שאלה 1

# Coding Style Guidelines

## Naming Conventions

### Class Names

Classes are written in UpperCamelCase.

### Method Names

Methods are written in lowerCamelCase.

* Setter methods are written as setField.
* Getter methods are written as getField for all field types except Boolean and boolean, which are written as isField.

### Non-Constant Field Names

Non-constant fields are written in lowerCamelCase.

### Constant Field Names

Constant fields are written in UPPER\_SNAKE\_CASE.

### Local Variable Names

Local variables are written in lowerCamelCase.

### Parameter Names

Parameters are written in lowerCamelCase.

### Enum Type Names

Enum types are written in UPPER\_SNAKE\_CASE, e.g.

enum Suit { HEARTS, SPADES, CLUBS, DIAMONDS; }

## Formatting

### Column Limit

Java code has a column limit of 120 characters.

### Indentation

Non-empty blocks are indented by 4 spaces.

### Horizontal Whitespace

A single space appears:

* Between reserved words (if, for, while, try, catch, etc.) and an open parenthesis on the same line, e.g.
  + if (true) , catch (Exception e)
* Between non-space characters and a close curly brace (}), e.g.
  + } else , } finally
* Between non-space character and an open curly brace ({), e.g.
  + class Foo {} , try {} , while (true) {}
* After commas, e.g.
  + Math.max(1, 2); , new int[] {8, 9, 11};
* Before and after a colon (:) and a question mark (?), e.g.
  + int n = true ? 1 : 0;
* Before and after operators, e.g.
  + while (a && b); , a = a + b; , x \*= z;

### Vertical Whitespace

At least a single new line appears:

* Between consecutive members or initializers of a class: fields, constructors, methods, nested classes, static initializers, and instance initializers.
  + A blank line between two consecutive fields (having no other code between them) is optional.

### Braces

Braces follow a variation of the [K&R Style](https://en.wikipedia.org/wiki/Indentation_style#K.26R), in which the opening curly brace is on the same line also for class or method declarations.

## Documentation

### Javadoc

Javadoc blocks follow the following formatting:

/\*\*

\* Multiple lines of Javadoc text are written here,

\* wrapped normally...

\*/

public int method(String p1) { ... }

### Other Comments

Comments that help describe the logic of lines of code are written in separate lines from the code itself, e.g.

* Allowed:
  + // Calculating the meaning of life

int lifeMeaning = 42;

* + /\* Calculating the meaning of life \*/

int lifeMeaning = 42;

* + /\* Calculating the meaning

of life \*/

int lifeMeaning = 42;

* Prohibited:
  + int lifeMeaning = 42; // Calculating the meaning of life

Comments should not explain trivial operations, e.g.

* // Incrementing x by 1
* x++;

שאלה 2

פלט המחלקה Flesch עבור הקובץ The little engine that could\_djvu.txt:

Flesch reading ease grade of "D:\Programming\Java\OOP\_046271\hw0\Flesch\_inputs\The little engine that could\_djvu.txt" is 91.94

פלט המחלקה Flesch עבור הקובץ grapeswrath00cablgoog\_djvu.txt:

Flesch reading ease grade of "D:\Programming\Java\OOP\_046271\hw0\Flesch\_inputs\grapeswrath00cablgoog\_djvu.txt" is 83.86

שאלה 3