1 Introduction

A convenient way to visualise a grammar is to imagine that it is a railway system, where any valid path taken by a train creates a sentence belonging to the language of the grammar. These pictorial representations of grammars are called railroad diagrams or more commonly syntax diagrams.

This note explores syntax diagrams for a subset of the Java language.

2 Types

To start with something simple, consider the syntax for specifying types in Java. In the following we can see two examples: int and String[].

```java
int iOffset;
static public void main(String[] args)
```

We can generalize what we see and and present it using the syntax diagrams in Figure 1.

![Syntax Diagrams](image)

Figure 1: Two syntax diagrams: the left is the type syntax diagram and the right is the type_specifier syntax diagram.
A terminal is something which is written literally, such as `int`, while a non-terminal is expanded into other terminals and non-terminals. In this notation, non-terminals appear in boxes and terminals are circled.

In this figure we see that types are specified with a type specifier followed by a number of optional `[]`. type specifier can be any native Java type or the name of a class or interface. `[]` specifies that the type is an array. For the purposes of this note class name and identifier name mean the same as identifier.

`String[]` is also a valid type since it is generated by choosing the path through class name and by taking the optional type path that adds `[]`. The path through class name is chosen because String is a class and the path through the `[ and then ]` is taken because in our example we have declared an array of String.

### 3 Identifiers

Java identifiers are the names given for classes, interfaces, packages, methods, and variables. Properly formed identifiers begin with a letter, underscore, or dollar sign, are case sensitive and have no maximum length. Figure 2 shows the syntax diagram corresponding to a Java identifier.

![Identifier syntax diagram](image)

Figure 2: Identifier syntax diagram.

### 4 Modifiers

Now let us look at another important part of Java: modifiers. Figure 3 shows the syntax diagram corresponding to modifiers.

![Modifier syntax diagram](image)

Figure 3: Modifier syntax diagram.
5 Variable Declaration

Now we can combine what we have learned so far (types, identifiers and modifiers) into something more interesting: variable declarations. The following example shows two variable declarations.

```java
int i, j, k;
private static HashMap _data[];
```

Figure 4 – 6 show the syntax diagram for variable declaration in Java. Try following the syntax diagrams using the example.

The `variable_declarator` non-terminal contains an identifier and an optional initialiser. Additionally, the variable declarator can specify that the variable is an array. Note that this is different than a `type` specified array as shown in Section 2. Figure 5 gives a closer look at this.

Finally, the `variable_initializer` (Figure 6) is responsible for giving the variable its initial values.

Figure 6: Syntax diagram for variable initializer.
6 Reference

Figure 7 – 9 show syntax diagrams for commonly used components of Java. Note that many of these examples are simplified for brevity.

Figure 7: Statements and expressions.
Figure 8: More statements and expressions.
for_statement

variable_declaration

expression ;; expression ;; expression ;; statement

logical_expression

true

expression

false

expression ;; expression ;; expression

class_declaration

modifier class identifier extends class_name implements interface_name

field_declaration

class_declaration

modifier
type identifier (parameter_list)

] statement_block
;
method_declaration

modifier
type identifier (parameter_list)

[ ] statement_block
;
parameter

type identifier

parameter_list

parameter

Figure 9: A couple more statements and expressions. Class declarations, and methods.