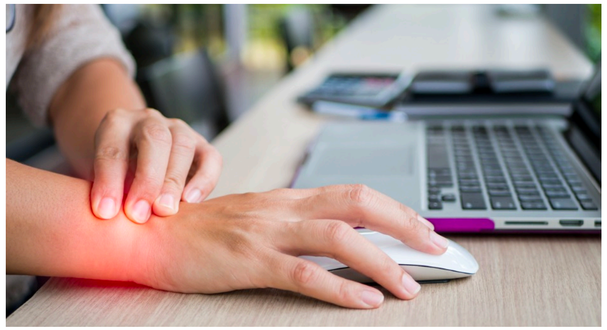
VoiceJava: Syntax-directed Programming by Voice

Tao Zan

2021.12.14

Why do we need voice coding ?

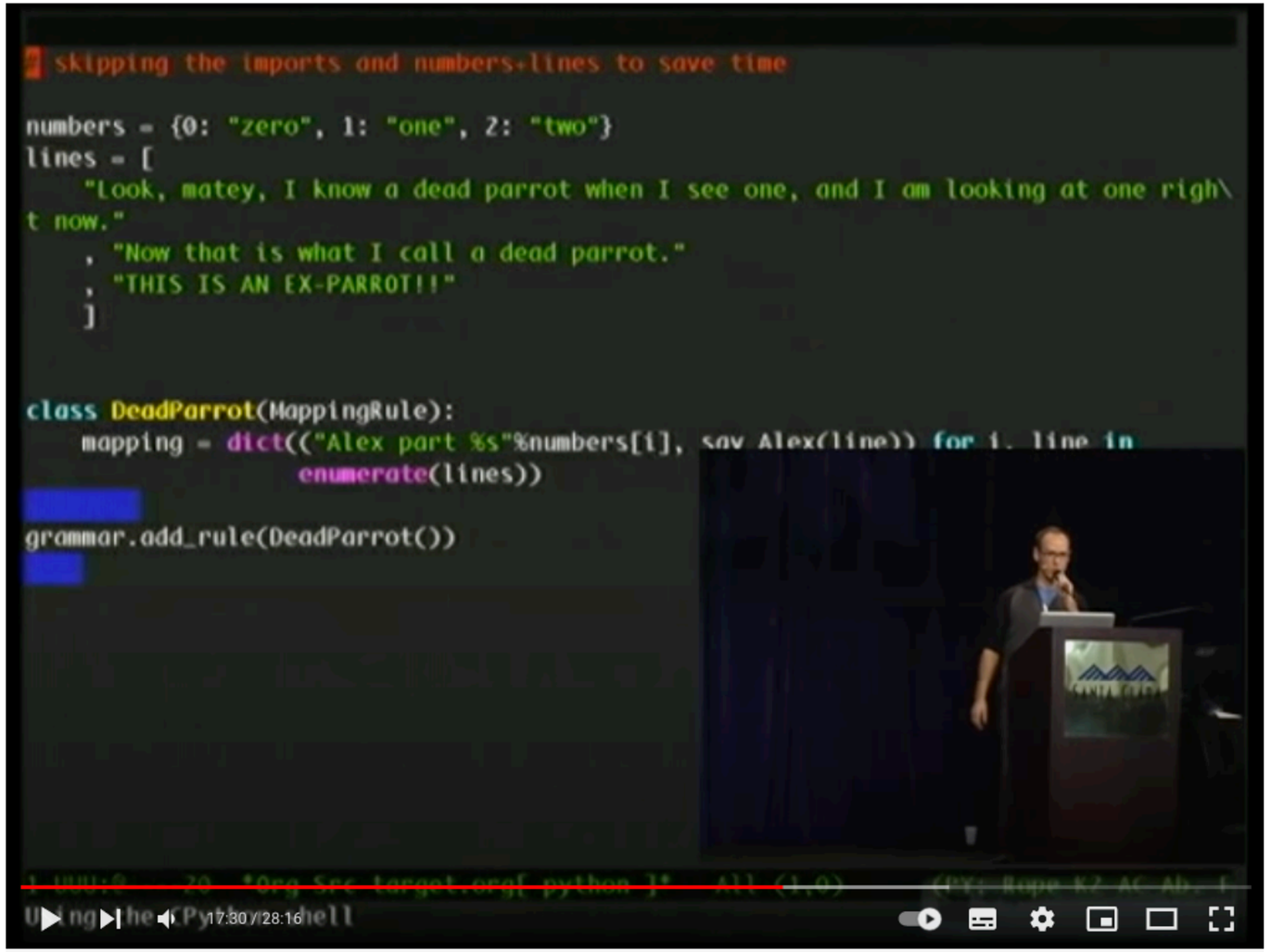
- Repetitive strain injury (RSI)



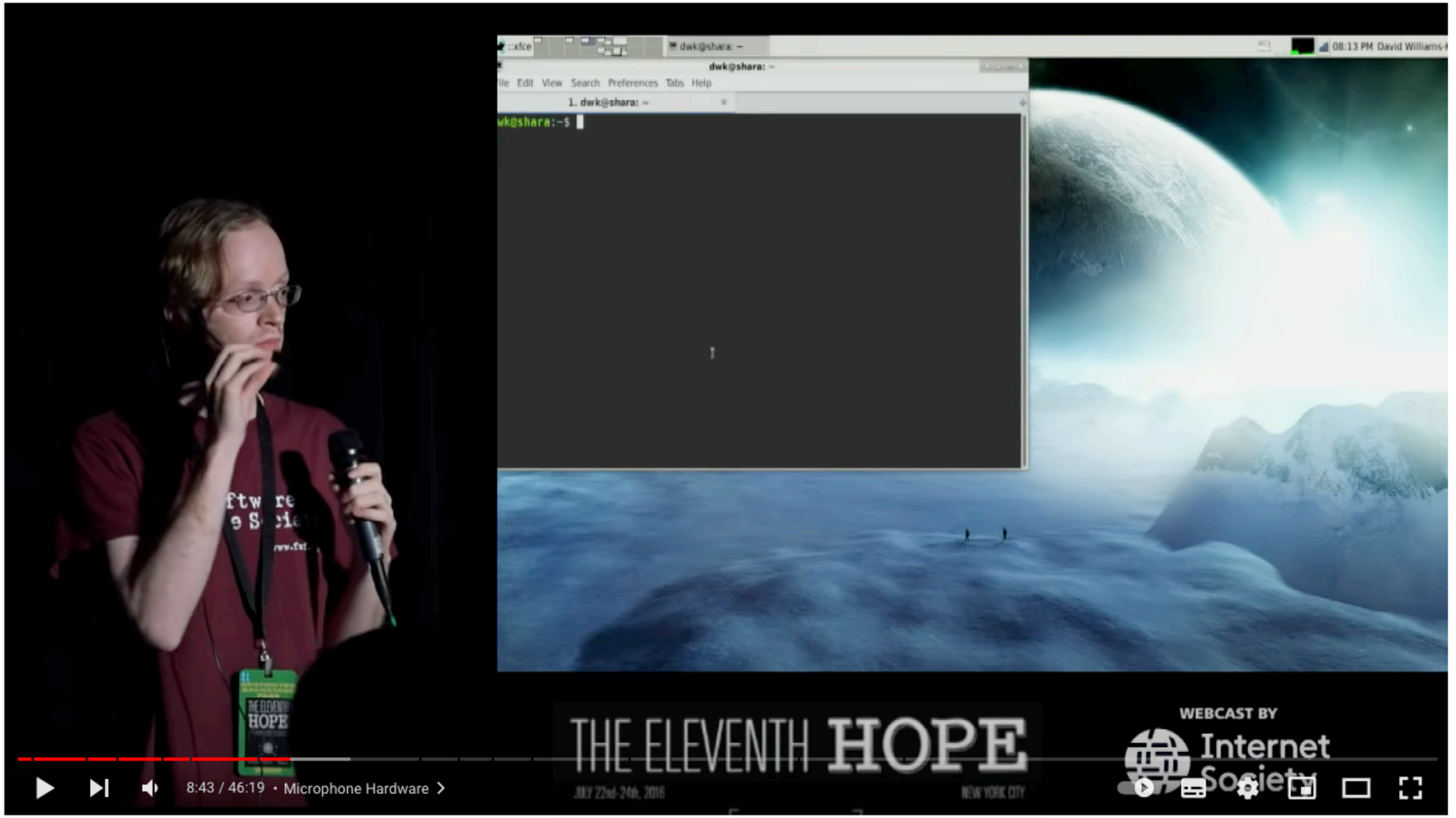
- Handicapped，but smart

Related Project

- Tavis Rudd: Code by voice with Python, 2010
- Silvius, 2016
- TalonVoice, 2018



Based on Dragonfly: https://dragonfly.readthedocs.io/en/latest/



https://voxhub.io/silvius

Silvius Comand

Letters 26

arch bravo charlie delta echo fox golf hotel india julia kilo line mike november oscar papa queen romeo sierra tango uniform victor whiskey xray yankee zulu

Uppercase letters 1

sky arch

Cursor movement 5

up, down, left, right, up three

Chaining 1

Say any sequence of commands: charlie delta space dot dot slap

Individual words 1

word something (types full word in lowercase)

Multiple words 2

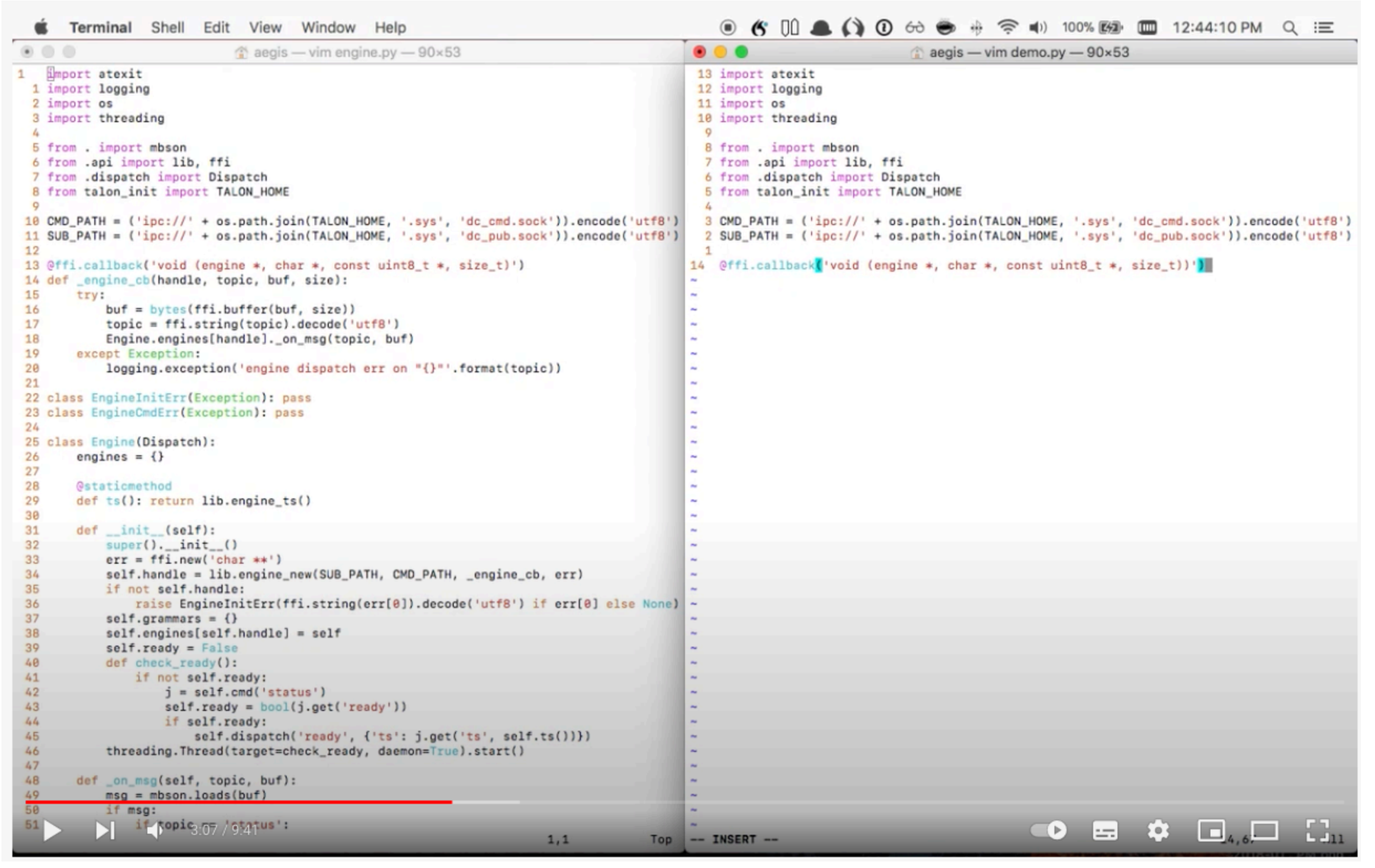
phrase hello there (all lowercase), sentence hello there (capitalizes first)

Basic editing 4

slap (newline), scratch (backspace), act (escape), slap three, etc

Other characters 23

act colon single-quote double-quote equal space tab bang hash dollar percent carrot ampersand star late rate minus underscore plus backslash dot slash question



Talon: https://talonvoice.com/

Related Work

- 1. Gonze et al. Coding with the voice:
  - Two phases recognition:
    - Structuring phase: dictate the structure of the code by a set of keywords
    - Naming phase: select one at a time, use a general purpose recognizer to recognize

```
def function openParenthesis variable comma variable comma variable closeParenthesis colun endline
   variable equal variable multiply 4 minus 3 multiply variable endline
   return variable endline

def function openParenthesis comma variable closeParenthesis colun endline
   variable equal zero enline
   variable equal function openParenthesis variable comma variable comma variable closeParenthesis endline
   if  variable equal zero colun endline
      variable equal one endline
   else colun endline
     variable equal zero endline
   return variable endline
```

Grammar: https://github.com/gonzeD/ingiVoice/blob/master/grammar/base.jsgf

Related Work

- 1. Gonze et al. Coding with the voice:
  - Web speech API: cannot use a custom grammer, not good for structuring phase
  - CMUSphinx's Pocketphinx.js: use python grammar as engine's grammar

Text uttered	no grammar	simplified grammar	full grammar
variable	one eight elif	variable	variable
equal	yield	equal	equal
five	while five eight if	five	five
endLine	in in eight	plus eight endLine	in eight in eight
if	if yield two one	raise	plus two
variable	while yield	variable	divide eight
colon	colon in	endLine else	multiply two
endLine	in nine two	colon endLine	endLine
while	while in	elif	while
variable	nine eight	variable	variable
colon	else colon	colon	colon
endLine	in two nine	endLine	endLine
return	three def in	continue	yield two
variable	comma yield	endLine	in variable
endText	two in two else	continue	endText

Author's conclusion: not sufficiently good for production use.

Related Work

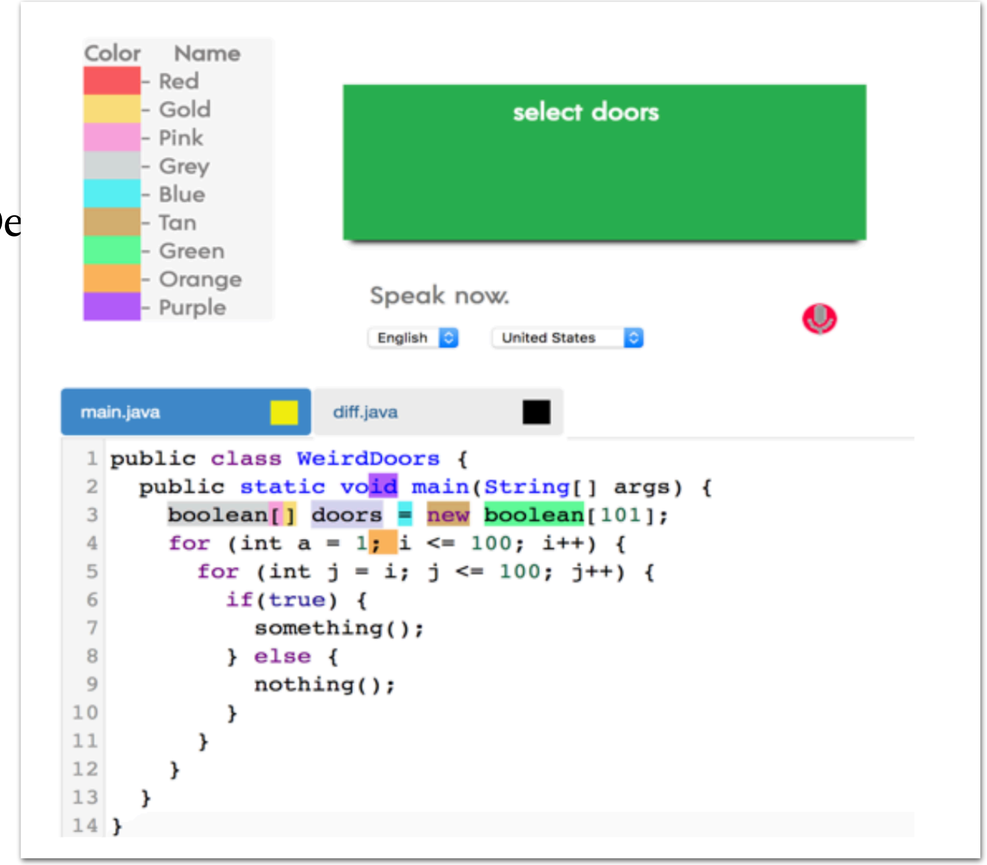
- 2. Lucas Rosenblatt, VocalIDE: An IDE for Programming via Speech Recognition
  - Wizard of Oz study:
    - Users will include filler words (“the”, “to”) that do not contribute to meaningful commands
    - Users are inefficient when navigating text using vocal commands
  - Implementation:
    - WebKitSpeechRecognition
    - Commands: Text Entry, Navigation, Text Selection, Replacement, Deletion, Undo, Smart Snippet Entry

      type open parenthesis i space less than space one close parenthesis

Related Work

- 2. Lucas Rosenblatt, VocalIDE: An IDE for Programming via Speech Recognition
  - Wizard of Oz study:
    - Users will include filler words (“the”, “to”) that do not contribute to meaningful commands
    - Users are inefficient when navigating text using vocal commands
  - Implementation:
    - WebKitSpeechRecognition
    - Commands: Text Entry, Navigation, Text Selection, Replacement, De
    - **Color Context Editing**

Author’s note: limited by insufficient speech recognition accuracy

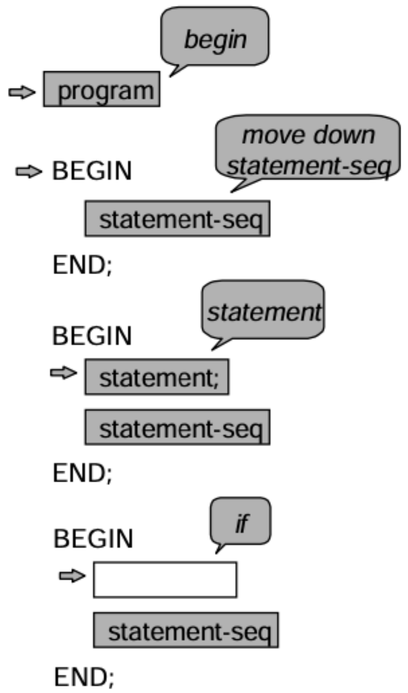


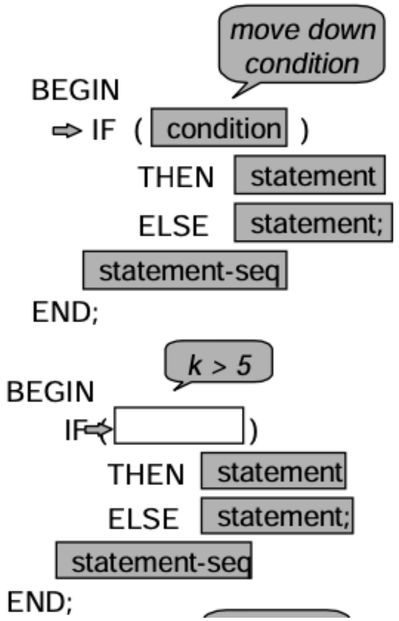
Related Work

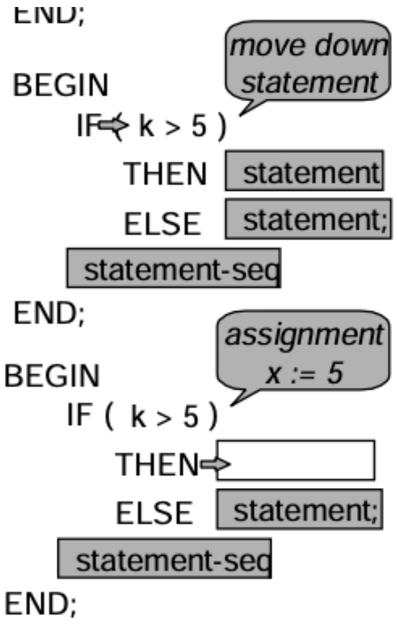
- 3. Ashish Shahane et al. V-IDE: Voice controlled IDE using NLP and AI and serveral related papers from India
  - Short paper, or lack of technical details, no source code

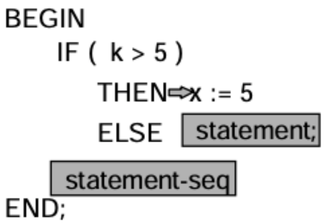
Related Work

- 4. Stephen C.Arnold et al. Programming by Voice, VocalProgramming, 2000
  - Aids the programmer by providing automatic completion of program text and appropriate navigation relative to the specific programming language



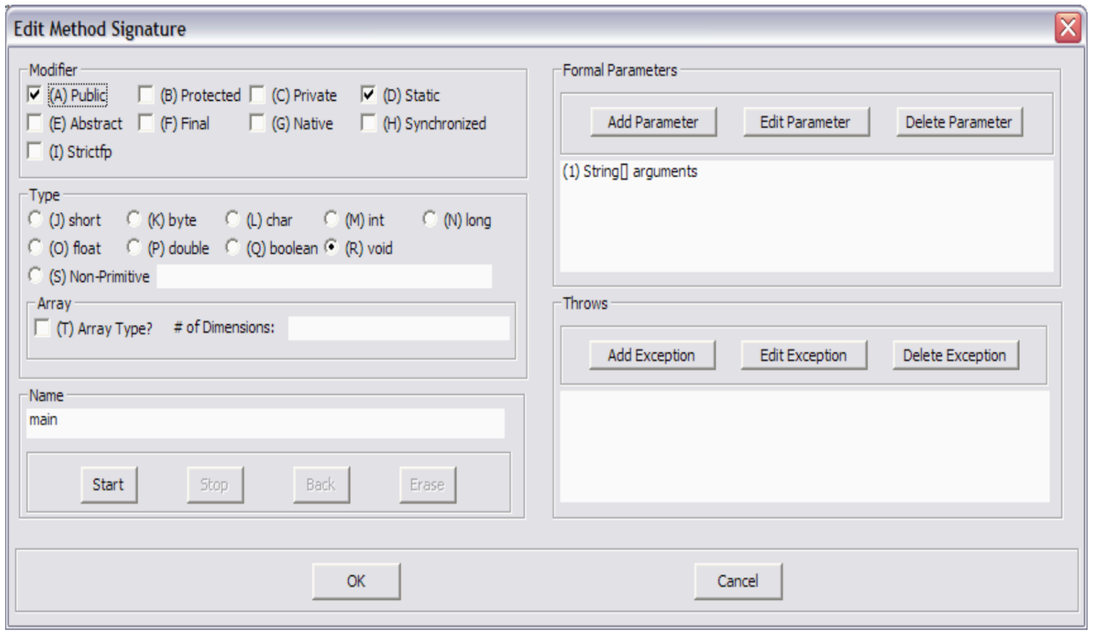




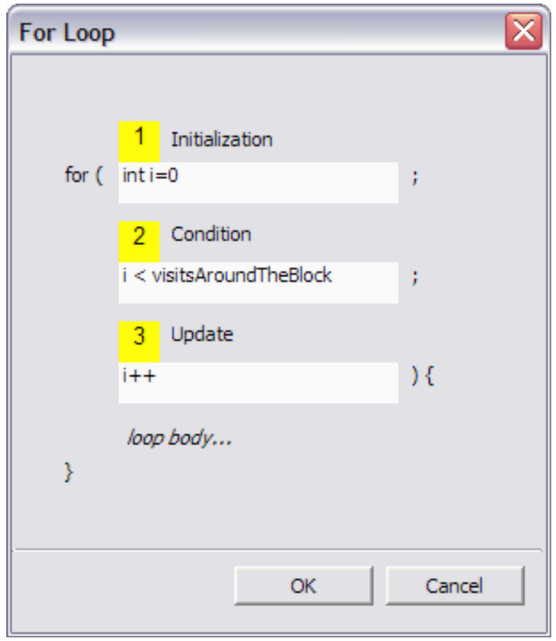


Related Work

- 5. Thomas Hubbell et al. A voice-activated syntax-directed editor for manually disabled programmers, 2006
  - By invoking a dialog (speech user interface, SUI)



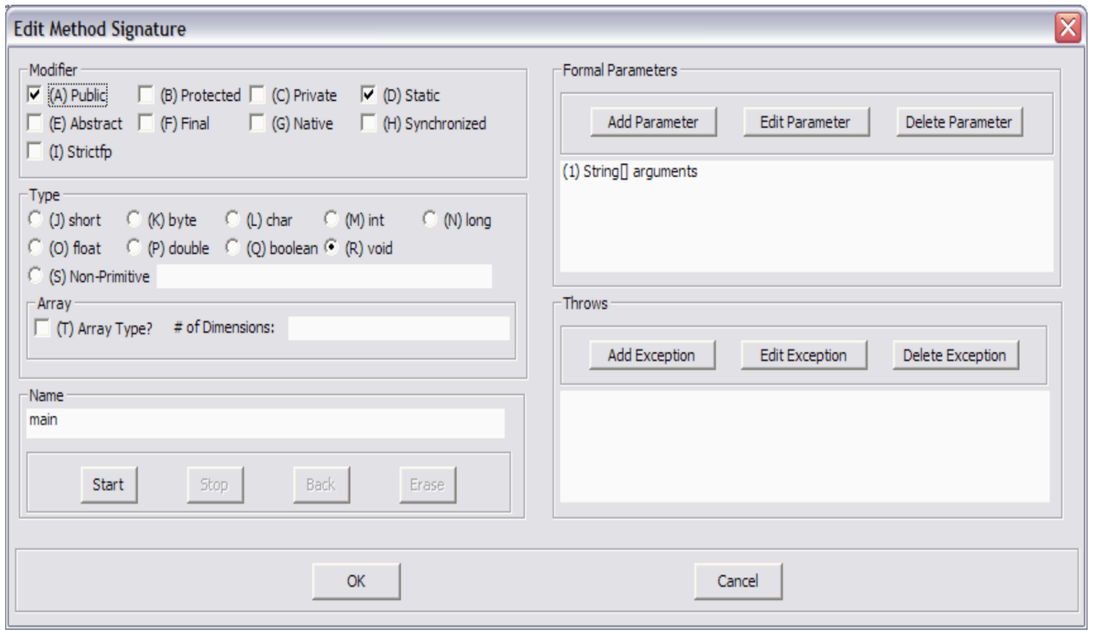
- Button labels are speakable, i.e. “select a”



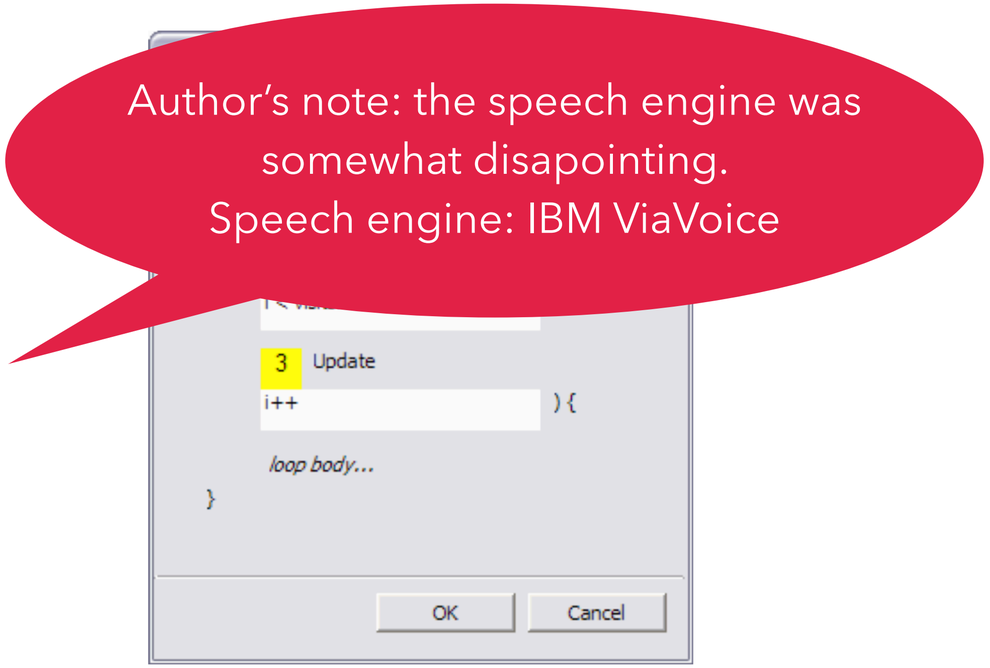
- Only permit user to edit the changable parts

Related Work

- 5. Thomas Hubbell et al. A voice-activated syntax-directed editor for manually disabled programmers, 2006
  - By invoking a dialog (speech user interface, SUI)



- Button labels are speakable, i.e. “select a”



- Only permit user to edit the changable parts

Problem

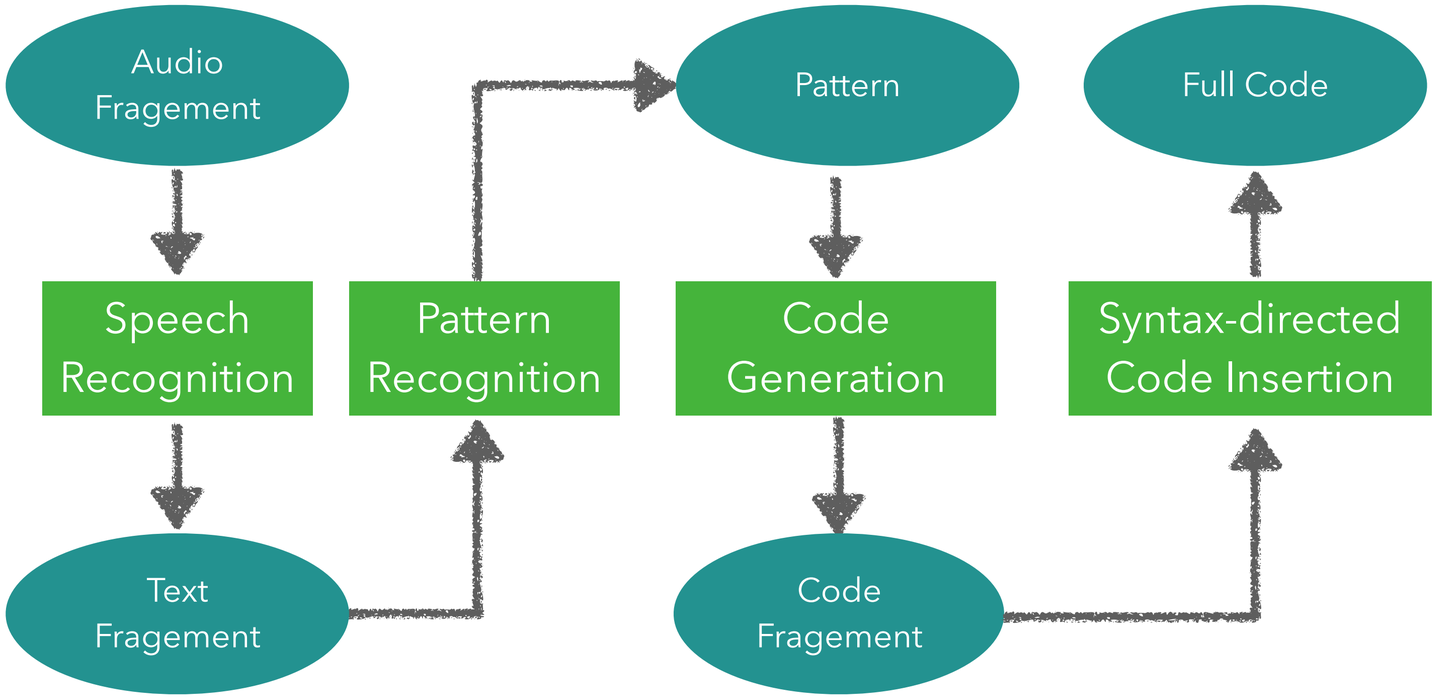
- Plain voice to text (1-to-1) translation, not voice to code.

  e.g. “return 0;” as “return space zero semmicon”

- Observation:
  - current programming languges are suitable for typing with keyboard, not suitable for speaking with voice.
  - we don’t really need to speak the same as the written code.

Demo

Architecture



VoiceJava Syntax

```
Name = [_]+ // _ means any word except keyword.
define package Name [dot Name]*
import static? Name [dot [Name | star]]*
define (Annotation | public | protected | private | abstract | static | final | strictfp )? interface Name
define (Annotation | public | protected | private | abstract | static | final | strictfp )* class Name [extends Name]? [implements Name]?
define (Annotation | public | protected | private | abstract | static | final | synchronized | native | strictfp)* function Name [throws Exception]?d
define (Annotation | public | protected |private | static | final | transient | volatile)* (Name list | Name [dot Name]? [with Name [and Name]*]?) variable Name
type (Name list | Name [dot Name]? [with Name [and Name]*]?) [extends _]?
type (Name list | Name [dot Name]? [with Name [and Name]*]?) variable Name
define [enchanced]? for
define [do]? while
define if
define switch
define try catch
break
continue
new instance Name [dot Name]*
throw new Name
let Name [dot Name]? equal call Name
let Name [dot Name]? equal Name [dot Name]* [call Name]+
let Name [dot Name]? equal Name [dot Name]*
let Name [dot Name]? equal [variable]? Name
let Name [dot Name]? equal (int | byte | short | long | char | float | double | boolean | string) Name
let Name [dot Name]? equal [expression]?
return call Name
return Name [dot Name]* [call Name]+
return Name [dot Name]*
return [variable]? Name
return (int | byte | short | long | char | float | double | boolean | string) Name
return [expression]?
expression? call Name
expression? Name [dot Name]* [call Name]+
expression? Name [dot Name]?
expression? [variable]? Name
expression? (int | byte | short | long | char | float | double | boolean | string) Name
expression? Name plus plus
expression? Name minus minus
expression? plus plus Name
expression? minus minus Name
expression? expression (op | compare) expression
expression? Name (op | compare) expression
expression? Name (op | compare) Name
expression? variable Name index Name
expression? string Name
expression? null
subexpression
```

Example: package

- Syntax

```
define package Name [dot Name]*
```

- Example

```
define package lyun
define package org dot elastic search dot search
define package org dot elastic search dot star
```

- Exptect Result

```
package lyun
package org.elasticsearch.search
package org.elasticsearch.*
```

Example: class

- Syntax

```
define (Annotation | public | protected | private | abstract | static | final | strictfp )*
class Name [extends Name]? [implements Name]?
```

- Example

```
define public class hello world
define private class hello world extends greeting
define protected class hello world extends greeting implements bonjour
```

- Exptect Result

```
public class HelloWorld {
}

private class HelloWorld extends Greeting {
}

protected class HelloWorld extends Greeting implements Bonjour {
}
```

Example: method

- Syntax

```
define (Annotation | public | protected | private | abstract | static | final | synchronized | native | strictfp)*
function Name [throws Exception]?d
```

- Example

```
define public function say hello
```

- Exptect Result

```
public void sayHello() {
}
```

Example: field

- Syntax

```
define (Annotation | public | protected |private | static | final | transient | volatile)*
(Name list | Name [dot Name]? [with Name [and Name]*]?) variable Name
```

- Example

```
define private int variable count
define public int list variable int list
define public node dot name variable name
define public node with integer and string variable node list
```

- Exptect Result

```
private int count;
public int[] intList;
public Node.Name name;
public Node<Integer, String> nodelist;
```

Example: type

- Syntax

```
type (Name list | Name [dot Name]? [with Name [and Name]*]?) [extends _]?
```

- Example

```
type int
type int list
type node dot name
type node with string and integer
```

- Exptect Result

```
private int count;
public int[] intList;
public Node.Name name;
public Node<Integer, String> nodelist;
```

Example: arguments

- Syntax

```
type (Name list | Name [dot Name]? [with Name [and Name]*]?) variable Name
```

- Example

```
type string variable name
type int list variable num list
type node dot name variable name
type Node dot id with string variable id
```

- Exptect Result

```
String name
int[] numList
Node.Name name
Node<Integer, String> nodelist
Node.ID<String> id
```

Example: for/while/if

- Syntax

```
define [enchanced]? for        define [do]? while
define if                      define switch
```

- Example

```
define for
define while
define if
define switch
```

- Exptect Result

```
for (; false; ) return;
while (false) return;
if (false)  return;
switch(empty) { }
```

Example: expression

- Syntax

```
expression? call Name
expression? Name [dot Name]* [call Name]+
expression? Name [dot Name]?
expression? [variable]? Name
expression? (int | byte | short | long | char | float | double | boolean | string) Name
expression? Name plus plus
expression? Name minus minus
expression? plus plus Name
expression? minus minus Name
expression? expression (op | compare) expression
expression? Name (op | compare) expression
expression? Name (op | compare) Name
expression? variable Name index Name
```

- Example

Example: expression

- Syntax

```
expression? call Name
expression? Name [dot Name]* [call Name]+
```

- Example

```
expression call sum
expression system dot out call println
```

- Exptect Result

```
sum()
System.out.println()
```

Example: expression

- Syntax

```
expression? expression (op | compare) expression
expression? Name (op | compare) expression
expression? Name (op | compare) Name
```

- Example

```
expression plus expression
expression times expression
```

- Exptect Result

```
false + false
false * false
```

A Full Example

Puppy.java

```
public class Puppy {

   private int puppyAge;

   private int puppyNum;

   public void setAge(int age) {
      puppyAge = age;
   }

   public int getAge() {
      return puppyAge;
   }

   public static void main(String[] args) {
      Puppy myPuppy = new Puppy("tommy");
      myPuppy.setAge(2);
      myPuppy.getAge();
      System.Out.println(myPuppy.puppyAge);
   }
}
```

Puppy.voiceJava generated by speech recognition line by line

```
define public class puppy
define private int variable puppy age
move next
define private int variable puppy num
move next
define public function set age
type void
type int variable age
move next
let puppy age equal variable age
move next
define public function get age
type int
move next
return variable puppy age
define public static function main
type void
type String list variable args
move next
define Puppy variable my puppy
new instance puppy
expression string tommy
move next
my puppy call set age
expression int 2
move next
my puppy call get age
move next
expression System dot Out call println
expression my puppy dot puppy age
```

Speech Recognition

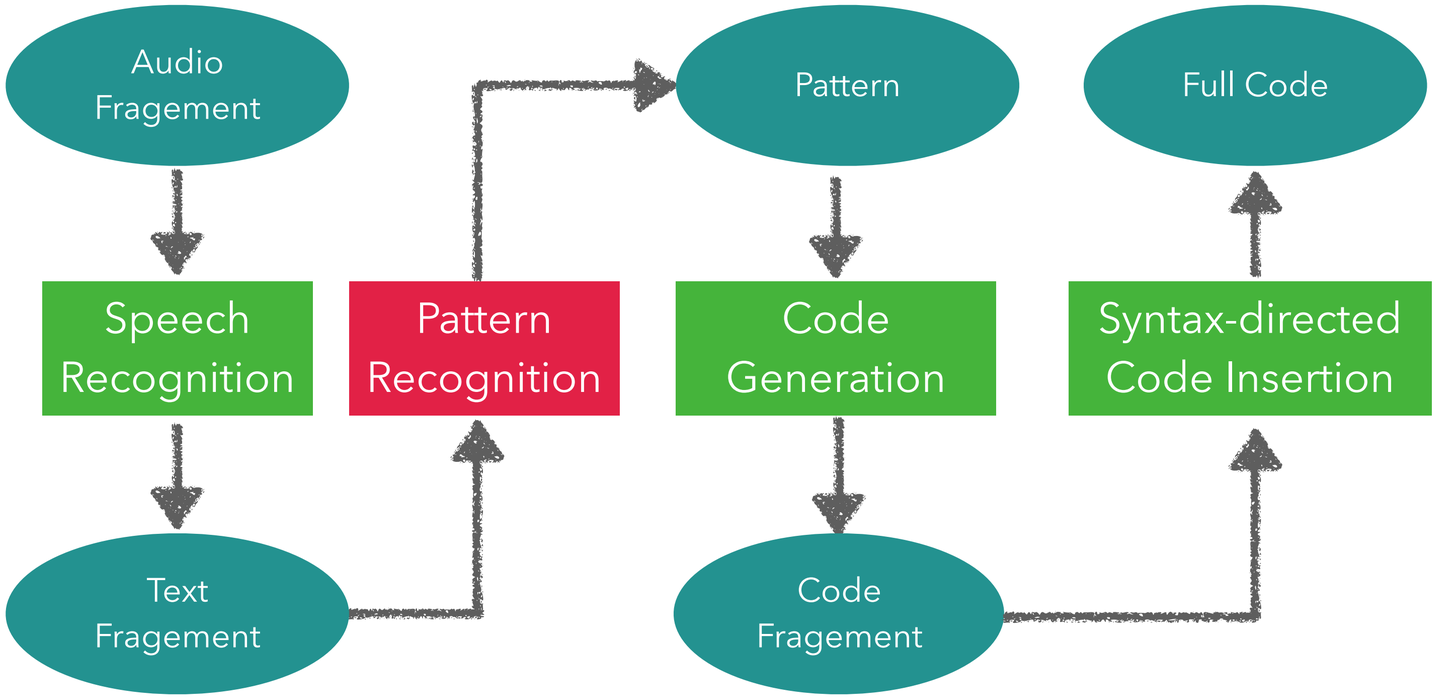
- DeepSpeech

- CMUSphinx (Vosk. 7)

DeepSpeech

- An open source Speech-To-Text engine
  - Based on Baidu's deep speech research paper
  - Implemented in Python with Tensorflow
  - Current version 0.9.3
- Use pre-trained model, accuracy is quite low
- Fine-tuning pre-trained mode with customized data set with 1000 wav video files, still quite low, because of small dataset.
- TODO: increate training data set
  - Generate dataset from opensource Java projects' source code

Architecture



Pattern Recognition

- Each line in (VoiceJava Language Syntax) correspond to one pattern

```
  define package [_]+ [dot [_]+]*

  new Pattern(
    "package", // pattern name
    "define package [_]+ [dot [_]+]*”, // pattern string representation
     new Unit[]{new Unit("define"), new Unit("package"), new Unit("plus", new Unit()),
new Unit("asterisk", new Unit("dot"), new Unit("plus", new Unit()))}); // real pattern
```

Pattern Recognition

- Convert each pattern into regex represented by NFA graph

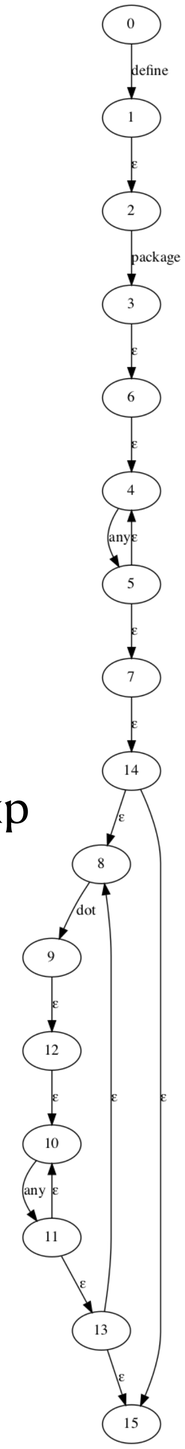
  Package pattern : define package [_]+ [dot [_]+]*

- Given a text

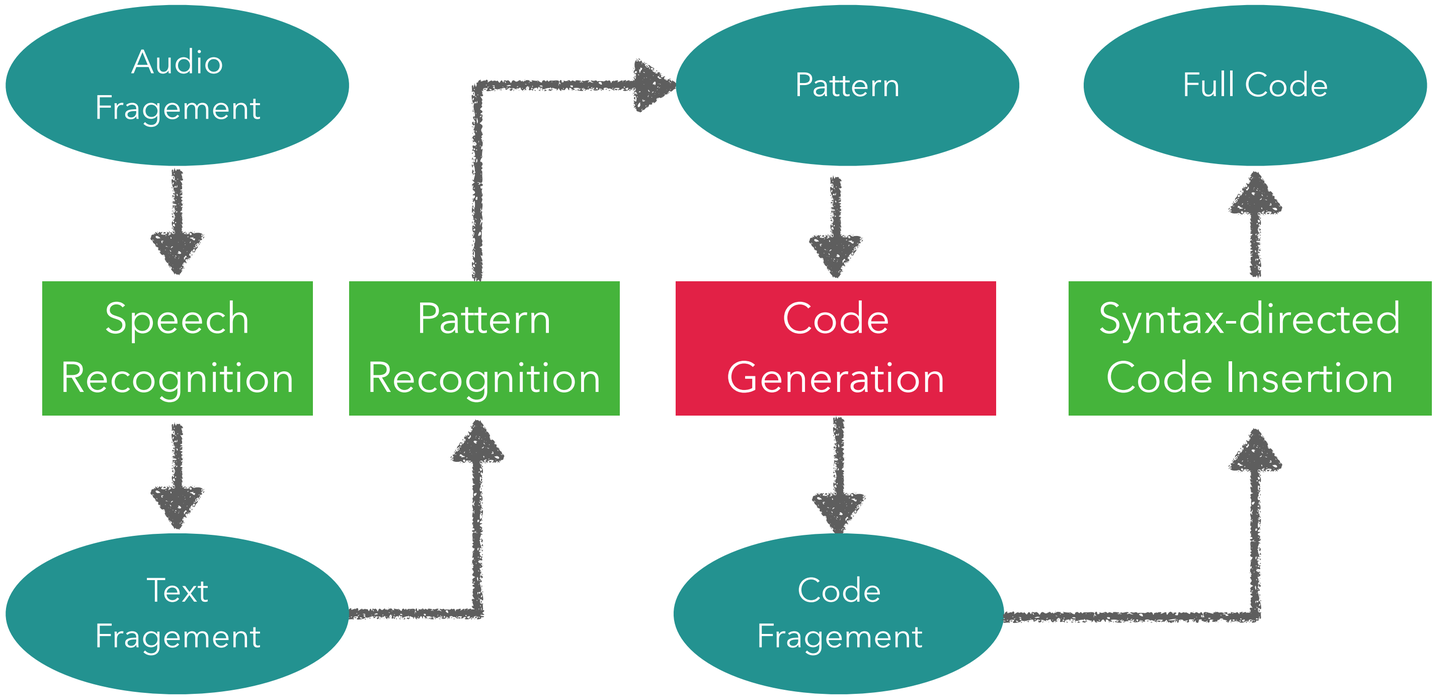
  define package org dot hello world dot star

  (Keyword define)(Keyword package)RegAnyExp (Keyword dot)RegAnyExp RegAnyExp (Keyword dot)RegAnyExp

- Optimization: Convert NFA to DFA
- Limitation: any cannot be keyword, this will restrict the expressiveness.
  - For example, package lyun.kexin.list cannot be expressed, as list is a keyword.



Architecture



Code Generation

- JavaParser 3.23.1
  - Lots of APIs for constructing Java AST

`package com.github.javaparser.ast`

Class Summary

Class	Description
ArrayCreationLevel	In `new int[1][2];` there are two ArrayCreationLevel objects, the first one contains the expression "1", the second the expression "2".
CompilationUnit	This class represents the entire compilation unit.
CompilationUnit.Storage	Information about where this compilation unit was loaded from.
DataKey<T>	A key to a piece of data associated with a Node at runtime.
ImportDeclaration	An import declaration.
Modifier	A modifier, like private, public, or volatile.
Node	Base class for all nodes of the abstract syntax tree.
Node.BreadthFirstIterator	Performs a breadth-first node traversal starting with a given node.
Node.DirectChildrenIterator	Performs a simple traversal over all nodes that have the passed node as their parent.
Node.ParentsVisitor	Iterates over the parent of the node, then the parent's parent, then the parent's parent's parent, until running out of parents.
Node.PostOrderIterator	Performs a post-order (or leaves-first) node traversal starting with a given node.
Node.PreOrderIterator	Performs a pre-order (or depth-first) node traversal starting with a given node.
NodeList<N extends Node>	A list of nodes.
PackageDeclaration	A package declaration.

https://www.javadoc.io/doc/com.github.javaparser/javaparser-core/latest/com/github/javaparser/ast/package-summary.html

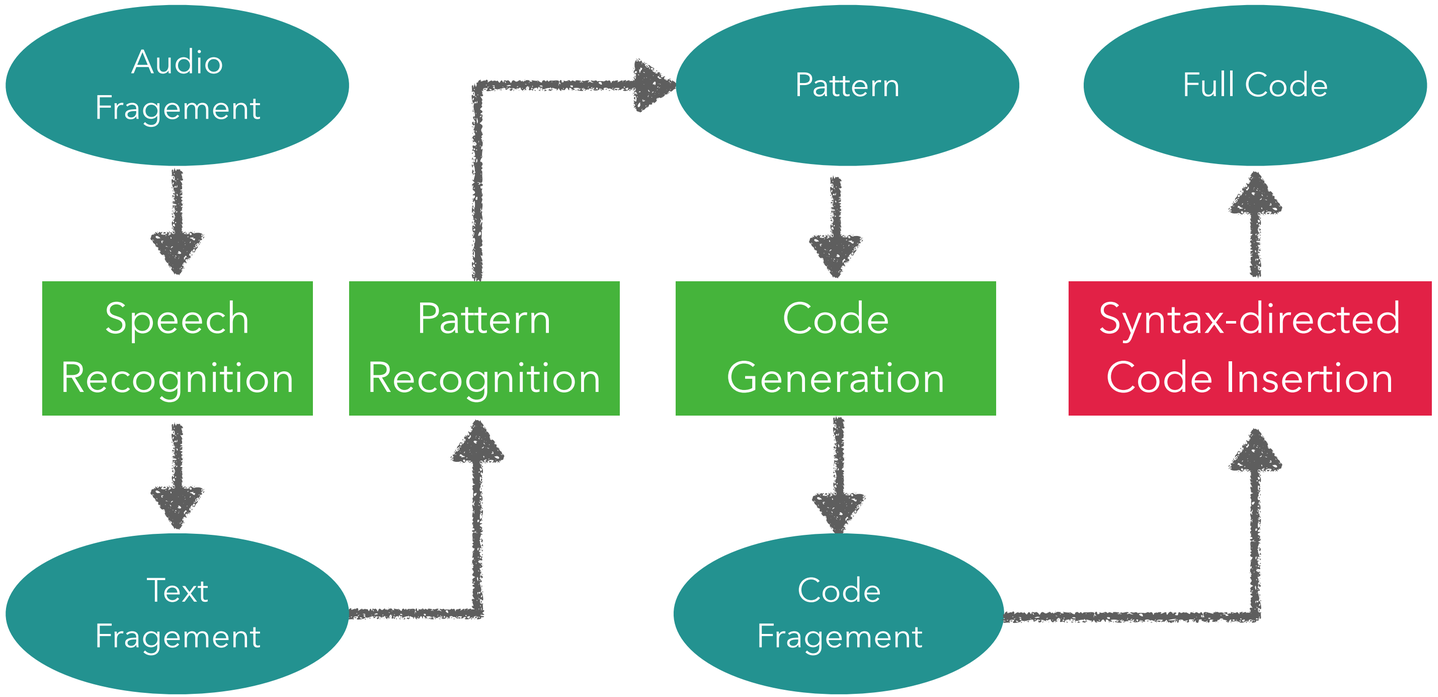
Code Generation

- For each pattern intance, generate corresponding AST

```
switch (pattern.getName()) {
  case "package":
    return new PackageAST().generate(pattern);
  case "import":
    return new ImportAST().generate(pattern);
  case "interface":
    return new InterfaceAST().generate(pattern);
  case "class":
    return new ClassAST().generate(pattern);
  case "constructor":
    return new ConstructorAST().generate(pattern);
  case "method":
    return new MethodAST().generate(pattern);
  case "arrowFunction":
    return null;
  case "field":
    return new FieldAST().generate(pattern);
  case "typeExtends":
    return new TypeExtendAST().generate(pattern);
  case "typeVariable":
    return new TypeVariableAST().generate(pattern);
  case "for":
    return new ForAST().generate(pattern);
```

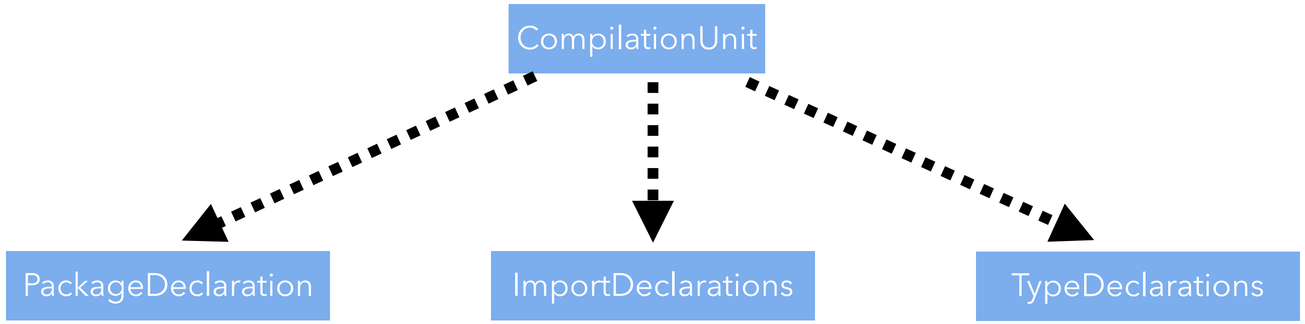
```
public class ForAST implements AST {
  public Node generate(Pattern pattern) {
    Unit[] units = pattern.getUnits();
    List<Unit> unitList = new ArrayList<Unit>(Arrays.asList(units));
    unitList.remove(0); // remove "define"

    if (unitList.get(0).getKeyword().equals("for")) {
      ForStmt forStmt = new ForStmt();
      return forStmt;
    } else {
      ForEachStmt forEachStmt = new ForEachStmt();
      return forEachStmt;
    }

  }
}
```

Architecture



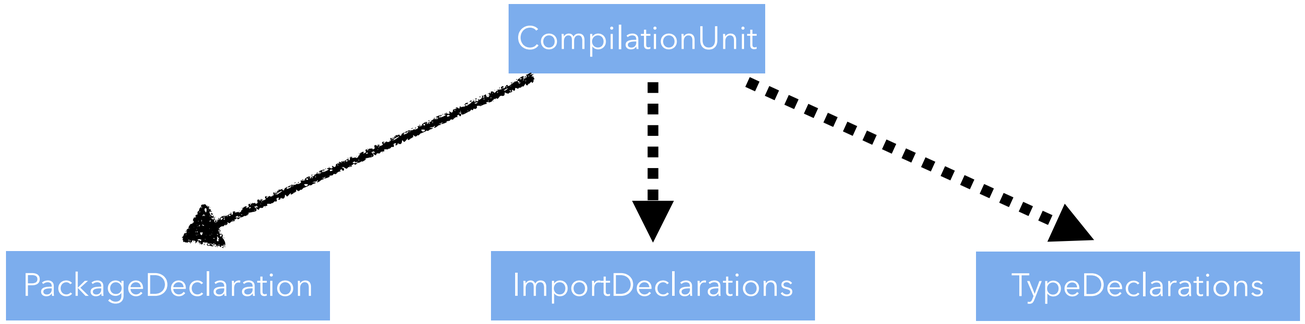
Syntax-directed Code Insertion

- A Java file is constructed in a structured manner



Syntax-directed Code Insertion

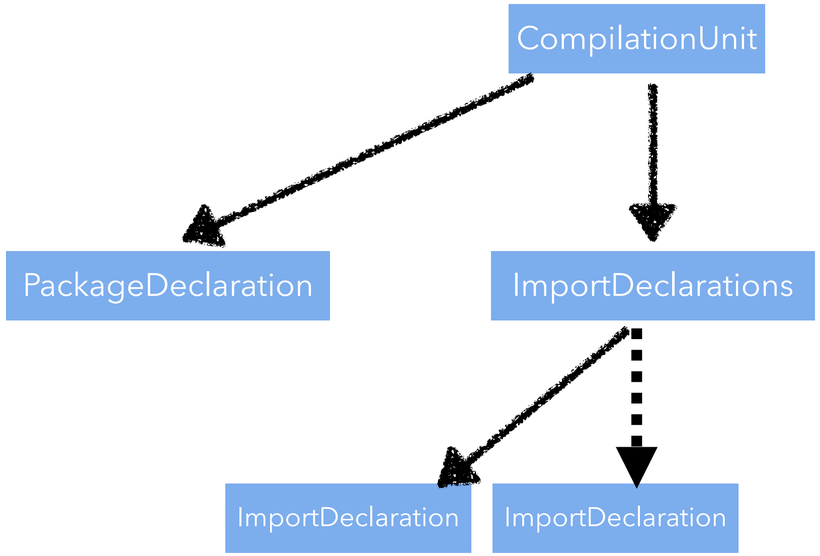
- A Java file is constructed in a structured manner




```
define package hello
```

Syntax-directed Code Insertion

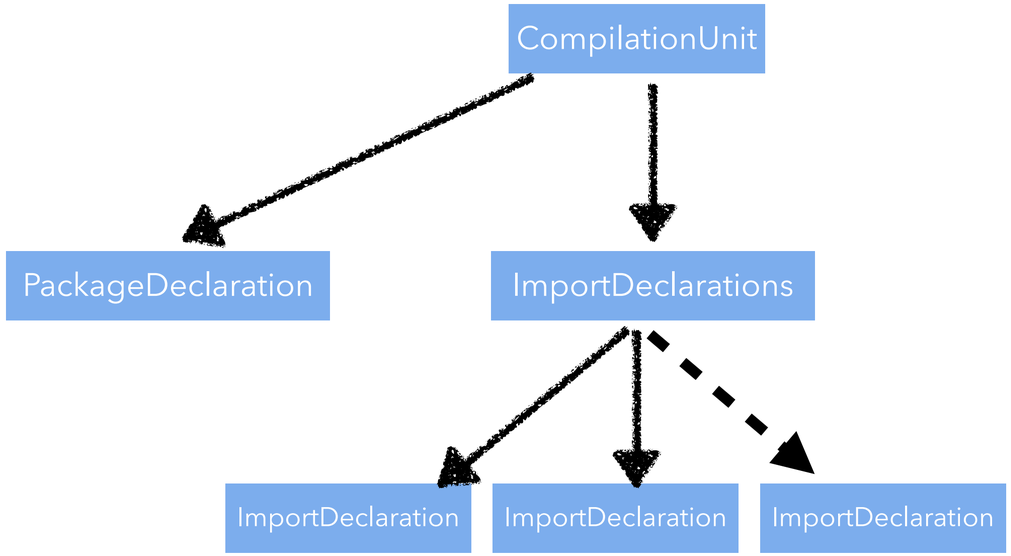
- A Java file is constructed in a structured manner




```
import java dot util dot star
```

Syntax-directed Code Insertion

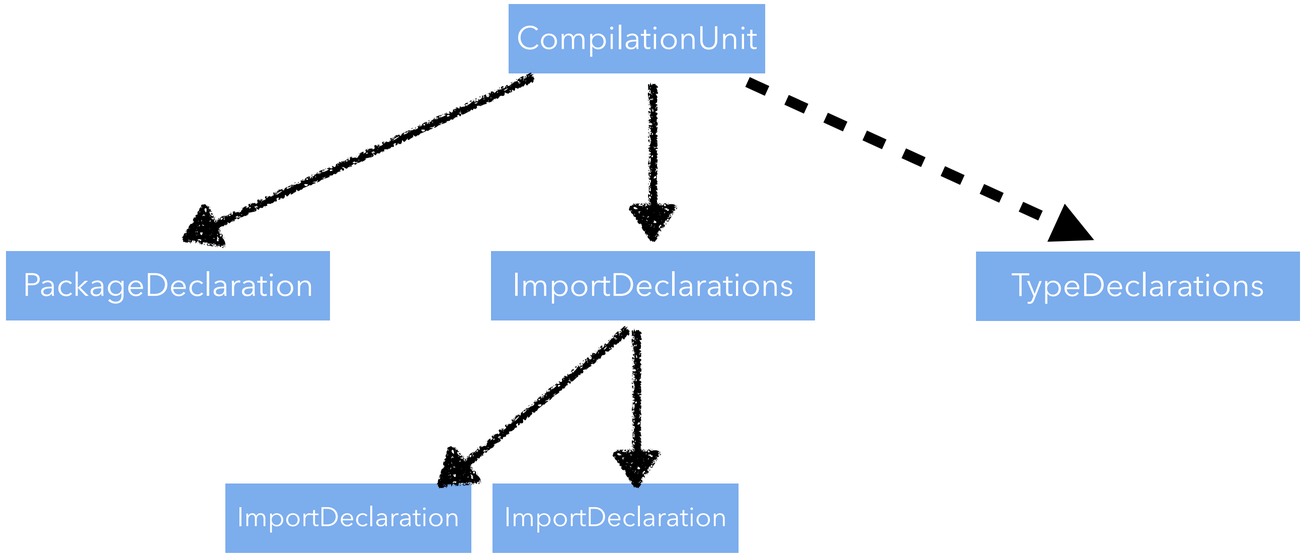
- A Java file is constructed in a structured manner




```
import java dot io
```

Syntax-directed Code Insertion

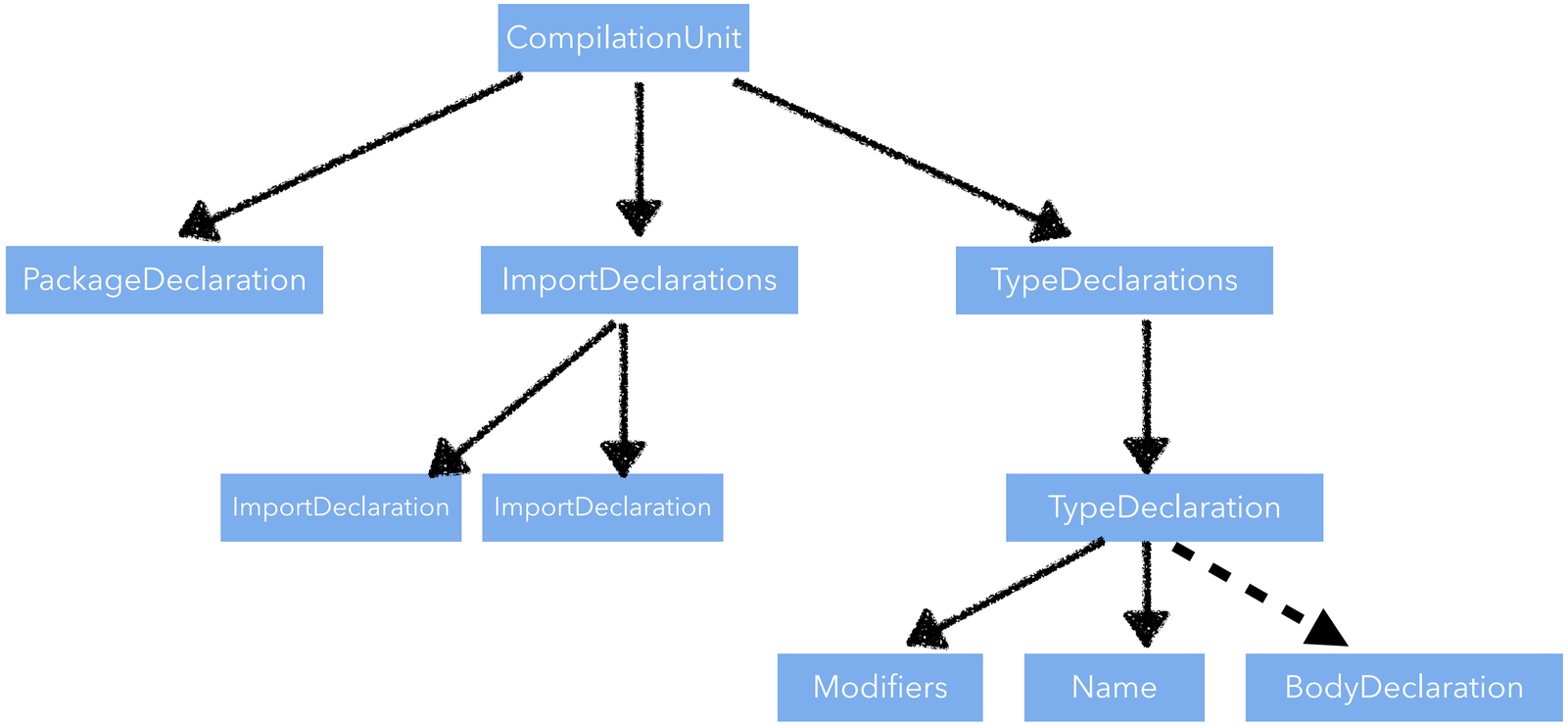
- A Java file is constructed in a structured manner




```
move next
```

Syntax-directed Code Insertion

- A Java file is constructed in a structured manner




```
define public class hello world
```

Syntax-directed Code Insertion

- A Java file is constructed in a structured manner
  - What can be inserted is restricted by syntax

    Initially, we can only say “define package ...”, “import ....”, “define class ...”, “define interface”, we cannot say “define function ....”, “define for/while/if”, etc.
  - The navigation command “move next” is guided by syntax

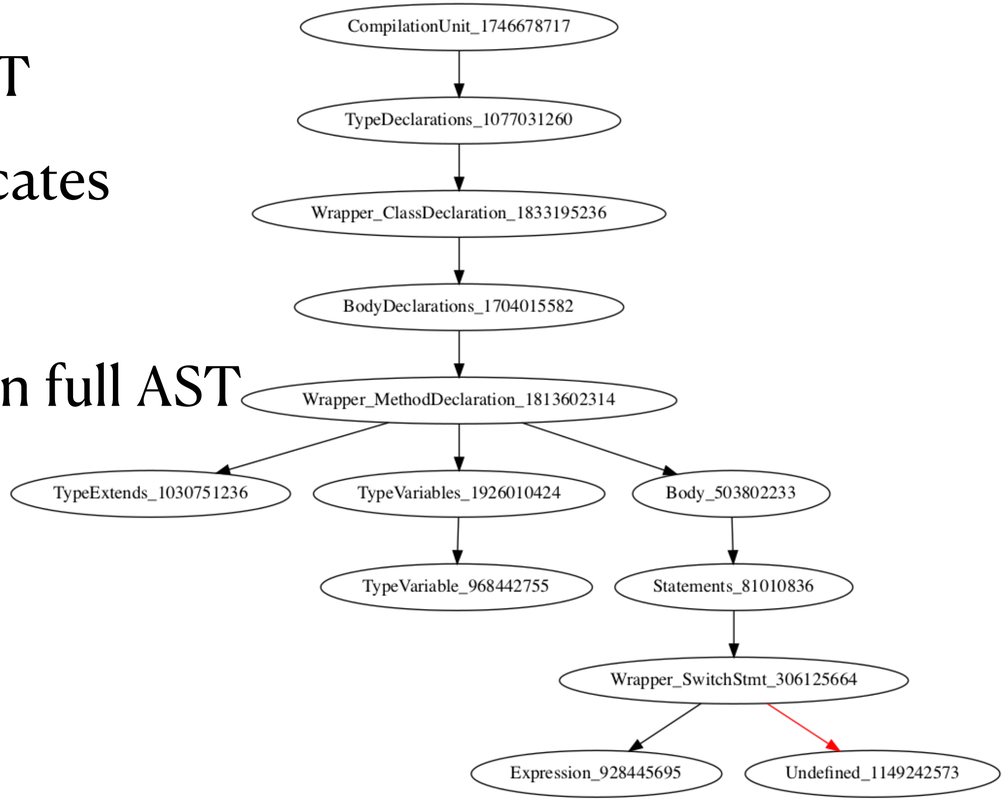
    “define public function get age, type int”, the cursor will be in between the paretheses.

    If we cay “move next”, it will jump to the body part of the function.

```
class Puppy {
    ....
    public int getAge(_) {
    }
}
```

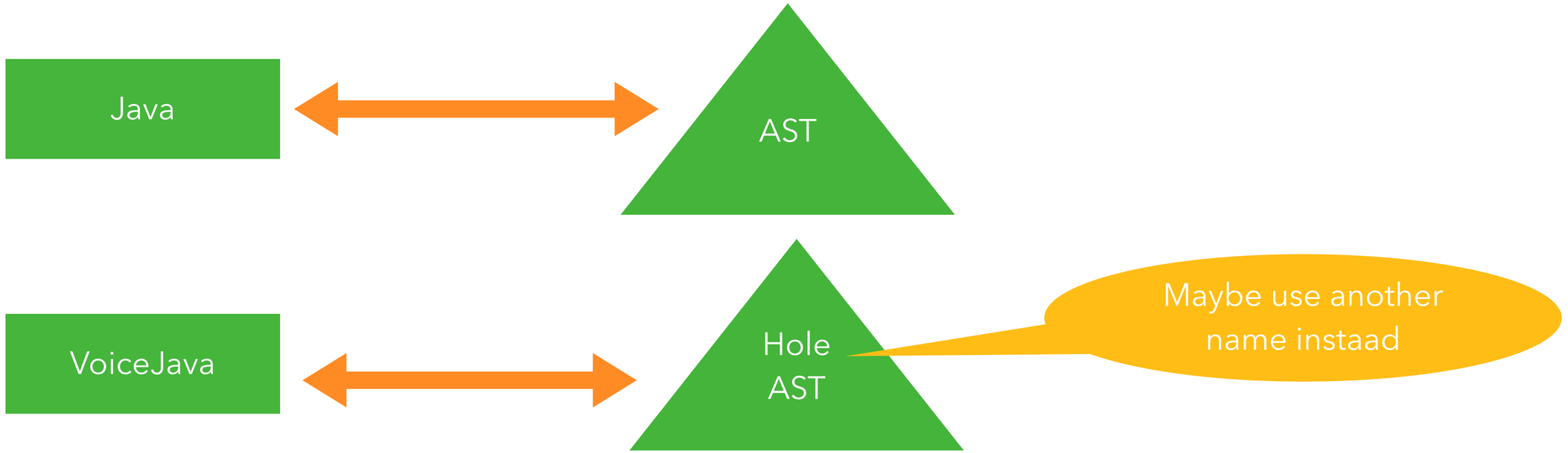
Syntax-directed Code Insertion

- Given an ast of a pattern instance (code fragement)
  - insert it into correct place in the full AST
  - update and create a hole for next insertion in HoleAST
- HoleAST is a data structure similar to full AST that indicates
  - next possible insertion point
  - path computed in HoleAST can be used to navigate on full AST



Why do we need HoleAST ?

- VoiceJava is defined at a higher level over Java code, one line of voiceJava corresponds to a small AST (normally more than one node) in Java.
- AST contains details of code, can be complex, implementing traverse over AST to find next insert hole for voiceJava is possible but difficult.



Insertion Algorithm

- 1. Case analysis on input pattern's type, use different insertion strategy according to current hole's parent.
  - A small portion of ast fragements can be only be inserted at one specific hole.

    ```
    define package hello
    parent must be CompilationUnit
    ```
  - Most of them can be inserted at many possible holes.

    ```
    define for
    ```

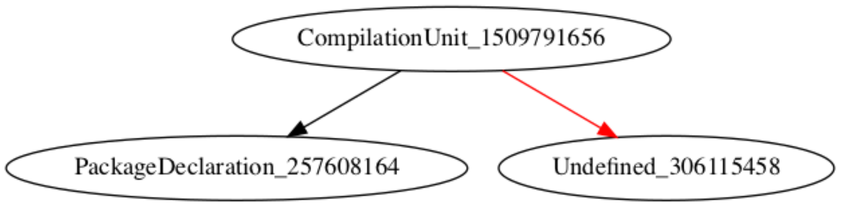
    - A ForStmt can be inside method declaration, a BlockStmt, WhileStmt, ForStmt etc.
    - Each case need to be handled.

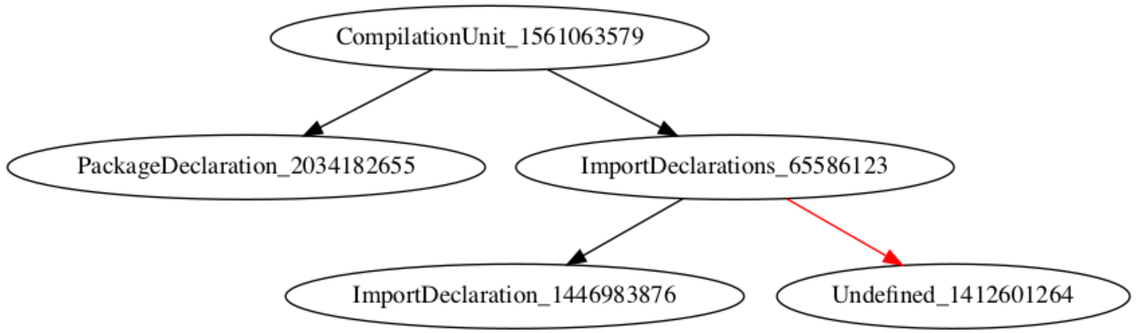
Insertion Algorithm

- 2.1 Instantiate holes on HoleAST
  - Based on insertion of AST fragement, may insert more than one holes
- 2.2 insert a new hole in next possible location in HoleAST guided by language syntax

```
define package hello
```

```
import java dot util dot star
```

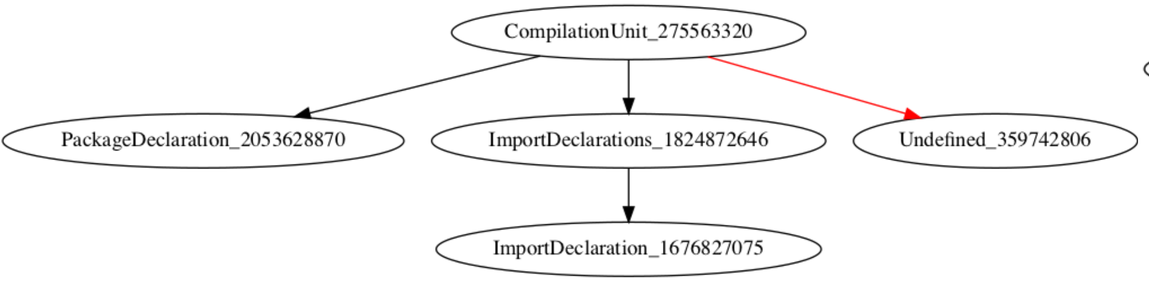




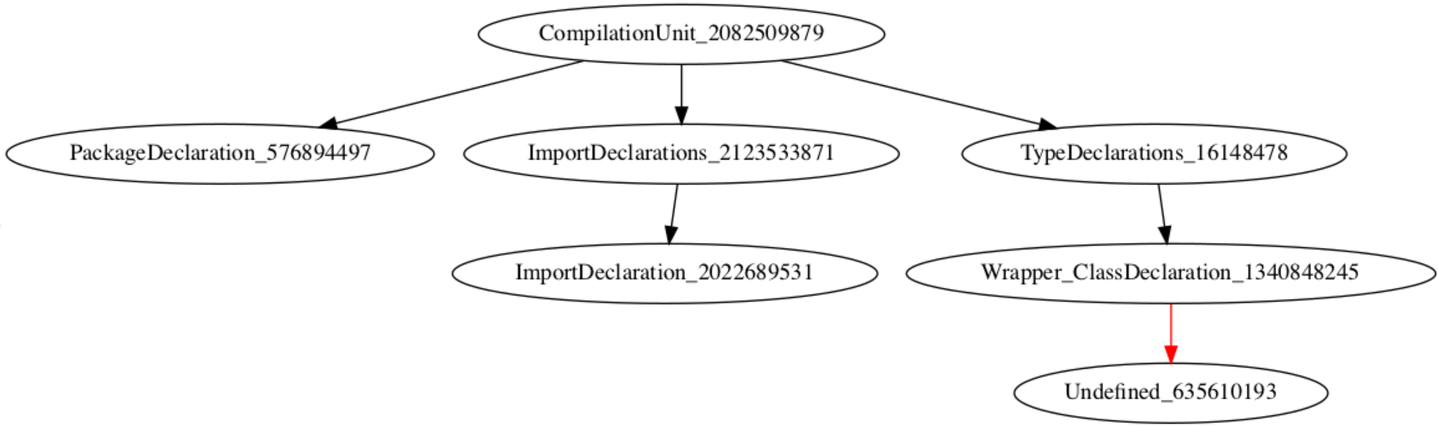
Insertion Algorithm

- 2.1 Instantiate holes on HoleAST
  - Based on insertion of AST fragement, may insert more than one holes
- 2.2 insert a new hole in next possible location in HoleAST guided by language syntax

`move next`



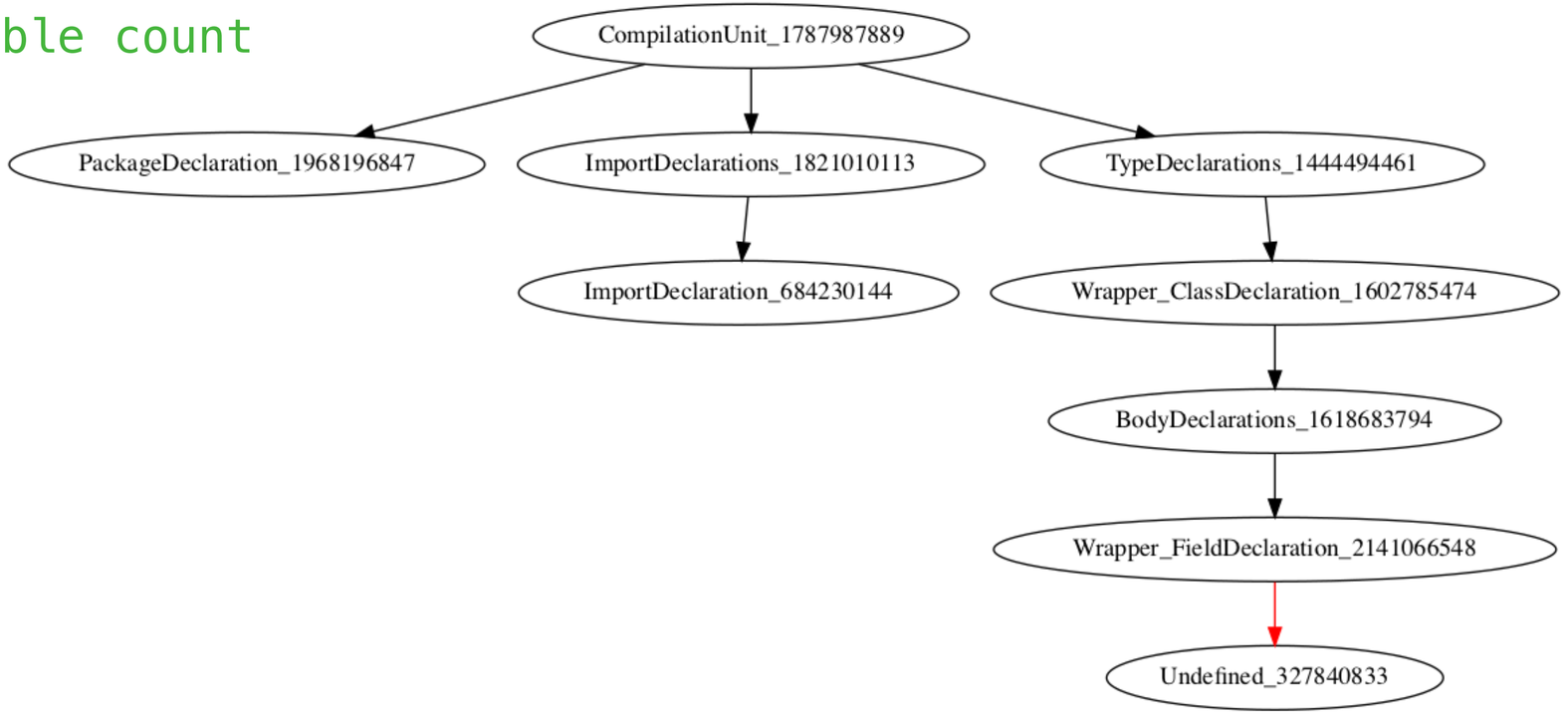
`define public class hello world`



Insertion Algorithm

- 2.1 Instantiate holes on HoleAST
  - Based on insertion of AST fragement, may insert more than one holes
- 2.2 insert a new hole in next possible location in HoleAST guided by language syntax

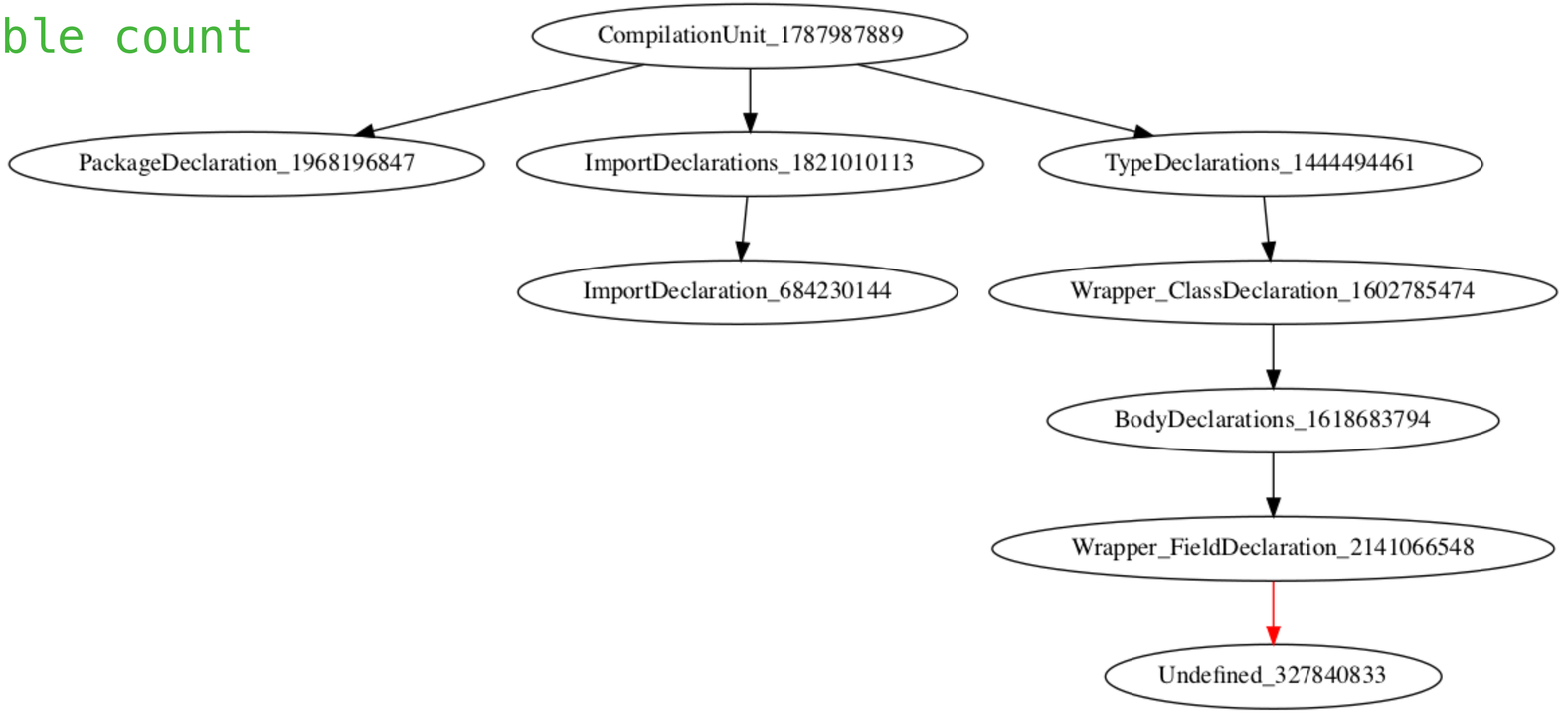
```
define private int variable count
```



Insertion Algorithm

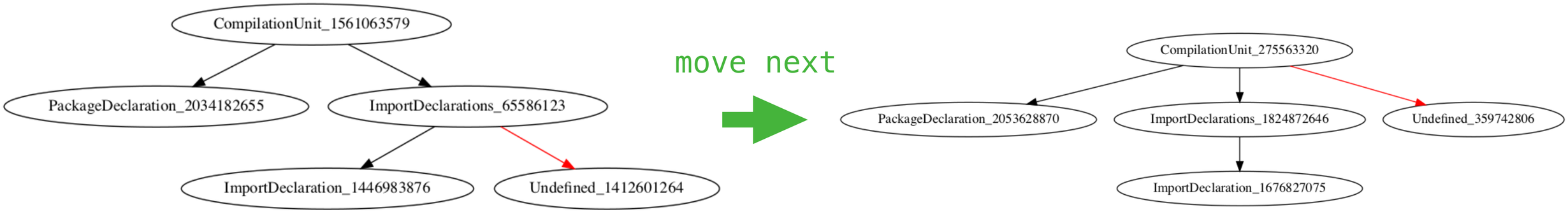
- 2.1 Instantiate holes on HoleAST
  - Based on insertion of AST fragement, may insert more than one holes
- 2.2 insert a new hole in next possible location in HoleAST guided by language syntax

```
define private int variable count
```



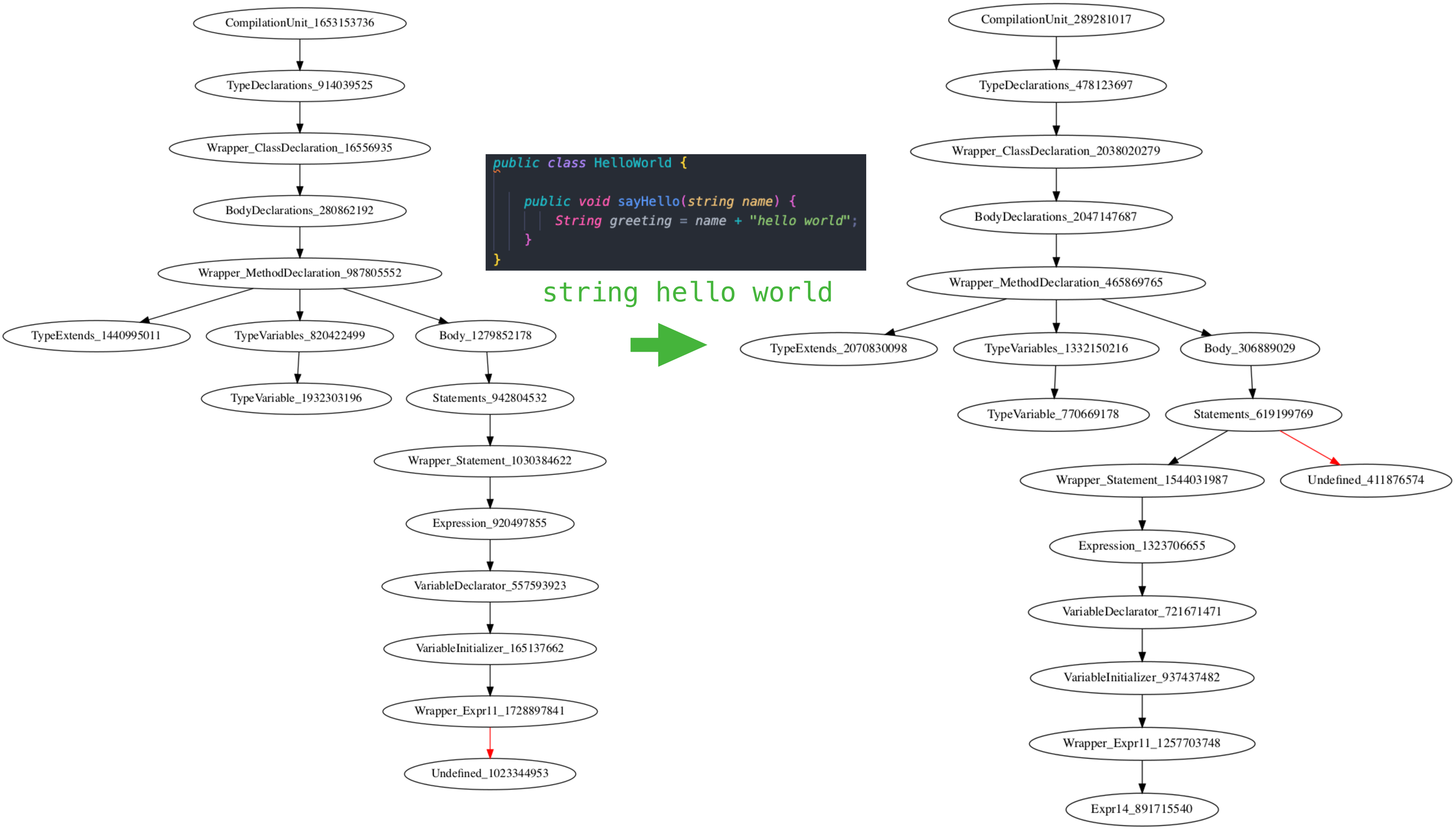
Clever MoveNext

- Simple case: create a child hole node at parent of parent node



Clever MoveNext

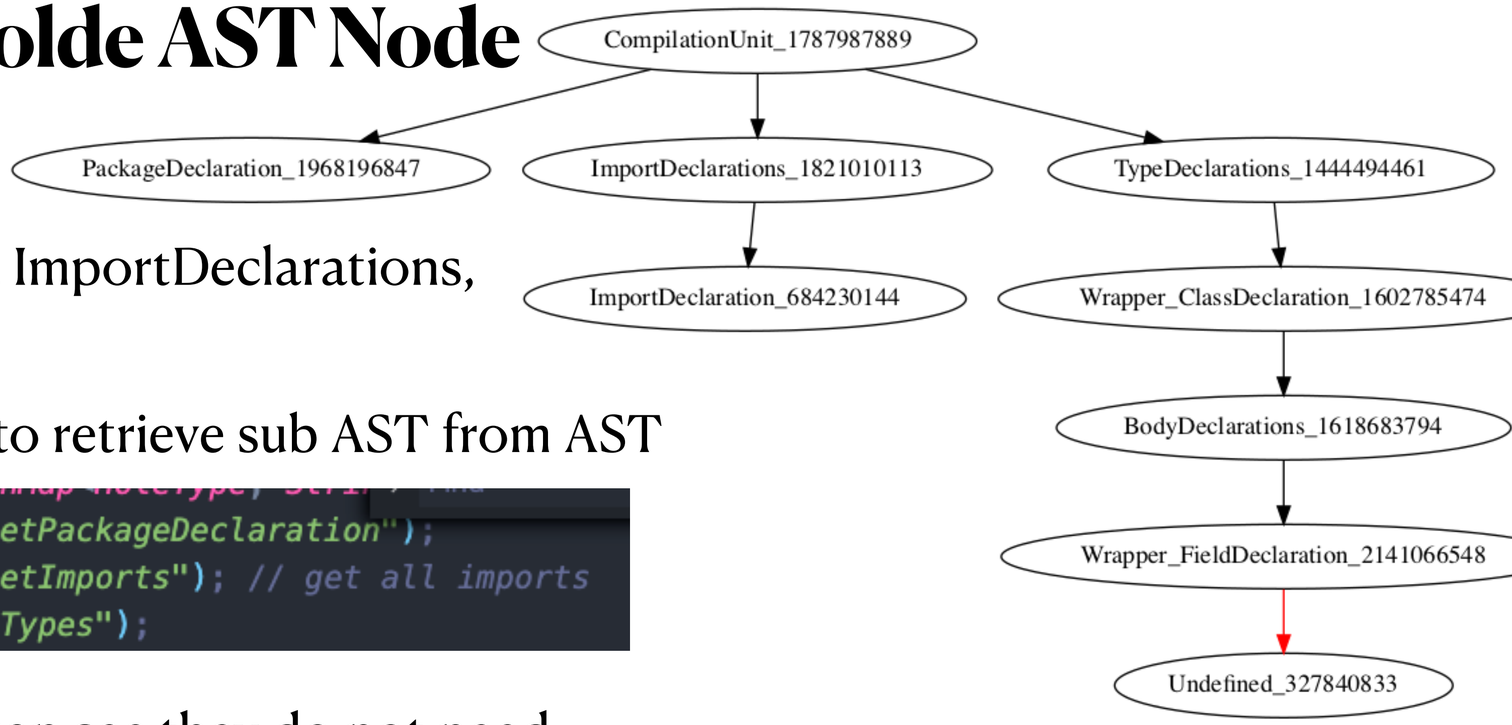
- Complex case: create a child node at parent of parent of .... node


CompilationUnit_1653153736
TypeDeclarations_914039525
Wrapper_ClassDeclaration_16556935
BodyDeclarations_280862192
Wrapper_MethodDeclaration_987805552
TypeExtends_1440995011
TypeVariables_820422499
Body_1279852178
TypeVariable_1932303196
Statements_942804532
Wrapper_Statement_1030384622
Expression_920497855
VariableDeclarator_557593923
VariableInitializer_165137662
Wrapper_Expr11_1728897841
Undefined_1023344953
public class HelloWorld {
public void sayHello(string name) {
String greeting = name + "hello world";
}
}
string hello world
CompilationUnit_289281017
TypeDeclarations_478123697
Wrapper_ClassDeclaration_2038020279
BodyDeclarations_2047147687
Wrapper_MethodDeclaration_465869765
TypeExtends_2070830098
TypeVariables_1332150216
Body_306889029
TypeVariable_770669178
Statements_619199769
Wrapper_Statement_1544031987
Undefined_411876574
Expression_1323706655
VariableDeclarator_721671471
VariableInitializer_937437482
Wrapper_Expr11_1257703748
Expr14_891715540

Clever MoveNext Algorithm

- Repeatly go back to parent node, if
  - parent node 's holeType is real type: has only one child
  - parent node 's holeType is a Wrapper: children are fullfilled according to syntax
- Maintain several sets
  - One child set: {VariableDeclarator, Statement, InnerExpr, FieldDeclaration, ...}
  - Two children set: {SwitchEntry, Let1Expr, Let2Expr, ...}
  - Three children set: { IfStmt, ...}

What is the Difference between Real Type and Wrapper in Holde AST Node



- Real type such as PackageDeclaration, ImportDeclarations, TypeDeclarations
  - Has corresponding getter function to retrieve sub AST from AST

```
map.put(HoleType.PackageDeclaration, "getPackageDeclaration");
map.put(HoleType.ImportDeclarations, "getImports"); // get all imports
map.put(HoleType.TypeDeclarations, "getTypes");
```

- Wrapper nodes does not have, or we can see they do not need.
  - But it is an import node in HoleAST, since we need the type information

Summary

- Speech Recognition part is not workable due to low accuracy
- Project contains 8500 lines of Java code
- 113 testcases are all passed which guarantee code works as expected

Improvements

- Support numbers: speech recognition engine only accepts English words
  - `int zero → int 0, float two point one two → float 2.12, int two hundreds → int 200`
- Better string support
  - e.g. add sentence command, support space, `?` , `!` marks
- Support input character
  - input a-z A-Z
- Flexible input support of name construction:
  - HelloWorld, hello_world, Helloworld
- Build a VoiceIDE for navigation/editing/execution
- Support storing and loading voiceJava project

Improvements

- Build an intelligent VoiceIDE that can automatic infer correct names based on context and knowedgebase
  - context: pre-defined variable, imported packages

```
Context: variable nameSum
expression variable name sun  ⟶  expression variable name sum
```

  - knowledgebase

```
import java dot you till      ⟶  import java dot util
```

Improvements

- Build an intelligent VoiceIDE that insertion can be guided by type information

```
System.out.println(String str)
```

```
expression system dot out dot println
```

Normally, function may have more than one arguments, we use move next command to finish the arguments insertion. According to println's type inforamtion, we know it has only one argument