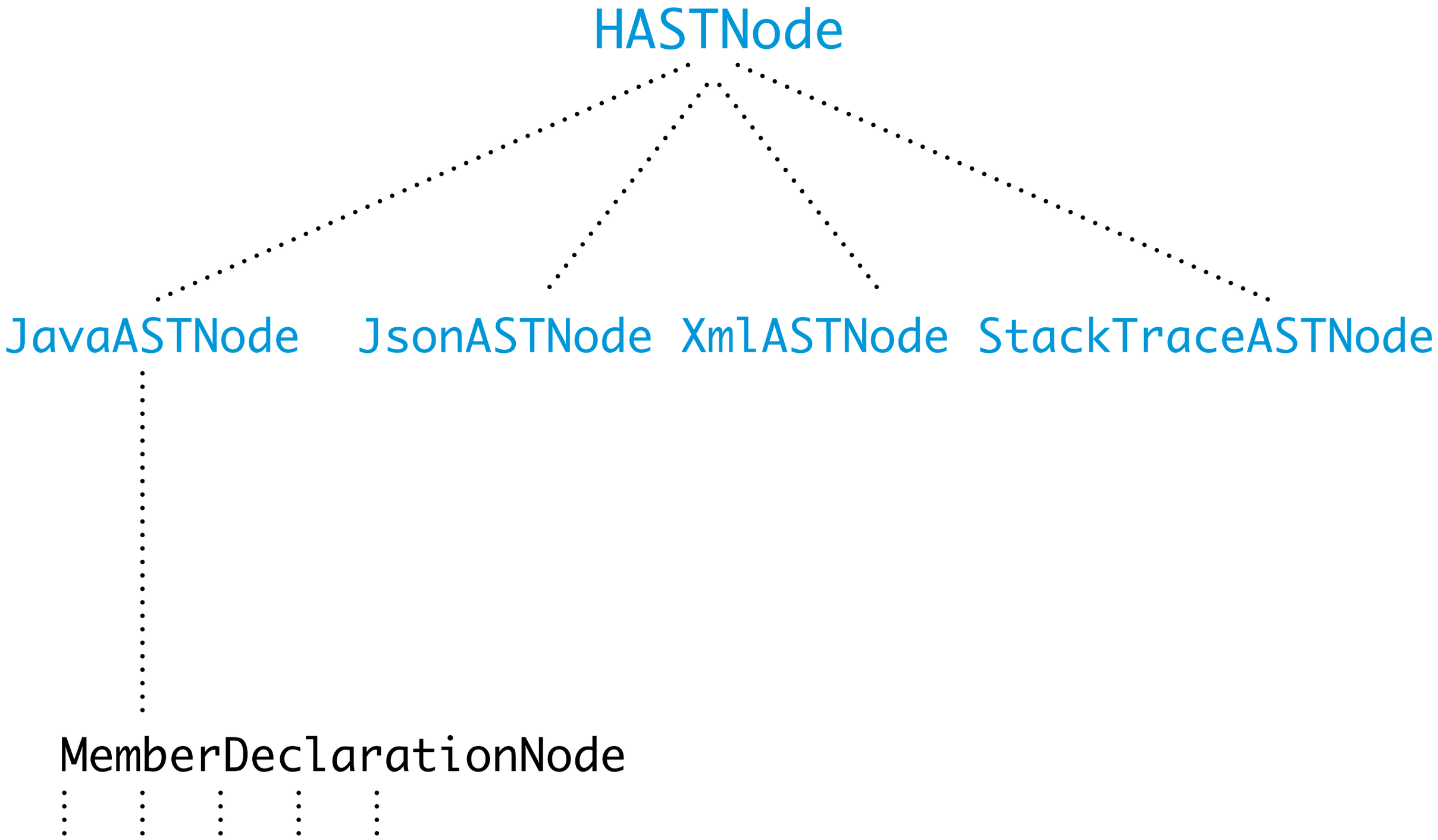
asked Apr 7 '10 at 9:00

Muhammad Maqsoodur Rehman

8,146 ● 29 ● 64 ● 111



Multilingual Support



Multilingual Support

StORMeD
Island Parser

http://stormed.inf.usi.ch

Free island parsing service

1000 queries per day

Scala development kit available

StORMeD
Island Parser

http://stormed.inf.usi.ch

StORMeD

Stack Overflow Ready Made Data

The Island Parsing Service

Do you need the **island parsing** technology behind **StORMeD**?

Now you have the chance to try it out. **For Free**.

Service Registration

To use the service a simple registration with a valid email address is needed. You can register here and obtain a key Every registered user is assigned of a daily quota of **1000** requests. The quota gets **renewed on midnight**. The registration phase is needed to identify users and avoid abuse. The information gathered during the registration are considered confidential and will not be disclosed to third parties.

Only **one key per email** address can be released. Please remember that the key is personal.

Key Recovery

Did you forget or lost your key? No worries, you can recover it. All you need to do is to access the Recovery Page and follow the instructions.

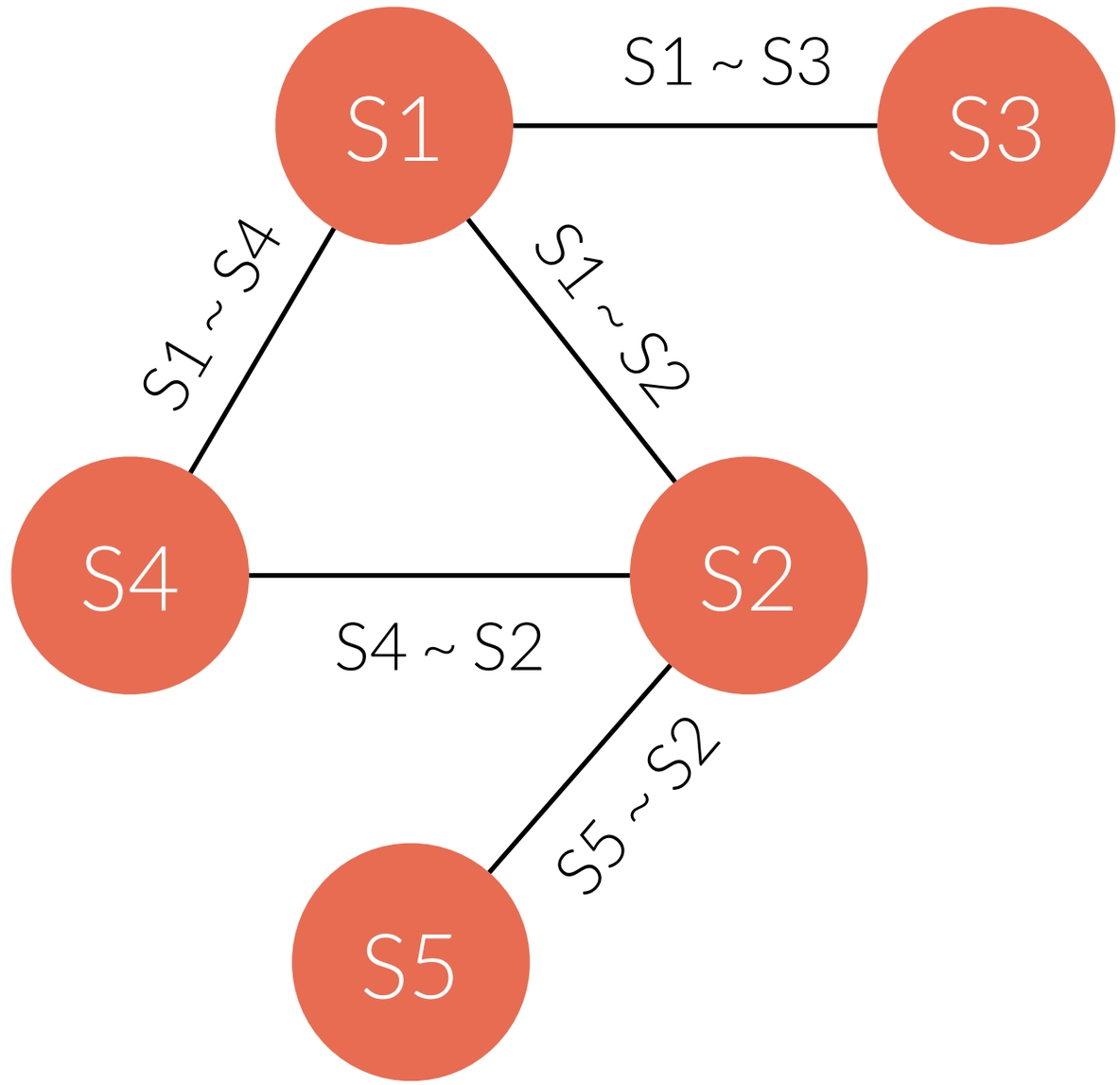
Service Methods

In the current version, the StORMeD API provides one single service *parse*. Every method requires a valid service key and the appropriate JSON with the correct parameters as POST request.

The table below contains all the methods provided by the service, with the input paramters and the service output.

Method	Parameters	Output
/parse	`text` : a string representing the text to be parsed by the service. `key` : a string representing a valid service key for a registered user.	`status` OK or ERROR, always present. if status equals ERROR

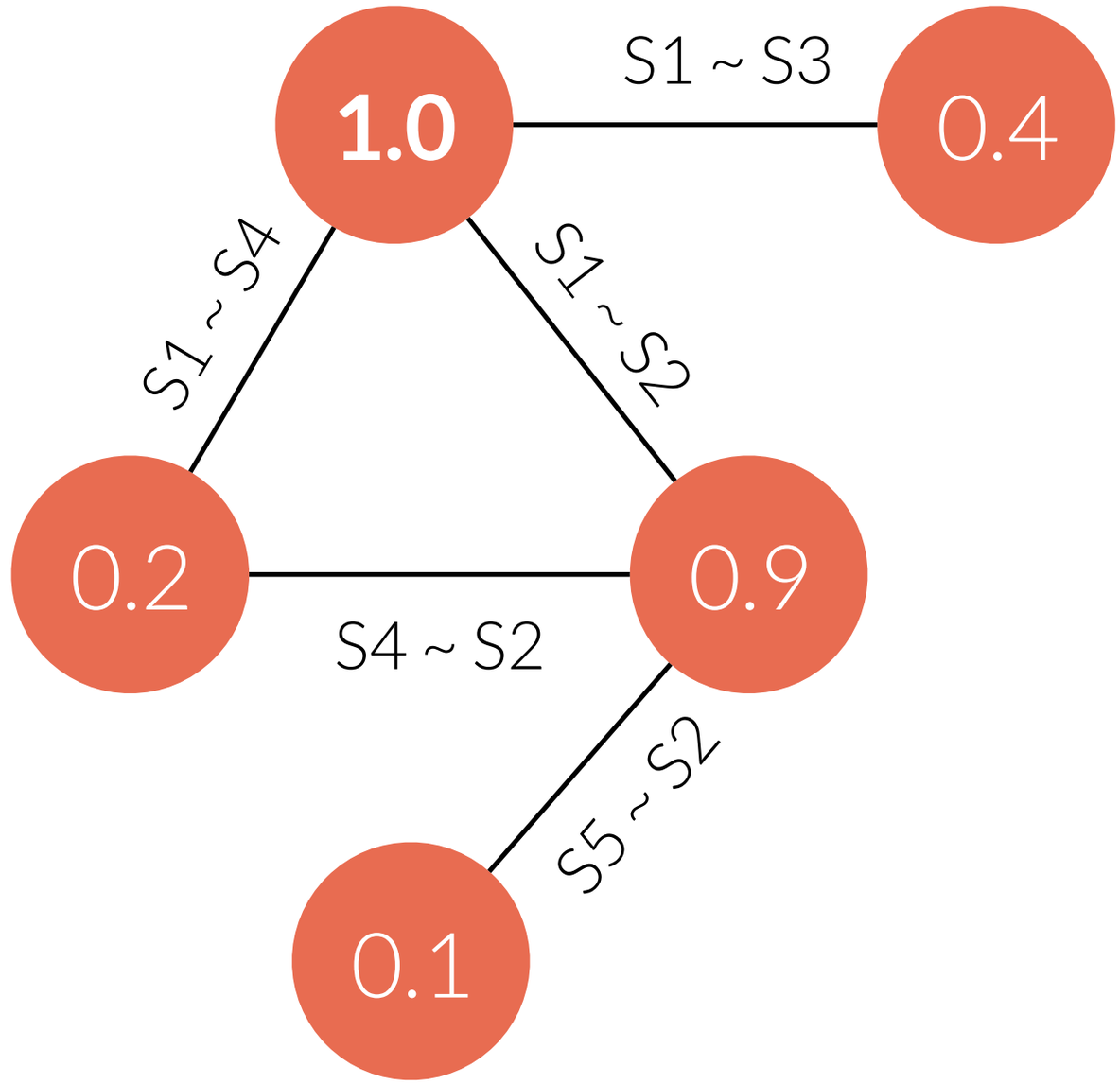
LexRank



Text Summarization

G. Erkan and D. R. Radev
LexRank: Graph-based lexical centrality as salience in text summarization
Journal of Artificial Intelligence Research, vol. 22, 2004

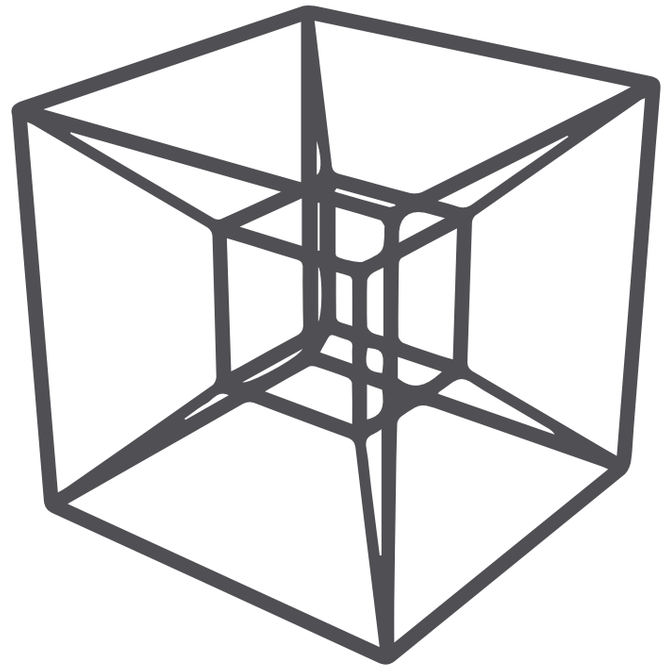
LexRank
Sentence Centrality



Text Summarization

G. Erkan and D. R. Radev
LexRank: Graph-based lexical centrality as salience in text summarization
Journal of Artificial Intelligence Research, vol. 22, 2004

The information is
heterogeneous
and **multidimensional**



Heterogeneous Information

Android Xml Handler keeps only the last objects in my xml file

I have a class named `ExampleHandler extends from DefaultHandler` .

This class reads data from an external xml file (on the web) the structure of this xml file is :

```
<resultsSet>
    <result>
        <title>WinRANI Web Services!</title>
        <nom>DADI</nom>
        <prenom>Morad</prenom>
        <adresse>DANS MES REVES</adresse>
    </result>
...
</resultsSet>
```

I've created a class with the same structure of this xml file called `ParsedExampleDataSet` (it has getters and setters).

I've also created an `ArrayList` which will contain all the 'result' objects but the problem is that when the handler reads all the objects, all items in the `ArrayList` are the same.

Here is my code :

```
package com.example.helloandroid;

import java.util.ArrayList;

import org.xml.sax.Attributes;
import org.xml.sax.SAXException;
import org.xml.sax.helpers.DefaultHandler;

import android.util.Log;


public class ExampleHandler extends DefaultHandler{

    // ===========================================================
    // Fields
    // ===========================================================

    private boolean in_resultset = false;
    private boolean in_result = false;
    private boolean in_title = false;
    private boolean in_nom = false;
```

asked 3 years ago
viewed 399 times
active 3 years ago

Related

0 Android Java Parsing XML via Web

1 Parsing a local XML file in android

0 parsing xml file from network database in android

2 Android: How Can I display all XML values of same tag name

2 Android: How to get values in under specific xml tags

1 How to draw a route by parsing xml to google map on android

0 Android: Handler returns empty string parsing XML with SAX Parser

2 Android SAXParser, parse into array and get child nodes

0 Xml parse android code

0 Reading XML from internet in android

Android Xml Handler keeps only the last objects in my xml file

0

1

I have a class named `ExampleHandler extends from DefaultHandler` .

This class reads data from an external xml file (on the web) the structure of this xml file is :

```
<resultsSet>
    <result>
        <title>WinRANI Web Services!</title>
        <nom>DADI</nom>
        <prenom>Morad</prenom>
        <adresse>DANS MES REVES</adresse>
    </result>
...
</resultsSet>
```

I've created a class with the same structure of this xml file called `ParsedExampleDataSet` (it has getters and setters).

I've also created an `ArrayList` which will contain all the 'result' objects but the problem is that when the handler reads all the objects, all items in the `ArrayList` are the same.

Here is my code :

```
package com.example.helloandroid;

import java.util.ArrayList;

import org.xml.sax.Attributes;
import org.xml.sax.SAXException;
import org.xml.sax.helpers.DefaultHandler;

import android.util.Log;


public class ExampleHandler extends DefaultHandler{

    // ===========================================================
    // Fields
    // ===========================================================

    private boolean in_resultset = false;
    private boolean in_result = false;
    private boolean in_title = false;
    private boolean in_nom = false;
```

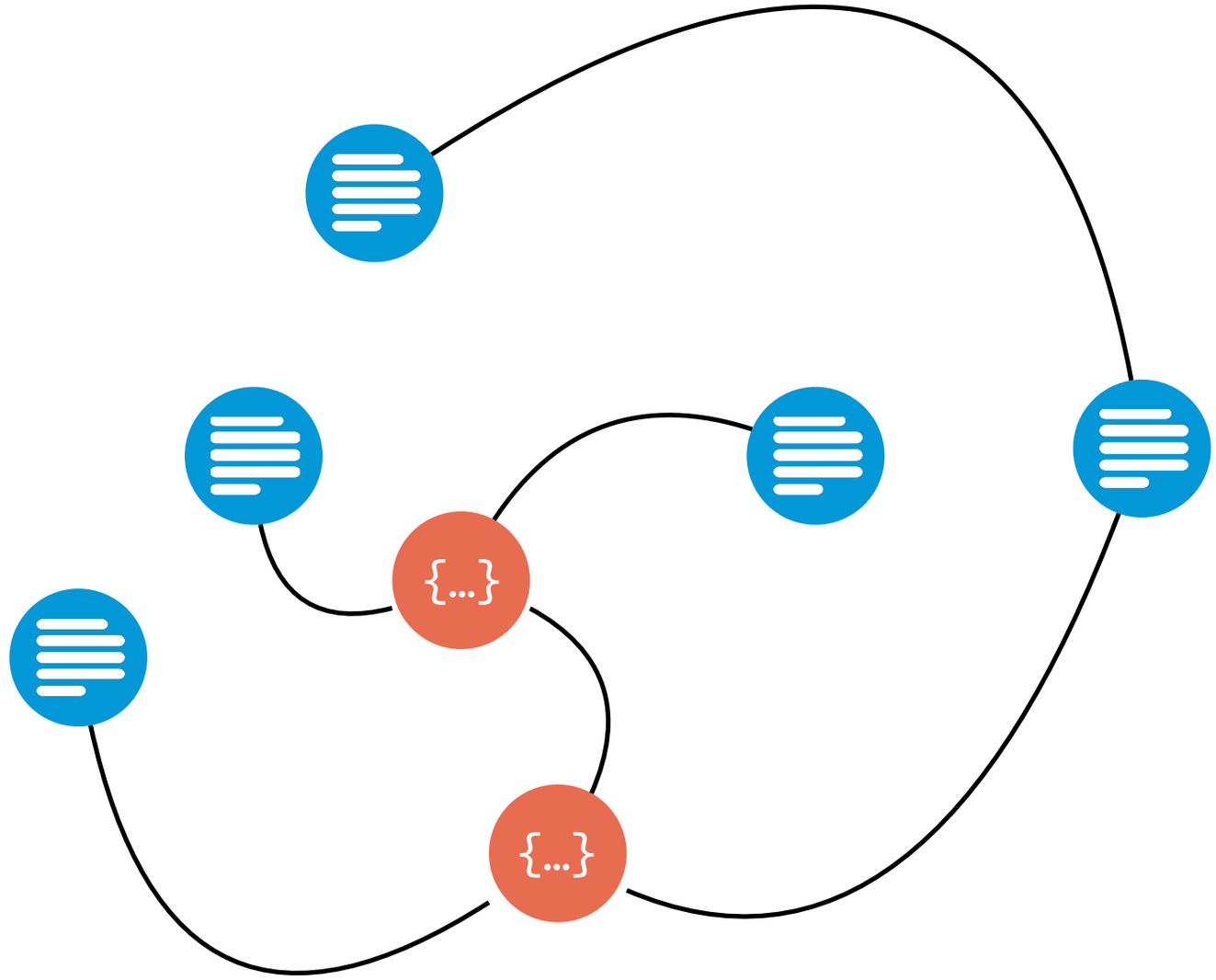
XML

Java

Code Unit

Text Unit

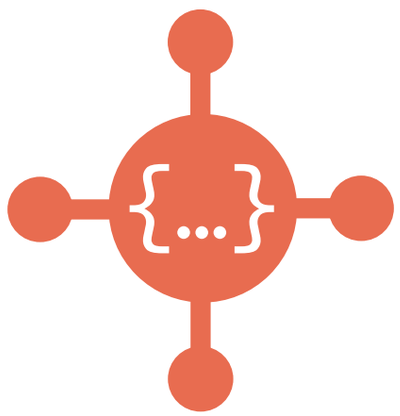
How to **analyze** Information Units?



Information Units

Types

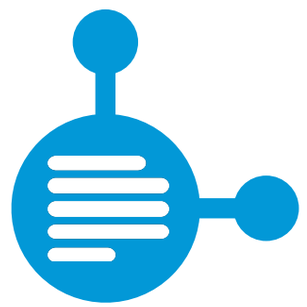
Natural Language



Variable Names

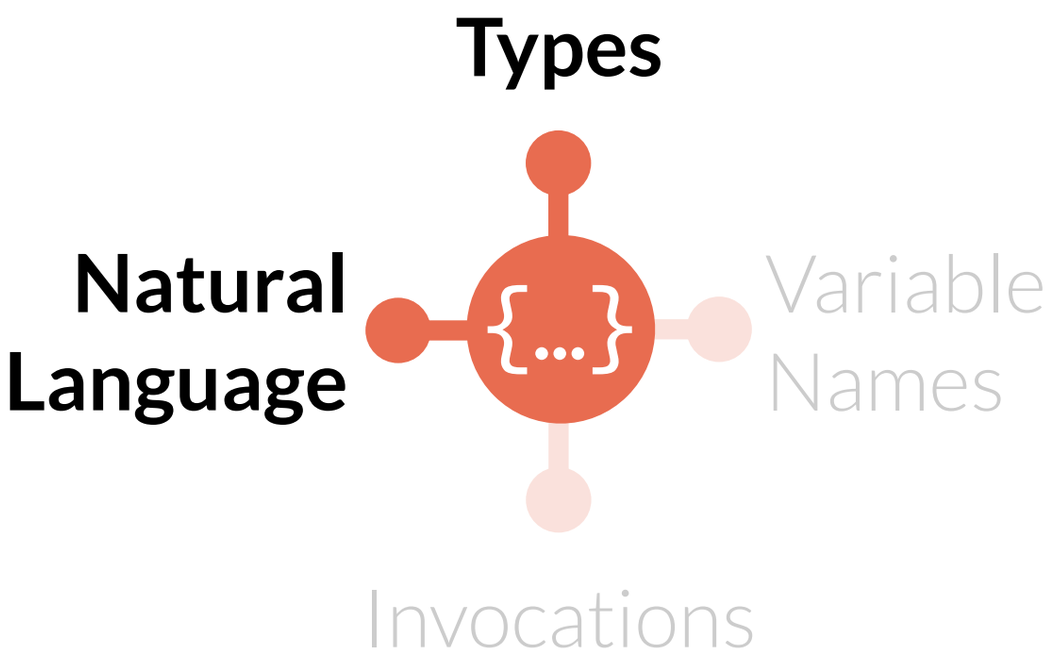
Invocations

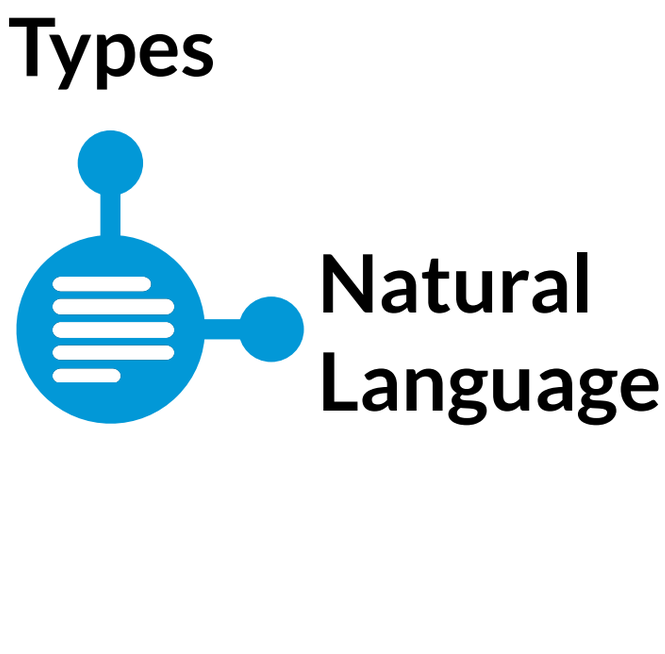
Types



Natural Language

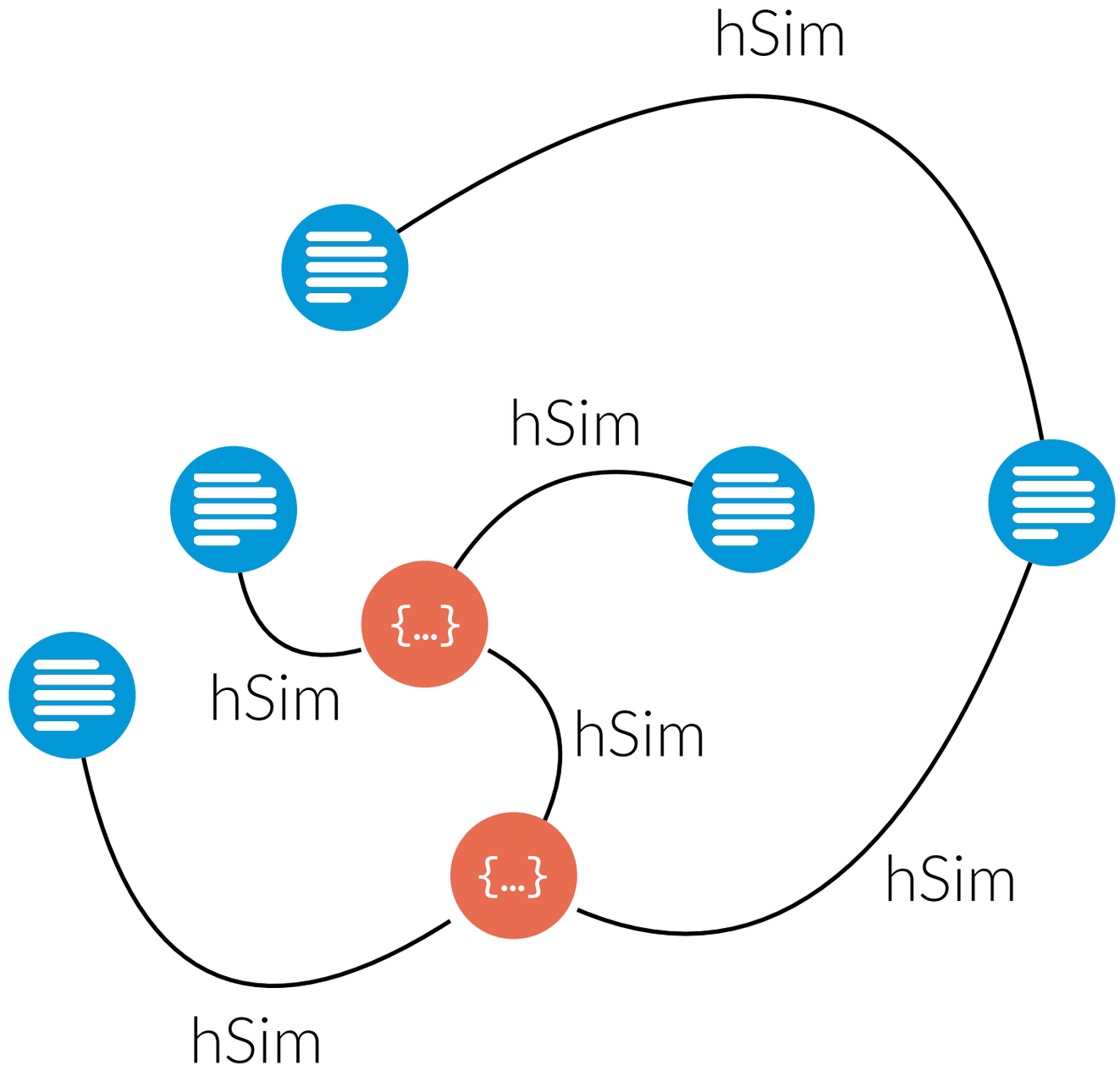
Meta Information





Meta Information

Holistic
Similarity



Information
Summarization

Q&A

L. Moonen
Generating Robust Parsers using Island Grammars
8th IEEE Int. Working Conference on Reverse Engineering (IWCRE 2001)

M.L. Collard, M. Decker, J.I. Maletic
Lightweight Transformation and Fact Extraction with the srcML Toolkit
11th IEEE Int. Working Conference on Source Code Analysis and Manipulation (SCAM 2011)

A. Bacchelli, A. Cleve, M. Lanza, A. Mocci
Extracting structured data from natural language documents with island parsing
26th IEEE/ACM Int. Conference On Automated Software Engineering (ASE 2011)

B. Ford
Parsing expression grammars: A recognition-based syntactic foundation
31st ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages (POPL 2004)

N. Synytskyy, J. R. Cordy, T. R. Dean
Robust multilingual parsing using island grammars
Conference of the Centre for Advanced Studies on Collaborative Research (CASCON 2003)

Selected References

C. Treude and M. Robillard
Augmenting API documentation with insights from Stack Overflow
38th ACM/IEEE Int. Conference on Software Engineering (ICSE 2016)

L. Ponzanelli, A. Mocci, M. Lanza
Summarizing complex development artifacts by mining heterogeneous data
12th Int. Working Conference on Mining Software Repositories (MSR 2015)

G. Erkan and D. R. Radev
LexRank: Graph-based lexical centrality as salience in text summarization
Journal of Artificial Intelligence Research, vol. 22, 2004

Selected References