https://www.bluej.org/objects-first/styleguide.html

Program Style Guide

Version 2.0

1. Naming

1.1 Use meaningful names.

Use descriptive names for all identifiers (names of classes, variables and methods). Avoid ambiguity. Avoid abbreviations. Simple mutator methods should be named set*Something*(...). Simple accessor methods should be named get*Something*(...). Accessor methods with boolean return values are often called is*Something*(...), for example, isEmpty().

1.2 Class names start with a capital letter.

1.3 Class names are singular nouns.

1.4 Method and variable names start with lowercase letters.

All three - class, method and variable names - use capital letters in the middle to increase readability of compound identifiers, e.g. numberOfItems.

1.5 Constants are written in UPPERCASE.

Constants occasionally use underscores to indicate compound identifiers: MAXIMUM_SIZE

2. Layout

2.1 One level of indentation is four spaces.

2.2 All statements within a block are indented one level.

2.3 Braces for classes and methods are alone on one line.

The braces for class and method blocks are on separate lines and are at the same indentation level, for example:

```
public int getAge()
{
    statements
}
```

2.4 For all other blocks, braces open at the end of a line.

All other blocks open with braces at the end of the line that contains the keyword defining the block. The closing brace is on a separate line, aligned under the keyword that defines the block. For example:

```
while(condition) {
    statements
}

if(condition) {
    statements
}
else {
    statements
}
```

2.5 Always use braces in control structures.

Braces are used in if-statements and loops even if the body is only a single statement.

2.6 Use a space before the opening brace of a control structure's block.

2.7 Use a space around operators.

2.8 Use a blank line between methods (and constructors).

Use blank lines to separate logical blocks of code. This means at least between methods, but also between logical parts within a method.

3. Documentation

3.1 Every class has a class comment at the top.

The class comment contains at least
- a general description of the class
- the author's name(s)
- a version number

Every person who has contributed to the class has to be named as an author or has to be otherwise appropriately credited.

A version number can be a simple number, a date, or other formats. The important thing is that a reader must be able to recognise if two version are not the same, and be able to determine which one is newer.

3.2 Every method has a method comment.

3.3 Comments are Javadoc-readable.

Class and method comments must be recognised by Javadoc. In other words: they should start with the comment symbol /**.

3.4 Code comments (only) where necessary.

Comments in the code should be included where the code is not obvious or difficult to understand (while preference should be given to make the code obvious or easy to understand where possible), and where it helps understanding of a method. Do not comment obvious statements - assume your reader understands Java!

4. Language use restrictions

4.1 Order of declarations: fields, constructors, methods.

The elements of a class definition appear (if present) in the following order: package statement; import statements; class comment; class header; field definitions; constructors; methods.

4.2 Fields may not be public (except for final fields).

4.3 Always use an access modifier.

Specify all fields and methods as either private, public, or protected. Never use default (package private) access.

4.4 Import classes separately.

Import statements explicitly naming every class are preferred over importing whole packages. E.g.

```
import java.util.ArrayList;
import java.util.HashSet;
```

is better than

```
import java.util.*;
```

4.5 Always include a constructor (even if the body is empty).

4.6 Always include superclass constructor call.

In constructors of subclasses, do not rely on automatic insertion of a superclass call. Include the super(...) call explicitly, even if it would work without it.

4.7 Initialise all fields in the constructor.

5. Code idioms

5.1 Use iterators with collections.

To iterate over a collection, use a for-each loop. When the collection must be changed during iteration use an Iterator, not an integer index.

---

https://www.bluej.org/objects-first/styleguide.html