Project 1: Bad Smells
Knowledge Analysis and Management

Brenda Ruiz

November 12, 2020

1 Ontology Creation

In this project we use ontologies to detect bad smells in code. Specifically, we analyzed the project *AndroidChess*, which is a mobile application to play chess. The types of bad smells to be detected are the following:

- Bloaters: code, methods and classes that have increased to very large proportions that they are hard to work with. These include long methods, large classes and long parameter lists.
- Object-Orientation Abusers: incomplete or incorrect application of object-oriented programming principles. For this type of code smell we detect the presence of Switch Statements in methods or constructors.
- Dispensables: something pointless and unneeded whose absence would make the code cleaner, more efficient and easier to understand. For example, a class which only has setter and getter methods (called a Data Class).

1.1 Class Hierarchy

The first step in this project was to create a class hierarchy in the ontology by parsing the file `tree.py` from the Javalang library. This file defines a syntax tree with different subclasses which represent elements in Java code. For example, the different types of statements in the Java language such as `IfStatement` or `ForStatement` are represented as subclasses of `Statement`. This hierarchy therefore represents the different syntactic elements found in Java code. The resulting hierarchy can be observed in Figure 1.

1.2 Ontology Statistics

During the ontology creation, different properties were created for each class. These were object and data properties. For methods the properties `body` and `parameters` were created to store individuals corresponding to the method's body and to its parameters. The data properties created were 65 in total and these correspond to other properties defined in the classes which are not `body` or `parameters`.

1. Number of classes: 78
2. Number of data properties: 65
3. Number of object properties: 2

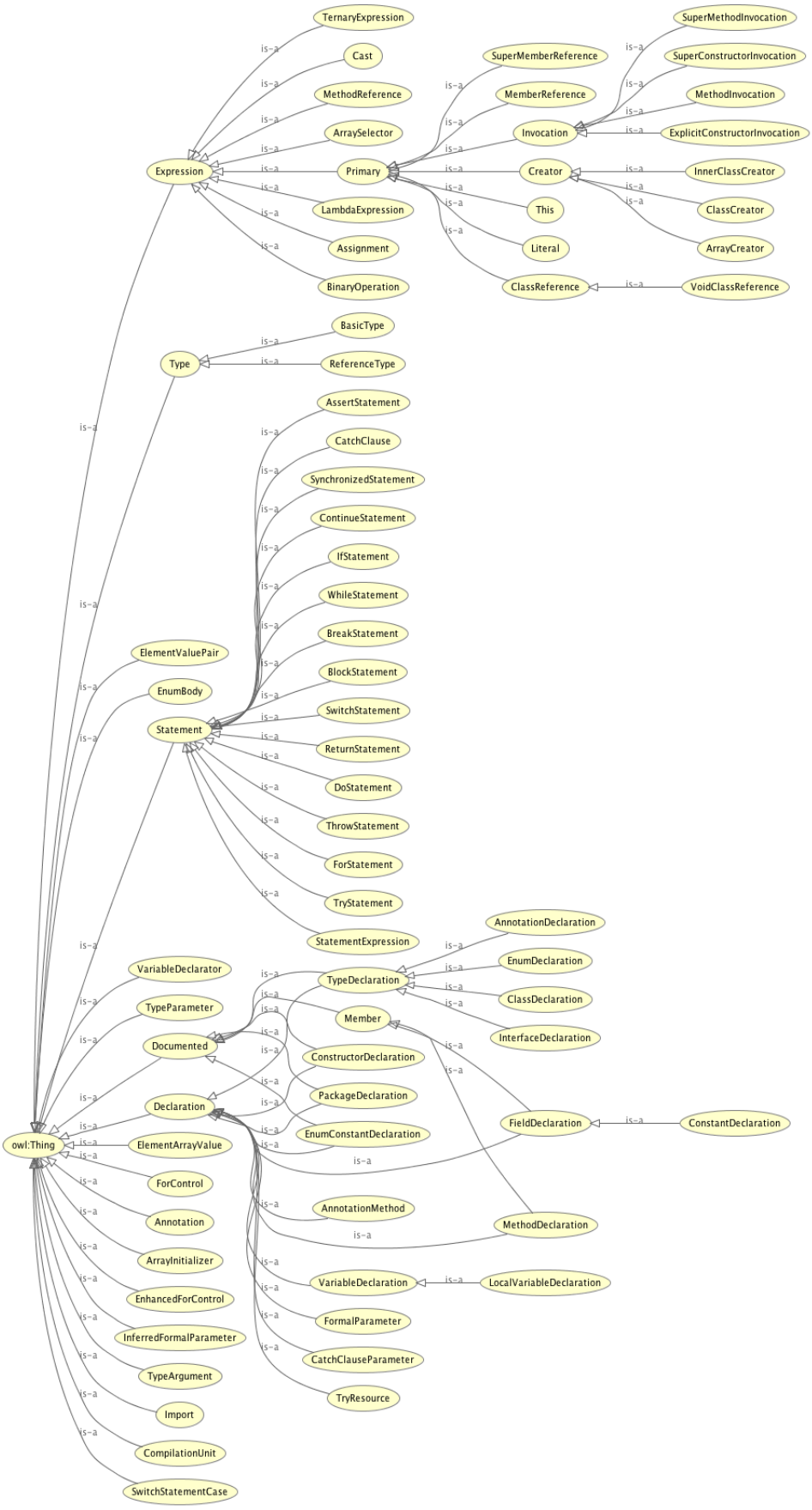

Figure 1: Class Hierarchy

2 Creation of Ontology Instances

The ontology was populated with instances of the classes created in the first step. The number of individuals created is reported in Table 1.

1. Number of individuals created: 1,344

Class	Number of created individuals
Field Declaration	105
Constructor Declaration	6
Formal Parameter	165
Method Declaration	152
Class Declaration	11
Block Statement	143
Break Statement	23
Catch Clause	8
Continue Statement	4
Do Statement	2
For Statement	6
If Statement	125
Return Statement	106
Statement Expression	446
Switch Statement	8
Synchronized Statement	1
Throw Statement	15
Try Statement	8
While Statement	10

Table 1: Number of individuals per class

3 Bad Smell Detection

The bad smell detection step was carries out by querying the ontology through SPARQL queries. The results are shown on Table 2 below. The project *AndroidChess* does not seem to have a lot of bad smells. The type of bad smell that was the most prevalent was long methods with 10 occurrences. The rest of the bad smells had lower occurrences, with long constructors, constructors with switch and constructors with long parameter list with no occurrences. The following tables below break down the instances of each bad smell found in the project.

Bad smell	Count
Long methods	10
Long constructors	0
Methods with switch	8
Large classes	3
Constructors with switch	0
Constructors with long parameter list	0
Methods with long parameter list	4
Data classes	1

Table 2: Bad smells (total)

Class Name	Method	Number of statements
PGNProvider	insert	31
ChessPuzzleProvider	query	25
ChessPuzzleProvider	insert	20
GameControl	loadPGNHead	26
GameControl	loadPGNMoves	96
GameControl	requestMove	76
GameControl	getDate	26
JNI	newGame	35
JNI	initFEN	88
JNI	initRandomFisher	87

Table 3: Large methods

Class Name	Number of methods
GameControl	63
JNI	44
Move	21

Table 4: Large classes

Class Name	Method name	Number of statements
PGNProvider	query	1
PGNProvider	getType	1
PGNProvider	delete	1
PGNProvider	update	1
ChessPuzzleProvider	query	1
ChessPuzzleProvider	getType	1
ChessPuzzleProvider	delete	1
ChessPuzzleProvider	update	1

Table 5: Methods with Switch statements

Class name	Method name	Number of parameters
PGNProvider	query	5
ChessPuzzleProvider	query	5
GameControl	setCastlingsEPAnd50	6
JNI	addPGNEntry	5

Table 6: Methods with long parameter list