Intelligent Tutoring System for New Java Learners - Variant One
Background and Spedification Progress Report

Pîrvulescu Miruna-Ioana
email: miruna-ioana.pirvulescu@kcl.ac.uk

December 17, 2020

Contents

1 Background and context

1.1 Introduction

An **Intelligent Tutoring System** is an expert system that guides students throughout their journey of learning a new subject. Such systems fall under the field of artificial intelligence, but they are strongly related to the world of cognitive sciences as well.
Knowledge Modelling is a technique of taking human-comprehensible information and making it machine-readable for the purpose of enhancing an expert system.
Ontology Design is one of the many ways to create a knowledge model and it organises the information in a hierarchical fashion. The term *ontology* is defined as "a particular theory of the nature of being or existence". (Russell, Norvig, 2016)[1]

1.2 Background

The purpose of this project is to create an Intelligent Tutoring System that can teach 1st-year students how to write code in the Java programming language. The specific concern of this variant of implementation focuses on the domain model, which is created with the use of knowledge modelling. This enables the system to understand the domain, present it to learners, and make inferences.

1.3 Context

The subjects covered by this system will teach students the basic rules of Java, such as syntax and the meaning of the introductory key terms, but it also goes in more depth and explains the different types of supported statements, such as conditional and repetitive statements. Implicitly, the system has to teach a student what is a variable, a type, and how to write basic arithmetic operations, as all this information is necessary for the understanding of statements.
The knowledge model of the system was created using ontology engineering, which is the best suited type of knowledge modelling design for this domain.
While the back end is represented by the knowledge model, a well-rounded Intelligent Tutoring System has a few more components, such as a student model, a pedagogical model, and an interface model. All these additional elements fall out of the scope of the project and will be replaced by a simple front end, designed as a web application. This front end will allow the visitor to press any button from a provided selection, which directs the tutoring system to reply by reasoning on the knowledge model and produce inferences.

2 Relevant Literature and Documentation

2.1 On Knowledge Modelling

When I started the research phase, I discovered three very important resources that introduced me to the meaning of knowledge modelling and representation. First, with the use of Wikipedia[2], I started scratching the surface of the subject and I learned the basic concepts behind what knowledge means in the world of Artificial Intelligence, as well as what are the ways of making human knowledge machine-readable. The page introduced me to some history of knowledge representation, its characteristics, as well as its properties. Most importantly, the page discusses the importance of ontology engineering during the growth of the field, which gave me an initial idea about how I would like to formalise my model.
Further on in my research, I came across a paper from University of Belgrade[3] that introduces knowledge modelling for newcomers to the field. The first talking point is a description of the different types of knowledge that can exist in a knowledge base, and it then dives into techniques of developing a design for such a structure. Some of the detailed elements refer to ontology design, knowledge management, and knowledge processing, and there is a significant amount of additional material that falls out of this project's scope, which can be used for a more consistent domain.
The most important source of information that I use is an online course taught by Dr. Tish Chungoora[4], where he introduces the practical design and development of a knowledge model. Here, I learned how to build a model from scratch with the use of ontologies. This course interested me very much as, by the time I came across it, I already made up my mind about using ontologies. The fact that this course teaches the basics of Protégé as well was a very compelling argument to use it as source material, and it proved very helpful in the early stages of building the ontology.

2.2 On Ontology Design

In the process of selecting a formalism for knowledge representation, I have assessed a number of alternatives. Starting with the winner, I will now briefly discuss each of them and say why they were (or were not) selected.

2.2.1 Ontologies

Ontologies encapsulate the formal design and definition of a class hierarchy. They are widely used within the field of education[5], as well as in knowledge representation and design.
Due to the rigorous hierarchical nature of ontology engineering, the popularity among engineers in the field, and because of its capacity of combining education and knowledge modelling elegantly, the use of ontologies is the best candidate for the development of this project.

2.2.2 Semantic Networks

Semantic networks (or semantic nets) are visually represented by graphs where the concepts are the vertices and the relations are the edges[6]. They share a lot of similarities with ontologies, however the they do not seem to have as rigorous a formalisation.
Because of this, I have decided not to move on with semantic nets as the ideal modelling technique. However, I did not discard them completely, as their visual interpretation technique is a very important component of my initial ontology design. The first lightweight ontology of the domain was created in the form of a mind map, which resembles the visual component of a semantic net very well.

2.2.3 Frames

Frames are very widely used for knowledge modelling in artificial intelligence. They create smaller ontologies that are interconnected to create the bigger picture.
This type of formalisation is a better fit for larger taxonomies than the one provided by the project's domain, and therefore its usage would have brought a risk of overcomplication. However, for the encapsulation of a larger variant of this domain, it would be a very good fit.

2.2.4 Systems Architecture

"Systems Architecture is a response to the conceptual and practical difficulties of the description and the design of complex industrial systems" (Golden, 2013)[7]. An example of such a complex system in the context of education is the coalesced material taught in every module that King's College London teaches in the Faculty of Natural and Mathematical Sciences.
Coming back to the point from the previous subsection, the domain of this project is not complex enough to fully leverage such a type of formalisation, which means that system architecture cannot be the optimal way to implement the knowledge model.

2.2.5 Rule-Base

As the name suggests, a rule-based system uses a set of rules in order to create its inferences. These use a classic set of if-else statements in order to teach the expert system about the domain, just as an expert system is usually constructed.
While Java obviously follows a set of rules, they are not rigorous that a rule-based system is automatically the best way to approach it. A conditional statement can do anything as long as the condition it receives is a boolean, an array can be of any type as long as it is recognised by the program, and there is almost no limit on the number of atomic preconditions that a statement can have.
Moreover, a general comparison study between the two suggests that rule-based systems are less descriptive than ontology-engineered ones[8].

2.2.6 Conclusion of the Comparison

The conclusion that I drew was that ontologies are the best fit for creating a knowledge model for Java, first and foremost because they are great for educative representations. Also, Java is a hierarchical programming language, so its concepts are very easy to model as hierarchical classifications.
Another good reason is that the other types of formalisation either rely on some level on the use of ontologies, are too complex for this domain, or they simply do not fit the final requirement. As the domain is not very extensive, using a more complex approach would render the project more difficult to implement than necessary, and the most beautiful piece of software the one with the simplest implementation.
There is also an element of subjectivity which cannot be avioded. Since I am the domain expert in this circumstance, my knowledge needs to be passed on to the knowledge model. Knowledge is best passed down when thoroughly understood by the teacher, otherwise the quality of the model becomes suboptimal. My best grasp of Java comes in the form of hierarchical classification, and ontologies are the best way for me to transmit my expertise to the machine.

2.3 On Recommendations

Given the recommendations I received from my project supervisor, Dr. Sahar Al-Sudani, I gravitated even more towards ontology design. By the time I received these recommendations, I was already inclining towards ontology designs, but the proposed software tools (such as the Protégé ontology editor) that I could use to put theory into practice were a definite bonus. These tools, as well as some further reading materials[9] and the encouragement to pursue the online course taught by Dr. Chungoora, represent the valuable support and feedback that contributed to the design of the knowledge model.
Another very important recommendation comes from Dr. Frederik Mallmann-Trenn, the lead lecturer of the Artificial Intelligence Reasoning and Decision Making module that I am taking, who pointed me towards Russell and Norvig's "Artificial Intelligence: A Modern Approach" for more information on knowledge representation and ontology engineering.

3 All The Details

This section covers the specific details of the project, namely the implementation and all the steps taken towards it. I will discuss the different elements that contributed towards the creation of the current design, from concept diagrams to actual implementation.

3.1 Development of the Ontology

While the core task of this project is to implement the intelligent tutoring system, this cannot be done without having developed the ontology first. After going through the research phase mentioned in the previous section, I started building the pillars of my ontology.
The most helpful piece of research for this implementation was the module taught by Dr. Chungoora, which describes the six most important steps in the development of a knowledge model. The results I got from following this module are pleasing and I am continuously using these materials for further development and recommendations.

3.1.1 Goal and Scope Definition

The most important initial task was to define the goal and the scope of my project. This helped me thoroughly understand what I have to do and how I want to tackle the tasks at hand.
While this task may seem unnecessary or time-consuming, it is essential to go into development with a clear idea of what needs to be done. I found this step very helpful in creating the project overview. This definition can be found in Appendix A.
Out of the covered objectives, the lightweight ontology has been built and completely transferred into a heavyweight, formalised ontology. Publishing and feedback collection refer to testing with the help of a group of individuals, and it is a development action that I plan on taking soon.

3.1.2 Information Gathering and Elicitation

Since I have to make the system teach introductory Java, I have decided to use, as the main documentation material, a website that helped me personally throughout my journey as a programmer. This website is called W3Schools[10] and it is an open-source teaching platform for coding. I used in the past to learn web development, and the materials covered on Java programming are satisfactory for this project.
I used this website to extract keywords that later turned into classes and relations in the lightweight ontology. This term classification was done with the use of a spreadsheet where each keyword was assigned a short description, the type of entity that it was going to become (class, relation, other), and a verdict of addition (that is, whether the keyword was to be considered for later development). The spreadsheet consists of 98 such terms, and an excerpt of it can be

found in Appendix C.
During this step, I also started creating the list of competency questions that the software needs to be able to reason about. These questions cover as much as possible on the topic of statements, from general questions about them to more specific questions about the different types of statements (conditional, repetitive, assignment). For a rounder knowledge base, I plan on adding some questions about other relevant topics to statements, such as variables and types. The list below is not yet exhaustive; it grows together with the ontology itself.

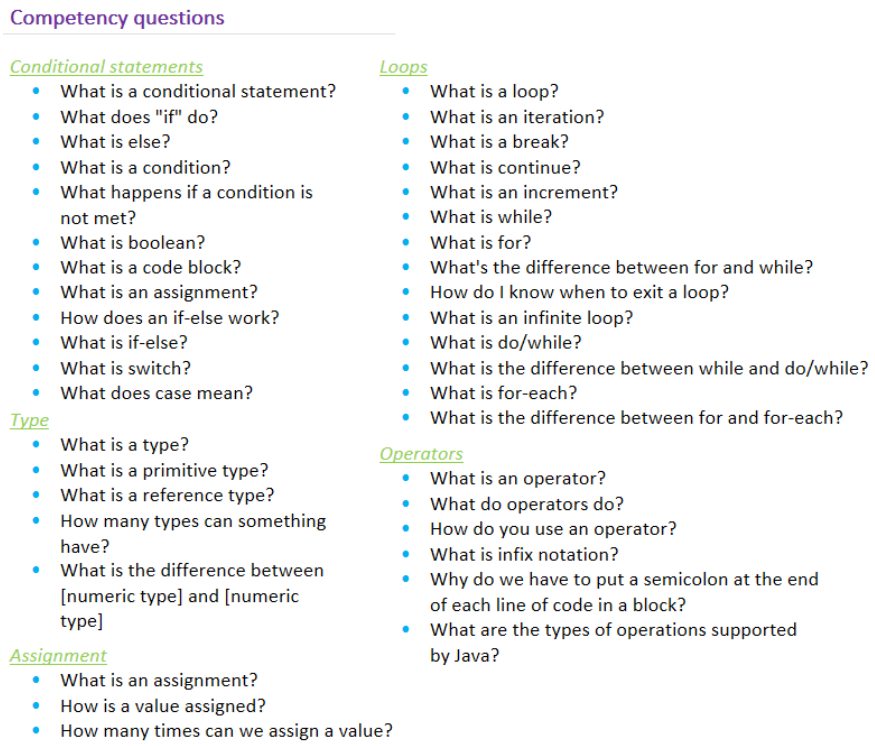

Figure 1: List of Competency Questions (per category)

At the advice of Dr. Al-Sudani, I check whether the reasoner can answer a few targeted questions at a time, and when it manages to answer all the selected questions, I move on to another few. I select the questions based on category, which allows me to concentrate on one sub-topic at a time.

3.1.3 Initial Structuring

This is the step where the lightweight ontology is created. This is a human-comprehensible representation of the knowledge model, and having such a structure in place allows for a clean formalisation of the software.

I started working on this model by taking a top-down approach, where I considered the general topic and went into further detail for each sub-topic. This lead to the creation of a mind map, which resembles a visual semantic network. The software used to create this map is provided by Mural[11], and a copy is available in Appendix B.
While this is a very informative approach, it is by far not the best one to model the lightweight ontology. It does not cover the proper relation in between classes. Therefore, I only used it as an initial stepping stone in the process of building something a little more formal.
Following this visual, I created a UML diagram which allowed me to further formalise the class hierarchy and the relations between them. For this step, I used Enterprise Architect, a software tool that specialises in visual representations[12].

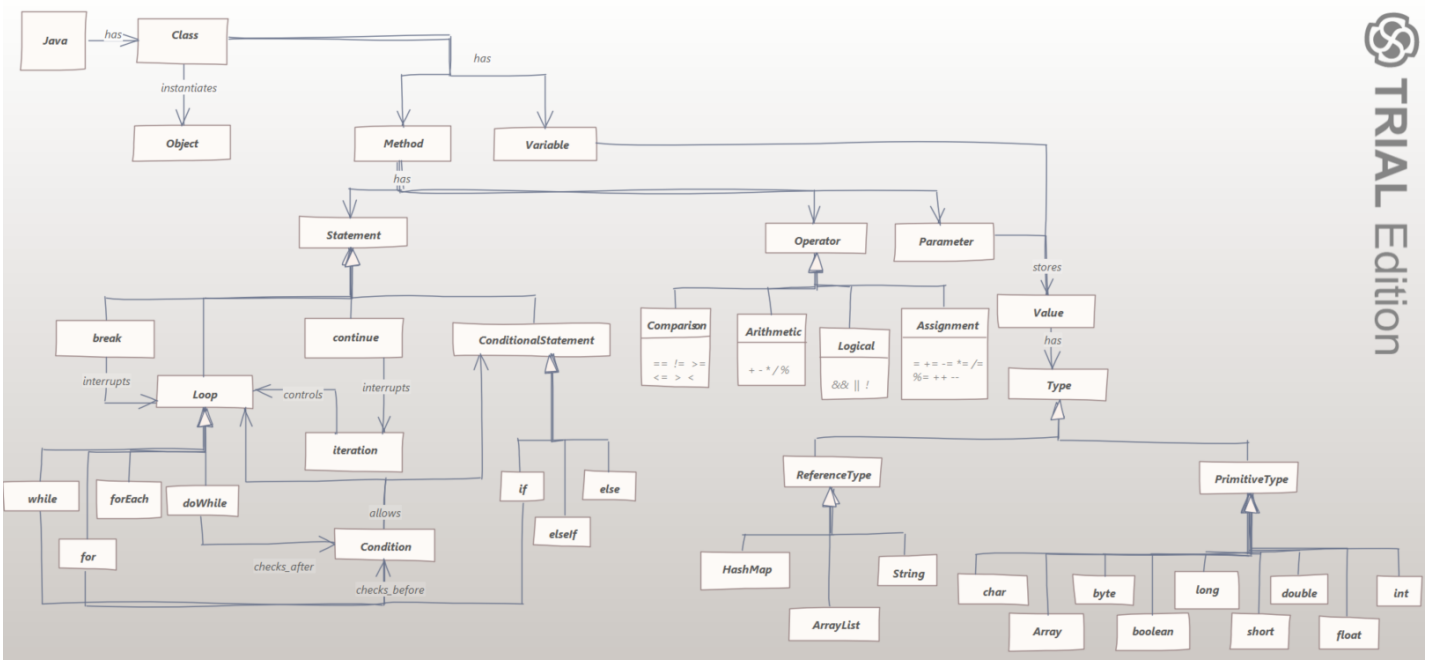

Figure 2: Formal Domain Structure: UML

As seen in the figure above, this visualisation is cleaner and it better defines the taxonomy. This model covers all the sub-aspects of Java statements.
In the following steps, I plan to further enhance this model by making it as accurate and simple as possible and by adding some more sub-topics of Java.

3.1.4 Formalization

I started developing the formalised ontology using the Protégé 5.5.0 environment and OWL as the development language.
The current stage of the ontology consists of twelve main classes which have a combined total of 31 subclasses, all of which are properly disjoint and commented.
Alongside these, there are 22 properties (relationships), including inverses. Every property has been rightfully characterised (as irreflexive, asymmetric, functional, or inverse-functional, according to case). The fact that this ontology is so rigorous allows just enough room for the reasoner to make correct inferences. Currently, I am working towards modelling the available knowledge in such a manner that the competency questions can be accurately answered.
The latest task accomplished on the formal ontology was choosing a reasoner. The purpose of a reasoner is to ensure that all the classes from the ontology can be instantiated without problems and that there are no inconsistencies in the logic behind the development, as well as to create inferences on the ontology.
The original choice for the reasoner was Racer[13], which was recommended to me by Dr. Al-Sudani. However, due to its deprecation in the more recent versions of Protégé, I have chosen HermiT 1.4.3.456, a top reasoner that became very popular in knowledge reasoning over the recent years[14].
Since the tutoring system needs to be "intelligent", rather than giving it all the necessary information, I am allowing the reasoner to infer a few details of properties. Such an example would be having an inverse property and letting the reasoner infer the domain and the range. The only case in which HermiT cannot infer one of these sets is when they are created as disjunctions of classes.
On the topic of instances, I have decided, at least for now, not to add any. I was going to represent some explicit terms as instances, however there was an issue: an instance cannot be assigned a propety. In other words, if I were to take a term from the pool and make it an instance, I could not create a relation in between it and something else. For example, if the `for` statement was an instance rather than a class, I could not have added the fact that it is a kind of statement that checks a condition before executing an iteration (unlike `do/while`), or that it can be interrupted in different ways by using the `break` and `continue` statements.
As the ontology is still growing, I am using Dr. Chungoora's course, as well as "A Practical Guide To Building OWL Ontologies"[9], to tackle any appearing problems and learn how to further develop the knowledge base. The current components of the ontology can be found at the end of this section.

3.1.5 Deployment & Evaluation

These refer to testing and feedback, which is planned to start around the January examination period. In the lifecycle of ontology development, the purpose of these steps is to help refine the ontology until it answers the competency questions accurately and comprehensively.

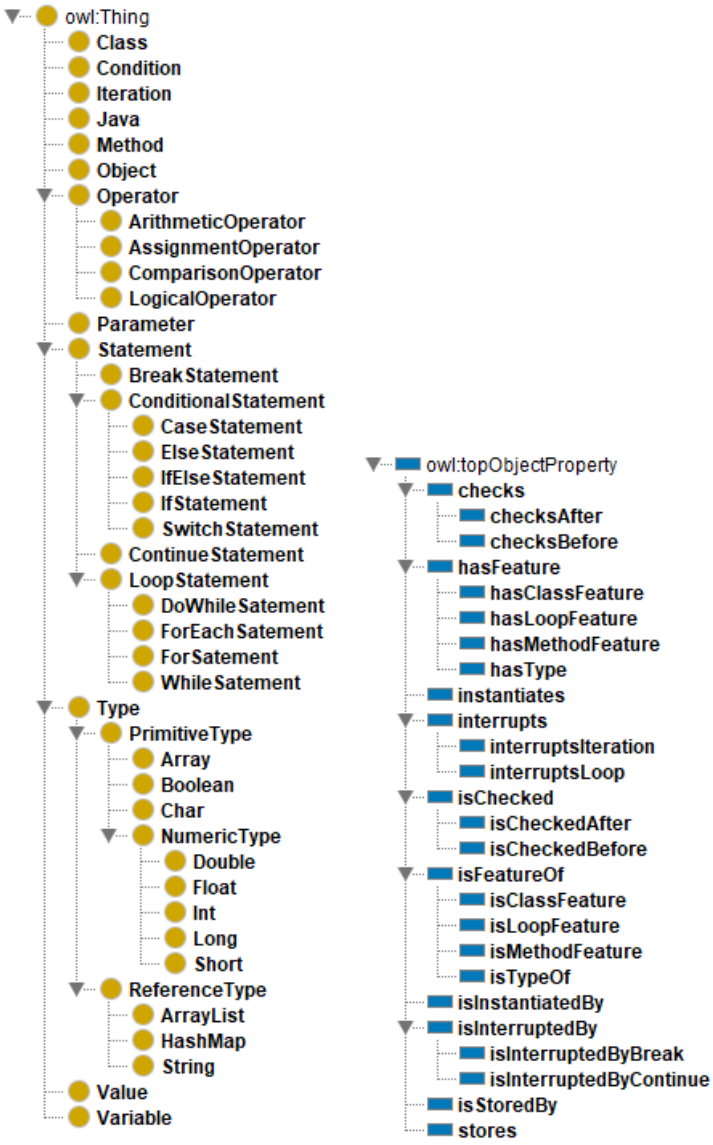

Figure 3: Current Classes and Properties of the Ontology

3.2 Building the Intelligent Tutoring System

This requirement is met by a website that I am going to build until the following meeting. I want to implement a front end that has a selection of buttons representing competency questions to be answered by the reasoner.
I have not yet chosen the technology for the front end. Mainly, I would have liked to use React.JS, but Java is a smarter technology to use here as it would allow for a smoother link between the front and back end. Every other component of the design already uses Java.
The domain model is vital for the existance of the tutoring system, which means that it took higher priority in development. While I am enhancing the domain model however, I want to start working on the front end as well, and by the time of the next progress meeting in February, I want to have a solid beta version of the deliverables.

4 Future Additions

The first future step is to finalise the current version of the ontology so that most of the competency questions can be answered. During this time, I want to also create the front end and link it to the knowledge base. Once this system becomes functional, I want to enhance the knowledge model, and later go into more optional tasks and make the ITS visually pleasing.
Provided I can be granted this approval, I would like to work on some user input for the phases of deployment and evaluation. This would mean sending the software for testing to a group of individuals with different levels of experience. Each individual receives a package based on their level of familiarity with Java: the software and a level-based survey. A beginner would be asked questions such as how comfortable do they feel with Java and a quiz on some of the concepts, and someone more advanced can provide input on the system's scientific accuracy. Allowing individuals with different levels of experience to review the software would allow for well-rounded feedback.

5 Conclusion

The project is in the middle of development at the moment, and I have a very clear idea of what I want to work on next. While the ontology is not yet finalised, it manages to answer a number of competency questions, but in the future it will be able to answer even more. As there is no one correct way to develop this knowledge model, the current ontology will change over time, although most of the change will consist of additions rather than alterations.
The reviewed literature was very helpful in developing the formal ontology, and it also influenced the list of next steps in the design. As there are some different types new of components to be added, the literature list is going to contain some new references on Java web development and more general information on Intelligent Tutoring Systems.

References

[1] Russell & Norvig, 2016, Artificial Intelligence: A Modern Approach, Pearson; 3rd edition, p 308

[2] Wikipedia.com, Knowledge Representation and Reasoning

[3] Devedzic, 2001, Knowledge Modeling - State of the Art

[4] Chungoora, 2020, Practical Knowledge Modelling: Ontology Development 101, udemy.com

[5] Wikipedia.com, Ontology (information science), Overview

[6] obitko.com, Ontologies and the Semantic Web, Semantic Networks

[7] Golden, 2013, A Unified Formalism for Complex Systems Architecture, p 14

[8] Czarnecki & Sitek, 2013, Ontologies vs. Rules — Comparison of Methods of Knowledge Representation Based on the Example of IT Services Management, p 108

[9] University of Manchester, A Practical Guide To Building OWL Ontologies Using Protege 4 and CO-ODE Tools, Edition 1.3

[10] W3Schools.com, Java

[11] Mural.co

[12] Enterprise Architect

[13] The Racer Reasoner, University of Lübeck

[14] The Hermit Reasoner, University of Oxford

Appendix

A Initial Project Structuring

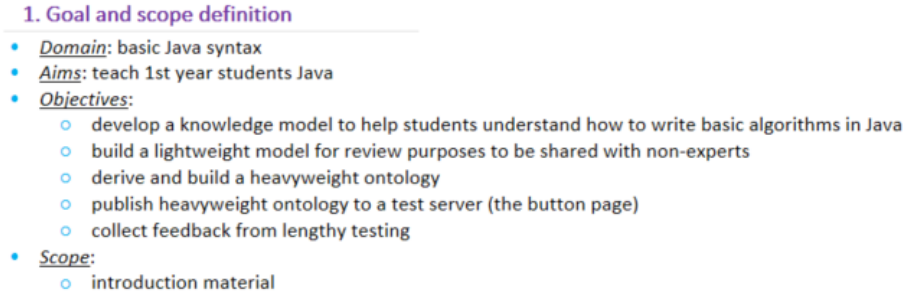

Figure 4: Board of project initial structuring

B Initial Mind Map

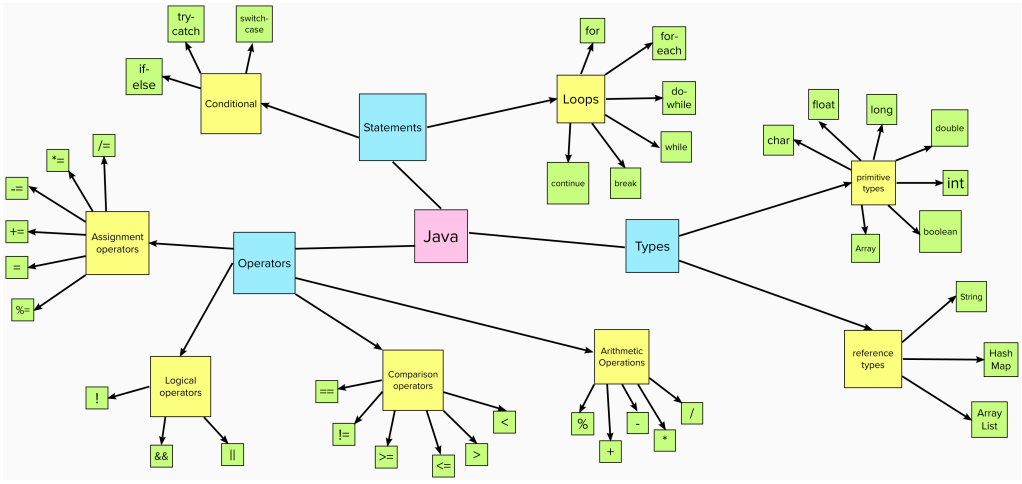

Figure 5: Initial Domain Structure: Mind Map

C Information Gathering Spreadsheet

TERM POOL						
Term	Description	Source Reference	Prospective Entity	Synonyms	Arguments	Action
Java	Java is a programming language	Wikipedia article	Class			
class-based	class-based means it works on classes	Wikipedia article	Other			
object-oriented	type of programming	Wikipedia article	Other			
programming	interacting with the computer	Wikipedia article	Relation		program	
language	means of programming	Wikipedia article	Class			
compiled	(very grossly) processed	Wikipedia article	Relation			
version	published release of code	Wikipedia article	Other			
open-source	free	W3 schools	Other			
line	a line of code	W3 schools	Class			
code	the "phrases" passed to a computer	W3 schools	Class			
run	process of compilation	W3 schools	Relation		code, computer	
class	a user-defined kind of thing	W3 schools	Class			
object	instance of a class	W3 schools	Class			
name	the name of a thing	W3 schools	Other			
comment	uncompiled data; can talk about what is going on in your code	W3 schools	Class			
single-line	//one line of comment only	W3 schools	Other			
multi-line	/*multiple lines of code*/	W3 schools	Other			
variable	something that can be changed	W3 schools	Class			
type	the type of a variable	W3 schools	Class			
string	a string of characters	W3 schools	Class			
text	combination of characters	W3 schools	Other			
double quotes	""	W3 schools	Class			
int	integer primitive type	W3 schools	Class			
float	decimal numbers	W3 schools	Class			
decimal	0.1, 0.2, 0.3 etc.	W3 schools	Other			
char	character	W3 schools	Class			
characters	A, a, B, b, etc.	W3 schools	Other			
single quotes	'	W3 schools	Other			
boolean	something that is either true or false, never both	W3 schools	Class			
equal	a = b	W3 schools	Other			
assign	attribute a value to a variable	W3 schools	Relation		variable, value	
create		W3 schools	Relation		anything	
final	cannot be changed	W3 schools	Other			
static	same value across all objects, even when	W3 schools	Other			
primitive	old-fashioned types	W3 schools	Other			

Figure 6: The Keyword Spreadsheet