Author(s): Lahti, Lauri

Title: Supplement to Lauri Lahti s conference article "ConceptMapWiki a collaborative framework for agglomerating pedagogical knowledge"

Year: 2015

Version: Pre-print

Supplement to Lauri Lahti's conference article "ConceptMapWiki – a collaborative framework for agglomerating pedagogical knowledge"

Lauri Lahti, 20110610 (updated in 20150317)
Department of Computer Science, Aalto University School of Science, Finland

Conference article: Lahti, L. (2011a). ConceptMapWiki – a collaborative framework for agglomerating pedagogical knowledge. Proc. 11th IEEE International Conference on Advanced Learning Technologies (ICALT 2011), 6–8 July 2011, Athens, Georgia, USA (eds. Aedo, I. et al.), 163-165. Online ISBN 978-0-7695-4346-8 and Print ISBN 978-1-61284-209-7. http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=5992312

This supplement has been available online at:
http://www.cs.hut.fi/u/llahti/publ/lahti_2011_data.pdf

Empirical experiment

(corresponding to analysis in Subchapter 8.2 of Lauri Lahti's doctoral dissertation "Computer-assisted learning based on cumulative vocabularies, conceptual networks and Wikipedia linkage" (Lahti 2015a) and Lahti (2015b, Appendix M))

We have carried out *empirical experiment* to evaluate educational gain of the proposed method. We report here corrected results that somewhat differ from those results reported in publication [P5], our analysis is based on material gathered from 147 university students of *introductory Java programming course* who we asked to draw with our method *concept maps* representing their knowledge about learning topic "programming". Among these 147 students there were 124 men and 23 women and average of age of students was 20.86 years (median 20 years). Although we present here the results in English, the experiment was carried out in Finnish but we present the results here in English. User interface of an prototype tool used in the experiment is shown in Appendix M.

After eliminating unclear responses and transforming all concepts to non-conjugated base forms, and considering only those concepts and relationships mentioned by at least two students, we identified 167 unique concepts and 167 unique conceptual relationships between them. A full listing of these unique concepts and unique relationships supplied with occurrences in concept maps is shown in Appendix M. Five most frequent concepts, number of students who mentioned the concepts shown in parenthesis, were programming (90), object (62), method (60), java (57) and class (49). Five most frequent relationships, number of students who mentioned the relationships shown in parenthesis, were object -> method (29), class -> object (27), programming -> programming language (27), programming language -> java (18) and programming -> language (17).

Table 8.1 shows how 147 students gradually introduced relationships to concept maps about programming. It appears that the most popular conceptual relationship that the students added as their first conceptual relationship to concept maps was programming -> language (mentioned by 11 students). The most popular conceptual relationship to be added as their second conceptual relationship was programming -> programming language (mentioned by 7 students). The most popular conceptual relationship to be added as their third conceptual relationship was object -> method (mentioned by 6 students).

To analyze pedagogical value of the method we compared evolution of drawn concept maps to an extensive *narrative from 28 lectures* of introductory Java programming course (Sahami 2010). We computed that this lecture narrative contained 6291 unique concepts that had altogether 101599 occurrences. We compared drawn concept maps to co-occurring words in 18142 unique sentences

of the lecture narrative. The high-ranking concepts and high-ranking conceptual relationships in drawn concept maps well matched with the high-ranking concepts and highest-ranking co-occurring concept pairs in the pedagogical narrative. For example, ten *highest-ranking concepts* of concept maps and ten highest-ranking concepts of narrative had overlap of about 65 percent, and ten *highest-ranking relationships* of concept maps and ten highest-ranking co-occurring concept pairs of narrative had overlap of about 50 percent. Motivated by additional analysis, we introduce here corrected results about experimentally gathered data and thus results reported here somewhat differ from those results originally presented in publication [P5].

Table 8.1. Listings showing how 147 students gradually introduced conceptual relationships to concept maps that they drew about programming, these three listings show the most occurring relationships in first, second and third relationship each student has added (n=147). Only those relationships are shown that were mentioned by at least two students.

The most actively introduced conceptual relationships when the student added the first relationship to her concept map		The most actively introduced conceptual relationships when the student added the second relationship to her concept map		The most actively introduced conceptual relationships when the student added the third relationship to her concept map	
Conceptual relationship	*Number of students mentioning this relationship*	*Conceptual relationship*	*Number of students mentioning this relationship*	*Conceptual relationship*	*Number of students mentioning this relationship*
programming -> language	11	programming -> programming language	7	object -> method	6
class -> object	8	class -> object	6	language -> python	4
programming -> programming language	8	programming language -> java	5	programming language -> c	4
programming -> object	4	language -> java	4	class -> method	3
programming -> object-oriented programming	3	language -> c	3	class -> object	3
programming -> program	3	object -> method	3	language -> java	3
object -> method	2	variable -> object	3	programming language -> java	3
program -> class	2	class -> method	2	method -> object	2
programming -> java	2	code -> program	2	object -> list	2
programming -> python	2	java -> object	2	object -> variable	2
programming -> tool	2	object -> variable	2	programming -> c++	2
programming -> variable	2	package -> class	2	programming -> java	2
variable -> object	2	programmer -> programming	2	programming -> language	2
		programming -> logic	2	programming -> program	2
		programming -> object	2		

From Table 8.2 it can be seen that among ten highest-ranking concepts for lecture narrative concepts (when counting concepts thing and things as one) there seem to be six concepts specific for describing learning topic of programming including: class, program, method, object, value and array. On the other hand ten highest-ranking concepts for concept maps about programming, if we first exclude language-related vocabulary and concepts directly referring to concept of programming itself, include five concepts: object, method, class, program and variable. Thus when comparing these two sets of concepts (six concepts and five concepts) four of them are shared (i.e. class, method, object and program) thus resulting in matching overlap of about 65 percent. We think that this result indicates that the proposed relatively self-guided method can assist learners to generate and process knowledge in a pedagogically rewarding way, even challenging the knowledge evolution process suggested by a professional teacher.

Table 8.2. Highest-ranking concepts in lecture narrative of introductory Java programming course (having at least 262 occurrences) available from Sahami (Sahami 2010) and concept maps about programming (having at least 8 occurrences) drawn by students (n=147). Conjugated forms of concepts of concept maps were transformed into base form but concepts of lecture narrative were kept in initial conjugated forms since reliable automated transformation seemed challenging and manual transformation laborious.

Highest-ranking concepts in lecture narrative of introductory Java programming course		Highest-ranking concepts in concept maps about programming	
Concept	*Occurrences*	*Concept*	*Occurrences*
thing	1007	programming	90
class	902	object	62
program	836	method	60
time	757	java	57
things	742	class	49
name	640	program	47
way	613	programming language	44
method	604	variable	41
object	585	python	31
value	558	c	29
array	511	programmer	25
string	485	language	24
sort	478	object-oriented programming	22
set	463	computer	21
number	435	user	21
stuff	395	compiler	20
people	387	c++	19
means	368	code	17
run	367	user interface	16
line	350	loop	13
call	349	debugger	12
use	343	eclipse	12
doing	342	problem	11
computer	342	algorithm	9
variable	338	conditional sentence	9
file	330	int	9
take	327	parameter	9
show	327	program code	9
java	325	ready program	9
point	313	starting method	9
code	291	tool	9
example	283	library	8
list	263	machine language	8
type	262	testing	8
world	258		
start	255		
bit	254		

When analyzing the *highest-ranking conceptual relationships* in concept maps about programming shown in Table 8.3 (based on Appendix M), and first excluding language-related vocabulary and concepts directly referring to concept of programming itself, we ended up observing those nine highest-ranking relationships marked with an asterisk (*) and one of those six relationships marked with a double asterisk (**) since these six relationships share the same ranking. In these ten relationships 5 concepts become mentioned anyway (object (6 occurrences), class (5 or 6 occurrences), method (3 or 4 occurrences), variable (3 or 4 occurrences), program (1 or 2 occurrences)) and additionally possibly one of three concepts become mentioned (code (0 or 1 occurrences), package (0 or 1 occurrences) and programmer (0 or 1 occurrences)).

Table 8.3. Comparison concerning conceptual relationships of concept maps about programming and co-occurring concepts of lecture narrative of introductory Java programming course available from Sahami (Sahami 2010). As explained in main text of Subchapter 8.2 in our further analysis we ended up observing those nine highest-ranking relationships marked with an asterisk (*) and one of those six relationships marked with a double asterisk (**) since these six relationships share the same ranking.

The highest-ranking conceptual relationships in concept	How many times each of 50 highest-ranking concepts of lecture

maps about programming drawn by students (n=147) (only those relationships occurring at least 5 times shown here, more shown in Appendix M)		narrative of introductory Java programming course co-occurs with any other word(s) of 50 highest-ranking concepts in a same phrase, and how many times each of these 50 concepts occurs irrespective of co-occurrences		
Conceptual relationship	*Occurrences*	*Conceptual relationship*	*Co-occurrences*	*Occurrences (ranking)*
* object -> method	29	thing	344	1007 (1)
* class -> object	27	things	240	742 (5)
programming -> programming language	27	way	230	613 (7)
programming language -> java	18	name	214	640 (6)
programming -> language	17	sort	214	478 (13)
* class -> method	14	method	203	604 (8)
java -> object	14	time	198	757 (4)
programming -> program	14	class	196	902 (2)
* object -> variable	12	program	194	836 (3)
language -> java	11	set	180	463 (14)
language -> c	10	object	165	585 (9)
* program -> class	10	show	165	327 (27.5s)
* object -> class	9	means	161	368 (18)
* variable -> object	9	call	160	349 (21)
java -> object-oriented programming	8	doing	160	342 (23.5s)
language -> python	8	value	158	558 (10)
programming language -> c	8	array	155	511 (11)
programming -> object	8	use	155	343 (22)
programming -> object-oriented programming	8	run	144	367 (19)
programming language -> python	7	stuff	130	395 (16)
* class -> variable	6	number	127	435 (15)
* method -> object	6	string	125	485 (12)
object-oriented programming -> java	6	take	125	327 (27.5s)
programming -> computer	6	inside	120	235 (43)
programming -> java	6	people	118	387 (17)
programming -> tool	6	bunch	115	222 (48)
c -> c++	5	variable	111	338 (25)
** code -> program	5	computer	107	342 (23.5s)
java -> class	5	type	105	262 (34)
** method -> class	5	bit	105	254 (37)
** method -> variable	5	start	103	255 (36)
** package -> class	5	code	94	291 (31)
** programmer -> program	5	line	92	350 (20)
programmer -> programming	5	java	92	325 (29)
programmer -> programming language	5	list	81	263 (33)
programming -> programmer	5	example	79	283 (32)
programming -> user interface	5	text	79	219 (50)
programming -> variable	5	point	77	313 (30)
** variable -> method	5	size	72	229 (45.5s)
		file	69	330 (26)
		integer	66	232 (44)
		move	63	246 (38)
		case	63	241 (41s)
		zero	61	245 (39)
		world	58	258 (35)
		box	55	220 (49)
		album	52	226 (47)
		times	51	229 (45.5s)
		loop	46	241 (41s)
		screen	37	241 (41s)

From lecture narrative we identified how many times each of 50 highest-ranking concepts co-occurs with any other concept(s) of 50 highest-ranking concepts in a same phrase. The number of these co-occurrences is shown in Table 8.3 for each of 50 highest-ranking concepts. In this listing it can be seen that among ten highest-ranking concepts for lecture narrative concepts (when counting concepts thing and things as one) there seems to be six concepts specific for describing learning topic of programming including: sort, method, class, program, set and object. Thus when comparing these two sets of concepts (5 or 6 actively used concepts in relationships of concept maps about programming and 6 actively used concepts in phrase-based co-occurrences of lecture narrative) four of them are shared (i.e. class, method, object and program) thus resulting in matching overlap of about 65 percent.

We analyzed the drawn concept maps in respect to the *learner's self-evaluation* about three characteristics based on responses given by students after drawing concept map in experiment: amount of earlier programming experience, difficulty of learning programming and the complexity of the concept map she had drawn, measured with five-point Likert scale (response alternatives are listed in Appendix M). Based on this analysis Table 8.4 shows distribution of rankings of concepts of concept maps about programming in respect to responses given by students and Table 8.5 shows distribution of rankings of conceptual relationships of concept maps about programming in respect to responses given by students. Here we took into account only such concepts and conceptual relationships that were mentioned by at least two students. We observed surprisingly coherent concept maps to be drawn irrespective of the responses given in self-evaluation. For example, for ten highest-ranking concepts as well as conceptual relationships there was overlap of about 50 percent between more experienced and less experienced learners, between learners considering learning more difficult and learners considering it less difficult, and between learners who drew more complex concept maps and learners who drew less complex concept maps.

We think that these results indicate that our proposed method can assist learners to generate and process knowledge in such a way that lets even challenged learners to reach same knowledge qualities in their concept maps as the less-challenged learners can.

Table 8.4. Distribution of rankings of concepts of concept maps about programming in respect to responses given by students, for concepts mentioned by at least two students.

How much you have experience about programming before participating programming course? (It can be expected to be clear for the students from the context that this question refers specifically to their current introductory programming course.)					
Very little or little (n=80+39=119)		Moderately (n=20)		Very much or much (n=1+7=8)	
Concept	*Occurrences*	*Concept*	*Occurrences*	*Concept*	*Occurrences*
programming	71	programming	14	java	5
object	57	compiler	7	programming	5
method	55	programming language	7	language	4
java	46	java	6	c	3
class	45	program	6	program	3
program	38	programmer	6	php	2
variable	36	method	5	programmer	2
programming language	35	object	5	programming language	2
python	28	object-oriented programming	5	python	2
c	23	variable	4		
Is it easy for you at the moment to learn programming?					
Very easy or easy (n=10+42=52)		Moderate (n=85)		Very difficult or difficult (n=1+9=10)	
Concept	*Occurrences*	*Concept*	*Occurrences*	*Concept*	*Occurrences*
programming	30	programming	55	object	5
java	21	object	41	programming	5
method	16	method	40	method	4
object	16	class	35	program	4
program	16	java	34	class	3
programming language	16	program	27	user	3
c	12	programming language	26	variable	3
object-oriented programming	12	variable	26	algorithm	2
variable	12	python	21	c	2
class	11	c	15	code	2
programmer	11			computer	2
				int	2
				java	2
				language	2
				object-oriented programming	2
				programmer	2
				programming language	2
Please give an estimate about how complex things your concept map is dealing with?					
Very simple or simple (n=32+83=115)		Moderate (n=26)		Very complex or complex (n=4+2=6)	
Concept	*Occurrences*	*Concept*	*Occurrences*	*Concept*	*Occurrences*
programming	71	programming	15	programming	4
object	51	object	10	java	2
java	50	method	9	language	2
method	50	program	9	program	2
class	41	programming language	9	python	2
program	36	class	7		
programming language	35	programmer	7		
variable	34	variable	6		
python	26	compiler	5		
c	24	java	5		

Table 8.5. Distribution of rankings of conceptual relationships of concept maps about programming in respect to responses given by students, for conceptual relationships mentioned by at least two students.

How much you have experience about programming before participating introductory programming course? (It can be expected to be clear for the students from the context that this question refers specifically to their current introductory programming course.)					
Very little or little (n=80+39=119)		Moderately (n=20)		Very much or much (n=1+7=8)	
Conceptual relationship	*Occurrences*	*Conceptual relationship*	*Occurrences*	*Conceptual relationship*	*Occurrences*
class->object	25	programming->programming language	5	language->java	3
object->method	24	object->method	5	programming->programming language	2
programming->programming language	20	programming->language	3	language->c	2
class->method	14	programming language->java	3	programming->language	2
java->object	14	*(many, shown in footnote)* [1]	2	programming language->java	2
programming->program	13				
programming language->java	13				
programming->language	12				
object->variable	10				
language->c	8				
language->java	8				
variable->object	8				
program->class	8				
object->class	8				

Is it easy for you at the moment to learn programming?					
Very easy or easy (n=10+42=52)		Moderate (n=85)		Very difficult or difficult (n=1+9=10)	
Conceptual relationship	*Occurrences*	*Conceptual relationship*	*Occurrences*	*Conceptual relationship*	*Occurrences*
programming->programming language	12	class->object	19	object->method	3
object->method	9	object->method	17	class->object	2
class->object	6	programming->programming language	15	programming->language	2
programming->program	5	programming language->java	13	object-oriented programming->java	2
language->java	5	java->object	12	programmer->code	2
programming language->java	5	programming->language	11	programming->object-oriented programming	2
java->object-oriented programming	4	class->method	10		
class->method	4	object->variable	9		
language->c	4	programming->program	8		
programming->language	4	program->class	7		
variable->object	4				
object->class	4				
programming->variable	4				

Please give an estimate about how complex things your concept map is dealing with?					
Very simple or simple (n=32+83=115)		Moderate (n=26)		Very complex or complex (n=4+2=6)	
Conceptual relationship	*Occurrences*	*Conceptual relationship*	*Occurrences*	*Conceptual relationship*	*Occurrences*
object->method	25	programming->programming language	6	programming->language	2
class->object	22	class->object	4		
programming->programming language	21	object->method	4		
programming language->java	16	object->variable	3		
programming->language	14	programmer->program	3		
java->object	13	programming language->object-oriented programming	3		
class->method	12	*(many, shown in footnote)* [2]	2		
programming->program	11				
language->java	10				
object->variable	9				
language->c	9				

[1] Two occurrences: class->method; class->variable; input->method; method->object; method->output; method->variable; package->class; program->bug; program->class; program->compiler; program->function; program->library; program->user; programmer->programming; programming->logic; programming->program; programming language->c; programming language->java; programming language->machine language.

[2] Two occurrences: c->c++; class->object; class->variable; input->method; java->object-oriented programming; method->variable; method->output; object->variable; program->bug; program->class; program->compiler; program->function; program->library; programming language->c;

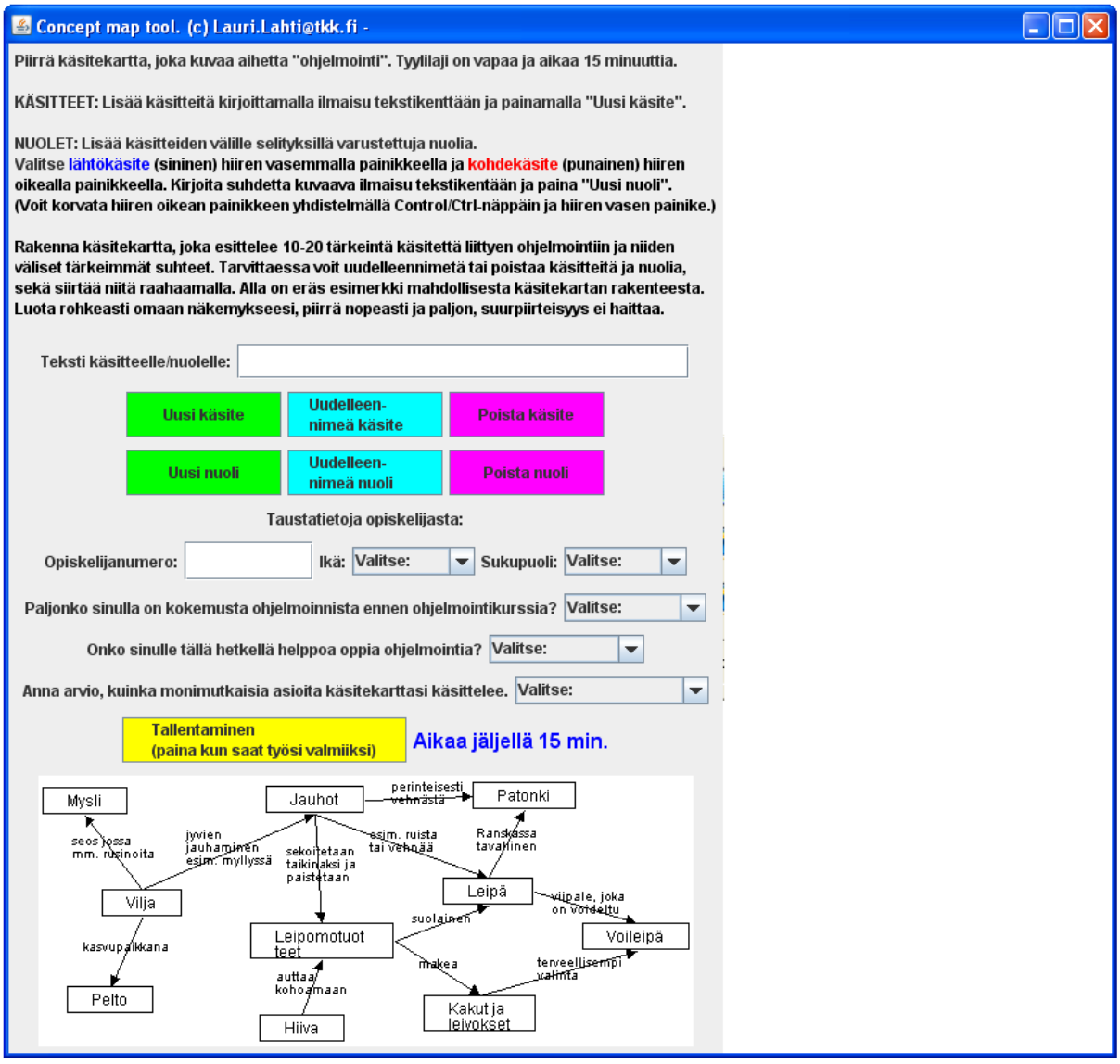
References:

Lahti, L. (2015a). Computer-assisted learning based on cumulative vocabularies, conceptual networks and Wikipedia linkage. Doctoral dissertation, Department of Computer Science, Aalto University School of Science, Finland.

Lahti, L. (2015b). Supplement to doctoral dissertation "Computer-assisted learning based on cumulative vocabularies, conceptual networks and Wikipedia linkage. Department of Computer Science, Aalto University School of Science, Finland.

Appendix M

User interface of a prototype tool used by 147 university students of introductory Java programming course who we asked to draw with our method concept maps representing their knowledge about learning topic “programming” (texts provided only in Finnish), as discussed in Subchapter 8.2.



English translation of texts of the user interface:

Draw a concept map that describes topic "programming". Presentation style is free and available time 15 minutes.

CONCEPTS: Add concepts by drawing an expression to text field and pressing "New concept".

ARROWS: Add between concepts arrows supplied with descriptions. Select start concept (blue) with left mouse button and end concept (red) with right mouse button. Write expression that describes relation to text field and press "New arrow". (You can replace right mouse button with combination Control/Ctrl button and left mouse button.)

Build a concept map that presents 10-20 most important concepts concerning programming and the most important relationships between them. If needed you can rename or remove concepts and arrows and move them by dragging. Below is an example of possible structure of a concept map. Have a confidence with your own opinion, draw fast and a lot, approximateness is not a problem.

Text for concept/arrow.
New concept, rename concept, remove concept, new arrow, rename arrow, remove arrow.

Background information about the student.
Student number. Age (select), gender (select).
How much you have experience about programming before participating programming course? (select).
Is it easy for you at the moment to learn programming? (select).
Please give an estimate about how complex things your concept map is dealing with? (select).

Saving (press when you have finished your work). Time left 15 min.
(An example of concept map structure.)

Responses of 147 university students of introductory Java programming course who we asked to draw with our method concept maps representing their knowledge about learning topic "programming". After eliminating unclear responses and transforming all concepts to non-conjugated base forms, there were 167 unique concepts and 167 unique conceptual relationships between them mentioned by at least two students. Both a listing of these unique concepts and a listing of these unique conceptual relationships are shown in table below showing number of occurrences in all 147 concept maps.

Concept	*Occurrences (at most one occurrence counted for each student)*		*Conceptual relationship*	*Occurrences (at most one occurrence counted for each student)*
programming	90		object -> method	29
object	62		class -> object	27
method	60		programming -> programming language	27
java	57		programming language -> java	18
class	49		programming -> language	17
program	47		class -> method	14
programming language	44		java -> object	14
variable	41		programming -> program	14
python	31		object -> variable	12
c	29		language -> java	11
programmer	25		language -> c	10
language	24		program -> class	10
object-oriented programming	22		object -> class	9
computer	21		variable -> object	9
user	21		java -> object-oriented programming	8
compiler	20		language -> python	8
c++	19		programming language -> c	8
code	17		programming -> object	8
user interface	16		programming -> object-oriented programming	8
loop	13		programming language -> python	7
debugger	12		class -> variable	6
eclipse	12		method -> object	6
problem	11		object-oriented programming -> java	6
algorithm	9		programming -> computer	6
conditional sentence	9		programming -> java	6
int	9		programming -> tool	6
parameter	9		c -> c++	5
program code	9		code -> program	5
ready program	9		java -> class	5

starting method	9		method -> class	5
tool	9		method -> variable	5
library	8		package -> class	5
machine language	8		programmer -> program	5
testing	8		programmer -> programming	5
constructor	7		programmer -> programming language	5
list	7		programming -> programmer	5
string	7		programming -> user interface	5
double	6		programming -> variable	5
function	6		variable -> method	5
gui	6		class -> constructor	4
operating system	6		code -> compiler	4
planning	6		java -> variable	4
assembly	5		object-oriented programming -> object	4
bug	5		program -> user	4
debugging	5		programming language -> c++	4
grafical user interface	5		programming -> code	4
hardware	5		programming -> python	4
instance variable	5		user -> program	4
package	5		c -> language	3
php	5		java -> language	3
prosessor	5		java -> method	3
application generator	4		language -> assembly	3
boolean	4		language -> paradigm	3
command	4		loop -> for	3
editor	4		loop -> while	3
information structure	4		method -> value	3
internet	4		object -> list	3
javascript	4		object-oriented programming -> class	3
lecture	4		object-oriented programming -> python	3
mathematics	4		problem -> programming	3
memory	4		program code -> object	3
paradigm	4		program -> code	3
primitive type	4		program -> library	3
primitive variable	4		program -> object	3
programming environment	4		program -> operating system	3
source code	4		program -> user interface	3
syntax	4		programmer -> code	3
value	4		programming language -> code	3
abstraction level	3		programming language -> machine language	3
aim	3		programming language -> object-oriented programming	3
application	3		programming -> assistive tool	3
application program	3		programming -> user	3
assistive tool	3		python -> language	3
basic	3		python -> object	3
char	3		tool -> compiler	3
client	3		tool -> debugger	3
coding	3		user -> code	3
concept	3		variable -> instance	3
for	3		variable -> local	3
functional programming	3		abstraction level -> high	2
functioning of program	3		abstraction level -> low	2
functioning program	3		assistive tool -> debugger	2
human	3		assistive tool -> eclipse	2
if	3		c++ -> language	2
instance	3		c++ -> object-oriented programming	2
keeper of the most recent	3		c++ -> program	2
local	3		class -> program code	2
logic	3		code -> programming language	2
parsing	3		compiler -> machine language	2
plan	3		computer -> code	2
structure	3		computer -> program	2
task	3		computer -> programming	2
while	3		eclipse -> debugger	2
virtual machine	3		editor -> code	2
visual basic	3		information -> variable	2

agile	2		input -> method	2
artificial intelligence	2		java -> concept	2
asm	2		java -> eclipse	2
basic data type	2		java -> loop	2
book	2		java -> program	2
c language	2		language -> c++	2
c#	2		library -> class	2
c/c++	2		loop -> do	2
clarity	2		loop -> programming	2
coder	2		memory -> prosessor	2
computation	2		method -> output	2
computer program	2		method -> parameter	2
constructor parameter	2		object -> algorithm	2
database	2		object -> object	2
development	2		object -> parameter	2
do	2		object -> programming	2
documentation	2		object-oriented programming -> variable	2
else	2		parameter -> method	2
environment	2		plan -> program code	2
example	2		primitive type -> boolean	2
for example java	2		problem -> program	2
function programming	2		problem -> programming language	2
function-based	2		program code -> class	2
google	2		program code -> variable	2
grafical	2		program -> application	2
hardware level	2		program -> bug	2
high	2		program -> compiler	2
history	2		program -> function	2
i	2		program -> hardware	2
ide	2		program -> method	2
information	2		program -> other programmer	2
information processing	2		program -> tool	2
input	2		program -> variable	2
integer	2		programmer -> user	2
java programming	2		programming environment -> eclipse	2
keeper of the most suitable	2		programming language -> php	2
local variable	2		programming language -> programming	2
logic operator	2		programming -> abstraction level	2
logic thinking	2		programming -> algorithm	2
loosing attention	2		programming -> c	2
low	2		programming -> c++	2
machine	2		programming -> class	2
not working	2		programming -> compiler	2
object-based	2		programming -> computer program	2
other	2		programming -> function	2
other language	2		programming -> information structure	2
other object	2		programming -> logic	2
other programmer	2		programming -> machine language	2
output	2		programming -> mathematics	2
pascal	2		programming -> method	2
private	2		programming -> other language	2
problem/task	2		programming -> program code	2
procedural programming	2		programming -> programming environment	2
public	2		programming -> style	2
result	2		programming -> theory	2
returning of value	2		prosessor -> program	2
role	2		starting method -> class	2
scheme	2		structure -> conditional sentence	2
software	2		testing -> programming	2
solution	2		tool -> application generator	2
studying	2		user interface -> grafical	2
style	2		user interface -> program	2
syntax error	2		user interface -> text-based	2
table	2		user interface -> user	2
text-based	2		user -> programmer	2
theory	2		variable -> class	2
type	2		variable -> double	2

utility program	2		variable -> instance variable	2
waterfall	2		variable -> int	2
web	2		variable -> keeper of the most recent	2
void	2		variable -> keeper of the most suitable	2
working life	2		variable -> parameter	2

This is a listing of response alternatives for self-evaluation of 147 university students of introductory Java programming course who we asked to draw with our method concept maps representing their knowledge about learning topic "programming" (analysis of responses given by students is discussed in Subchapter 8.2)

For three questions the student replied by selecting a most suitable answer from a scale of five given alternatives that are listed here next.

Response alternatives for question "How much you have experience about programming before participating introductory programming course?" (it can be expected to be clear for the students from the context that this question refers specifically to their current introductory programming course):
Very much; Much; Moderately; Little; or Very little.
(In Finnish: Paljonko sinulla on kokemusta ohjelmoinnista ennen ohjelmontikurssi?
Erittäin paljon; Paljon; Kohtalaisesti; Vähän; tai Erittäin vähän.)

Response alternatives for question "Is it easy for you at the moment to learn programming?":
Very easy; Easy; Moderate; Difficult; or Very difficult.
(In Finnish: Onko sinulle tällä hetkellä helppoa oppia ohjelmointia?
Erittäin helppoa; Helppoa; Kohtalaista; Vaikeaa; tai Erittäin vaikeaa.)

Response alternatives for question "Please give an estimate about how complex things your concept map is dealing with?":
Very complex; Complex; Moderate; Simple; or Very simple.
(In Finnish: Anna arvio, kuinka monimutkaisia asioita käsitekarttasi käsittelee.
Erittäin monimutkaisia; Monimutkaisia; Kohtalaisia; Yksinkertaisia; tai Erittäin yksinkertaisia.)