Assessment of Java Programming Self-Efficacy among Engineering Students in a Typical Nigerian University

[*1]FASOGBON S.K., [2]P.O. JEGEDE, [3]D.A. ADETAN & [4]A.A. ADERIBIGBE

[1]Department of Mechanical Engineering, University of Ibadan, Ibadan, Nigeria
[2]Institute of Education, Faculty of Education, Obafemi Awolowo University, Ile-Ife, Nigeria
[3]Department of Mechanical Engineering, Obafemi Awolowo University, Ile-Ife, Nigeria
[4]Department of Mechanical Engineering, Ladoke Akintola University, Ogbomosho, Nigeria
[*]Email: kolasogbon@yahoo.com; sk.fasogbon@ui.edu.ng

Abstract

This work reports an assessment of the impact of prior non-programming computing knowledge, gender and course of study on Java programming self-efficacy among engineering students in a typical Nigerian university. An instrument for assessing Java programming self-efficacy was developed from the computer programming self-efficacy scale of Ramalingam and Wiedenbeck, 1998. The instrument was administered to the final year engineering students in Obafemi Awolowo University, Ile-Ife, Nigeria, along with a questionnaire on gender, course of study and computer experience. Results indicated that self-efficacy of males were stronger than that of females, computer engineering students' self-efficacy scores were significantly higher than that of students from the other engineering departments and 6.6 % of the variance in self-efficacy was explained by computer experience. The study concluded that self-efficacy is highly relevant in the acquisition of Java programming skills and is in line with Bandura's theory.

Keywords: Java programming, self-efficacy, engineering students, Nigerian universities

Introduction

Self-efficacy is the belief that one is capable of performing in a certain manner to attain certain goals. Unlike efficacy, which is the power to produce an effect (in essence competence), self-efficacy is the belief (whether or not accurate) that one has the power to produce that effect. For example, a person with high self-efficacy may engage in a more health related activity when an illness occurs, whereas a person with low self-efficacy would harbour feelings of hopelessness (Bandura, 1997 and Çigdem and Yildirim, 2014). The concept of self-efficacy lies at the centre of Bandura's social cognitive theory which emphasizes the role of observational learning and social experience in the development of personality (Bussey and Bandura, 1999). According to Bandura's theory, people with high self-efficacy are more likely to view difficult task as something to be mastered rather than something to be avoided. People will be more inclined to take on a task if they believe they can succeed. People generally avoid task where their self-efficacy is low, but will engage in tasks where their self-efficacy is high. People with a self-efficacy significantly beyond their actual ability often overestimate their ability to complete tasks, which can lead to difficulties. On the other hand, people with a self-efficacy significantly lower than their ability are unlikely to grow and expand their skills (Stajkovic and Luthans, 1998). Research shows that the 'optimum' level is a little above ability which encourages people to tackle challenging tasks and gain valuable experiences (Barrows, 1996). People with high self-efficacy in a task are more likely to make more of an effort, and persist longer than those with low efficacy. The stronger the efficacy or mastery expectations, the more active the efforts, On the other hand, low self-efficacy provides an incentive to learn more about the subject. As a result, someone with a high efficacy may not prepare sufficiently for a task. Low self-efficacy can lead people to believe tasks are harder than they actually are. This often results in poor task planning, as well as increased stress. Observational evidence shows that people become erratic and unpredictable when engaging in a task in which they have low efficacy. On the other hand, people with high self-efficacy are shown to be encouraged by obstacles to make greater efforts.

Self-efficacy also affects how people respond to failure. A person with a high self-efficacy will attribute the failure to external factors, where a person with low self-efficacy will attribute failure to low ability (Stajkovic and Luthans, 1998). Bandura showed that people of differing self-efficacy

perceive the world in fundamentally different ways. People with a high self-efficacy are generally of the opinion that they are in control of their own lives; that their own actions and decisions shape their lives. On the other hand, people with low self-efficacy may see their lives as somewhat out of their hands.

Bandura points to four factors affecting self-efficacy. These factors are as follows:

(i) Experience - Mastery experience is the most important factor deciding a person's self-efficacy. Simply put, success raises self-efficacy, failure lowers it.

(ii) Modelling (Vicarious Experience) - "If they can do it, I can do it as well". This is a process of comparison between a person and someone else. When people see someone succeeding at something, their self-efficacy will increase; and where they see people failing, their self-efficacy will decrease. This process is more effectual where the person sees himself as similar to his or her model.

(iii) Social Persuasions - This relates to encouragements or discouragements. These can have a strong influence. Most people remember times where something said to them significantly altered their confidence. Where positive persuasions increase self-efficacy, negative persuasions decrease it. It is generally easier to decrease someone's self-efficacy than it is to increase it.

(iv) Psychological Factors - In unusual, stressful situations, people commonly exhibit signs of distress, shakes, aches and pains, fatigue, fear, nausea, etc. A person's perception of these responses can markedly alter his self-efficacy. If a person gets butterflies in the stomach before public speaking, Someone with low self-efficacy may take this as a sign of his own inability, thus decreasing his efficacy further, while someone with high self-efficacy is likely to interpret such psychological signs as normal and unrelated to his or her actual ability. Thus it is the person's belief in the implications of his physiological response that alters his self-efficacy, rather than the sheer power of the response.

Java is a programming language originally developed by James Gosling at sun Microsystems and released in 1995 as a core component of Sun Microsystems, Java platform. The language derives much of its syntax from C and C++ but has a simpler object model and fewer low-level facilities. Java applications are typically compiled to byte code that can run on any Java virtual machine (JVM) regardless of computer architecture.

There were five primary goals in the creation of the java language (Byous, 2009): (i) It should be simple, object oriented and familiar; (ii) It should be robust and secure; (iii) It should be architecture neutral and portable; (iv) It should execute with high performance; and (v) It should be interpreted, threaded and dynamic. One characteristics of Java is portability, which means that computer programs written in the Java language must run similarly on any supported hardware operating system platform. One should be able to write a program once, compile it once, and run it anywhere. This is achieved by compiling the java language code, not to machine code but to java byte code – instruction analogous to machine code but intended to be interpreted by a virtual machine (VM) written specifically for the host hardware. End – users commonly use a java Run time Environment (JRE) installed on their own machine for standalone java applications, or in a web browser for java applets. Java uses an automatic garbage collector to manage memory in the object lifecycle. The programmer determines when objects are created, and the Java run time environment is responsible for recovering the memory once objects are no longer in use. The syntax of Java is largely derived from C++. Unlike C++, this combines the syntax for structured, generic, and object-oriented language. All codes are written inside a class and everything is an object, with the exception of the intrinsic data types (ordinal and real numbers, Boolean values and characters), which are not classes for performance reasons.

The advent of Information and Communication Technology (ICT) has resulted in changes in some ways that we conduct teaching and learning in schools from conventional drill and practice and direct instruction to a more constructivist, learner-centered teaching and learning. A constructivist learning environments refers to a place where learners may work together and support each other as they use a variety of tools and information resources in their guided pursuit of learning goals and problem-solving activities (Wilson, 1996). The implications of this in education are obvious, and have been discussed and promoted in educational research (Wilson, 1996). Some of the instances of constructivist learning environment are problem-based learning (Barrows, 1996), learning through relate-create-donate **(**Kearsley and Shneiderman, 1998), activity-based learning (Jonassen and Rohrer-Murphy, 1999; Slavin, 1995), collaborative learning (Johnson, and Johnson, 1996) and learning through construction (Kafai, and Resnick, 1996). Jonassen and Rohrer-Murphy (1999) proposed the idea of constructivist teaching and learning through

"Engagement Theory", they suggested that students must be meaningfully engaged in learning activities through interaction with others and worthwhile tasks. They proposed three basic engagement principles. They are:

(i) Relate - emphasizes team effort that involves communication, planning, management and social skills. Students are requested to clarify and verbalize their problems (in groups), thereby facilitating solutions and multiple perspectives;

(ii) Create - Act of creating makes learning a creative, purposeful activity (conducting a project is more interesting than answering sterile textbook problems). Project-based learning is the essence of problem-based learning (PBL) approaches; and

(iii) Donate or have an authentic focus. This stresses on the value of making a useful contribution or service (to any kind of customers) while learning. This will increase motivation and the meaning of learning.

Many researchers have explored the relationship among academic performance, self-efficacy and demographical variables on different fields. Computer self-efficacy is one of those that have attracted the interest of several researchers from a variety of disciplines (Albion, 2001; Askar and Umay, 2001; Delcourt.and Kinzie, 1993; Karsten, and Roth, 1998; Compeau and Higgins, 1995; Kurbanoglu, 2003; Usluel, 2007). Despite this, surprisingly little attention has been paid to its role in the acquisition of computer programming skills. Ramalingam and Wiedenbeck (1998) developed and validated a self-efficacy scale for the C++ language programming. Their results using this scale seem to support the applicability of self-efficacy to C++ programming. They found that the self-efficacy of males and females did not differ substantially in practical terms. Female self-efficacy scores were found to be significantly lower than those of males, even after 12 weeks of instruction. However, the difference was small accounting for only 2% of the variance. They explained this by noting that the female students in their study were self-selected and probably had better Mathematics skills and more computer experience than average. Since such abilities are assumed to be highly relevant to the acquisition of computer programming skills, the lack of practical difference was perhaps not surprising. Of more importance was their observation of a general increase in self-efficacy on the post-instruction measure, independent of gender, but with the greatest gains seen in those with lower initial self-efficacy scores. This is in line with theory, which suggests that the self-

efficacy of beginning students is highly responsive to performance accomplishments in the early stages of skill acquisition.

In another study (Ramalingam et al., 2004), the researchers investigated the effects of self-efficacy and mental models of programming. The results showed that self-efficacy for programming was influenced by previous programming experience. The factors which relate to self-efficacy for java programming among first year engineering students were examined by (Askar and Davenport, 2009). The result indicated that self-efficacy of males were stronger than that of females. Considering the significant global demand for computer literate engineers, it is crucial to understand what factors influence an individual's choice of computing as a profession and subsequently affect their ability to acquire computer programming skills. Nowadays, computer programming is a common, often mandatory part, of an undergraduate engineering education. Yet, in contrast to subjects such as mathematics and physics, students tend to perceive programming courses as being somewhat difficult (Cassidy and Eachus, 1998). In part, this is undoubtedly due to unfamiliarity with the subject, since computing (unlike mathematics and physics) has not traditionally been part of the high school curriculum. But in spite of the fact that students are increasingly likely to have had prior exposure to computers both at home and school –albeit generally as a tool for writing reports, communicating with friends or simply playing games, rather than as something that they can program to perform new tasks– such negative perceptions seem to persist. This study thus focuses on the relationship between programming self-efficacy beliefs and other variables (including gender, subject/department choice, and computer experience).

Materials and methods

A Questionnaire was designed and a Java programming self-efficacy scale was developed to investigate the following research questions, which form the core of the study:

(i) Is there a significant difference between male and female engineering students' self-efficacy for Java programming?
(ii) Are there significant differences in self-efficacy of Java programming between students from the various engineering departments?
(iii) How do prior computing skills and frequency of computer use affect self-efficacy beliefs?

Final year first degree engineering students of Obafemi Awolowo University, Ile-Ife, Nigeria who had completed all computer programming courses required for their graduation were asked to complete the questionnaire and the self-efficacy scale at their various departments.

The subjects in this study were 216 final year engineering students chosen at random (50 from the computer science and engineering, 50 from chemical engineering, 40 from mechanical engineering departments, 40 from electrical and electronic, 20 from civil engineering, 16 from Material science and engineering). All the students completed the self-efficacy scale and returned demographic data including gender, age, department and years of computer experience. All the participants were students at Obafemi Awolowo University, Ile-Ife, Nigeria. As such, they should be considered very good academically, having attended one of the best institutions in Nigeria. All instructions at Obafemi Awolowo University are in English language and students are required to pass an English language proficiency examination prior to entering their departments. Computer programming is a required course for all engineering students in the school. The choice of university-level engineering students was informed by the belief that students at this level will naturally have higher self-efficacy beliefs than their counterparts in other levels.

The Java programming self-efficacy scale was first developed in line with the internationally accepted format as standard (Ramalingam and Wiedenbeck, 1998). The overall reliability of the self-efficacy scores for their C++ scale was 0.98. The corrected item-total correlations ranged between 0.5 and 0.85. The new scale for Java Programming consisted of 32 items and the reliability of the scores was 0.99 (taken across all 216 students). The scale, which was in English, is given in Appedix 1. The participants were given instructions to: "*Rate your confidence in doing the following Java programming related tasks using a scale of 1 (not at all confident) to 7 (absolutely confident).*" In addition to the scale, a questionnaire for collecting information related to gender, department, and prior computer experience, was prepared and delivered to the students by direct contact in their lecture rooms.

Analysis of variance, principal components and regression analyses were undertaken using SPSS to investigate any relationship among self-efficacy for Java programming and the other variables obtained from the questionnaire. The overall self-efficacy scores were used in the analyses.

Results and Discussions

Gender issues

Means and standard deviations of the self-efficacy scores of female and male students are given in Table 1. It shows that on the average, the self-efficacy of males is higher than that of females.

Table 1: Mean and SD of Self-Efficacy Scores according to Gender of the Students

	Gender	N	Mean	Std. Deviation	Std. Error of Mean
Java self efficacy	Male	190	69.2316	51.70203	3.75086
	Female	26	54.5769	39.43924	7.73467

This difference between males and females with regard to the self-efficacy for Java programming is statistically significant at less than 5% significance level. This can be seen from analysis of variance (ANOVA) Table 2. Thus, gender issue significantly influences self-efficacy among engineering students in the typical Nigerian university. From our results, it can be seen that the mean 69.2316 self-efficacy obtained for males is closer to the optimum mean 100 value expected, this is however contrary in the case of females, as the mean value of 54.5769 self-efficacy obtained is relatively far away. In general, we perceived that self-efficacy of males is higher than that of females because males tend to be naturally tougher and more courageous than their females counterparts.

Table 2: Analysis of variance for gender

Java self-efficacy	Sum of Squares	DOF	Mean Square	F	Sig.
Between Groups	109648.900	5	21929.780	10.454	.000
Within Groups	440526.933	210	2097.747		
Total	550175.833	215			

Course of study

Though the main effect of the course of study on self-efficacy among the students appear statistically insignificant, p being higher than 0.010 ($F(2; 217) = 1.884$, $p=0.114$) as shown in the ANOVA Table 3.

Table 3: Analysis of variance for department

Java Self-efficacy					
	Sum of Squares	**Df**	**Mean Square**	**F**	**Sig.**
Between Groups	18873.293	4	4718.323	1.884	.114
Within Groups	533496.762	213	2504.680		
Total	552370.055	217			

The multiple comparison analysis (Table 4) shows that the average computer engineering students' self-efficacy score was significantly higher than those of students on the other engineering programs. This result is perhaps as expected because computer engineering students should naturally be more comfortable and confident with computer programming software application given the fact that at their final year level, they would have gone through more computer programming courses than the other engineering students.

Table 4: Multiple comparison analysis

					95% Confidence Interval	
		Mean Difference				
(I) course of study	(J) course of study	(I-J)	Std. Error	Significance	Lower Bound	Upper Bound
Chemical Eng	Civil Eng	32.03463*	14.96669	.033	2.5304	61.5388
	Computer Sc/Eng	-37.42328*	8.49381	.000	-54.1673	-20.6792
	Electrical Eng	34.76190	21.28020	.104	-7.1883	76.7121
	Mech Eng	18.28114*	8.58131	.034	1.3646	35.1977
	Met Eng	2.92320	10.04823	.771	-16.8851	22.7315
Civil Eng	Chemical Eng	-32.03463*	14.96669	.033	-61.5388	-2.5304
	Computer Sc/Eng	-69.45791*	15.15096	.000	-99.3254	-39.5905
	Electrical Eng	2.72727	24.70331	.912	-45.9710	51.4255
	Mech Eng	-13.75350	15.20018	.367	-43.7180	16.2110
	Met Eng	-29.11144	16.07400	.072	-60.7985	2.5756
Computer Sc/Eng	Chemical Eng	37.42328*	8.49381	.000	20.6792	54.1673
	Civil Eng	69.45791*	15.15096	.000	39.5905	99.3254
	Electrical Eng	72.18519*	21.41020	.001	29.9787	114.3916
	Mech Eng	55.70442*	8.89879	.000	38.1620	73.2468
	Met Eng	40.34648*	10.32068	.000	20.0011	60.6919
Electrical Eng	Chemical Eng	-34.76190	21.28020	.104	-76.7121	7.1883
	Civil Eng	-2.72727	24.70331	.912	-51.4255	45.9710
	Computer Sc/Eng	-72.18519*	21.41020	.001	-114.3916	-29.9787
	Mech Eng	-16.48077	21.44506	.443	-58.7560	25.7944
	Met Eng	-31.83871	22.07303	.151	-75.3518	11.6744
Mech Eng	Chemical Eng	-18.28114*	8.58131	.034	-35.1977	-1.3646
	Civil Eng	13.75350	15.20018	.367	-16.2110	43.7180
	Computer Sc/Eng	-55.70442*	8.89879	.000	-73.2468	-38.1620
	Electrical Eng	16.48077	21.44506	.443	-25.7944	58.7560
	Met Eng	-15.35794	10.39281	.141	-35.8456	5.1297
Met Eng	Chemical Eng	-2.92320	10.04823	.771	-22.7315	16.8851
	Civil Eng	29.11144	16.07400	.072	-2.5756	60.7985
	Computer Sc/Eng	-40.34648*	10.32068	.000	-60.6919	-20.0011
	Electrical Eng	31.83871	22.07303	.151	-11.6744	75.3518
	Mech Eng	15.35794	10.39281	.141	-5.1297	35.8456

*. The mean difference is significant at the 0.05 level

Computer experience

The study found a relatively moderate correlation of 0.59 between self-efficacy beliefs and prior (non-programming) experience with computers. Simple Regression analysis revealed that the number of years of experience a student had with computers had a significant linear correlation with their self-efficacy scores. (F (1, 217) = 1.868, p=0.118). About 6.6% of the variance in self-efficacy was explained by the number of years of computer experience a student had. As the computer experience increases, there is a tendency to gain self-efficacy in programming. The analysis of variance is shown in Table 3.5.

Table 5: Analysis of variance for computer experience

Java Self-efficacy					
	Sum of Squares	**DOF**	**Mean Square**	**F**	**Sig.**
Between Groups	19091.225	4	4772.806	1.868	.118
Within Groups	480373.490	188	2555.178		
Total	499464.715	192			

A general observation from the results of this study is that the Java programming self-efficacy of the typical Nigerian university engineering student is generally low considering the expected optimum 100 mean value self-efficacy with the obtained male 69.2316 mean value and female 54.5769 mean value. This is perhaps engendered by the fact that these students have no prior pre-university education exposure to computer-related studies; most Nigerian high schools and colleges do not have computer studies in their curricula. Furthermore, students in Nigeria often select their major subject area on the basis of social and family pressures, course of study being chosen simply on the basis of the score achieved in the university entrance exam, without any real notion of what the subject might involve.

Conclusion

The results of this work indicated that female students had significantly lower initial self-efficacy beliefs compared with those of their male peers. The results also confirmed the fact that the subject/ career choice of study has a direct link with self-efficacy belief as the computer engineering students recorded higher self-efficacy for Java programming (their chosen major) than those who selected electrical and electronics,

mechanical, civil, chemical and material science and engineering. On the other hand, the self-efficacy scores of all the students were remarkably low. Overall, our results confirm the relevance of self-efficacy to the acquisition of Java programming skills and are in line with Bandura's theory.

References

Albion P.R. (2001). "Some factors in the development of self-efficacy beliefs for computer use among teacher education students", *Journal of Technology and Teacher Education*, Vol. 9, Issue 3.

Askar, P. and D. Davenport (2009). An investigation of factors related to self-efficacy for java programming among engineering students, The Turkish Online Journal of Educational Technology, TOJET January 2009 ISSN: 1303- 6521, volume 8 Issue 1.

Askar, P. and A. Umay (2001). "İlköğretim matematik öğretmenliği öğretmen adaylarının bilgisayarla ilgili özyeterlikalgısı" ("Perceived computer literacy of the students in the elementary mathematics program"), *Hacettepe Üniversitesi Eğitim Fakültesi Dergisi*, Vol.21.

Bandura, A. (1997). *Self-efficacy: The exercise of control*. New York: Freeman

Barrows, H. S. (1996). Problem-based learning in medicine and beyond: A brief overview. In L. Wilkerson and W. H. Gijselaers (Eds.), Bringing problem-based learning to higher education: Theory and Practice (pp. 3-12). San Francisco: Jossey-Bass.

Bussey, K. and A. Bandura (1999). Social cognitive theory of gender development and differentiation. *Psychology Review, 106*, 676-713.

Byous, J. (2009) "Notes on Design Goals of the Java TM programming language" retrieved from: http://java.sun.com/docs/white/ langenv/Intro.doc2.html.

Cassidy, S. and P. Eachus (1998). " Developing the computer self-efficacy (CSE) scale; investigating the relationship between CSE, gender and experience with computers", Computer Self-Efficacy Web Site, available at: www.chssc.salford.ac.uk/healthSci/ selfeff/ selfeff.htm (accessed May, 5, 2002).

Çigdem, H. and O.G. Yildirim (2014). Predictors of C# programming language self efficacy among vocational college students. International Journal on New Trends in Education and their Implications. Volume 5 Issue 3 Article 15 ISSN 1309-6249.

Compeau, D.R. and C.A. Higgins (1995). "Computer self-efficacy: development of a measure and initial test", *MIS Quarterly*, June, pp.189-211.

Delcourt, M. and M. Kinzie (1993). "Computer technologies in teacher education: the measurement of attitudes and self-efficacy", *Journal of Research and Development in Education*, Vol.27, pp.31-7.

Jonassen, D.H., and L. Rohrer-Murphy (1999). Activity theory as a framework for designing constructivist learning environments. Educational Technology Research and Development (ET R&D), 47(1), 61-79.

Johnson, D.W. and R.T. Johnson, R.T. (1996). Cooperation and the use of technology. In Jonassen, D.H. (Ed.) *Handbook of research for educational communications and technology: A project of the Association for Educational Communications and Technology (AECT),* New York: Macmillan, 1017-1044.

Kafai, Y., and M. Resnick (Eds.). (1996). Constructionism in Practice: Designing, thinking, and learning in a digital world. New Jersey: Lawrence Erlbaum Associates.

Karsten, R. and M.R. Roth (1998), "The relationship of computer experience and computer self-efficacy to performance in introductory computer literacy courses", *Journal of Research on Technology Education*,Vol.31 No.1, pp. 14-24.

Kearsley, G. and B. Shneiderman (1998). Engagement theory: A framework for technology-based teaching and learning. Educational Technology, 38(5), 20-23.

Kurbanoglu, S.S. (2003), "Self-efficacy: a concept closely linled to information literacy and lifelong learning", *Journal of Documentation*, Vol. 59 No.6, pp. 635-646.

Ramalingam, V. and Wiedenbeck, S. (1998) Development and Validation of Scores on a Computer Programming Self-efficacy Scale and Group Analysis of Novice Programmer Self-efficacy. *Journal of Educational Computing Research*, Vol. 19 (4), pp367-381.

Ramalingam, V., D. La Belle and Wiedenbeck, S (2004) Self-efficacy and mental models in learning to program. ACM SIGCSE Bulletin, Vol. 36 Issue 3, 171-175.

Slavin, R.E. (1995). Cooperative learning and intergroup relations. In Banks, J.A. and McGee Banks, C.A. (Eds.) *Handbook of research on multicultural education,* New York: Macmillan Publishing, 628-634.

Stajkovic, A.D. and F. Luthans (1998) Self-efficacy and work-related performances: A meta-analysis. *Psychological Bulletin, 124*, 240-261

Usluel,Y.K. (2007). Can ICT usage make a difference on student teachers' information literacy self-efficacy, *Library and Information Science Research*, 29, 92-102.

Wilson, B.G. (1996). What is a constructivist learning environment? In B. G. Wilson (Ed.), Constructivist learning environment: Case studies in instructional design (pp. 3-8). New Jersey: Educational Technology Publications.

Appendix 1

Java Programming Self-efficacy Scale

Rate your confidence in doing the following Java programming related tasks using a scale of 1 (not at all confident) to 7 (absolutely confident). **If a specific term or task is totally unfamiliar to you, please mark 1.**

Not confident at all	Mostly not Confident	Slightly Confident	50/50	Fairly confident	Mostly Confident	Absolutely Confident
1	2	3	4	5	6	7

1. I could write syntactically correct **Java** statements. _____
2. I could understand the language structure of **Java** and the usage of the reserved words. _____
3. I could write logically correct blocks of code using **Java** _____
4. I could write a **Java** program that displays a greeting message. _____
5. I could write a **Java** program that computes the average of three numbers. _____
6. I could write a **Java** program that computes the average of any given number of numbers. _____
7. I could use built-in functions that are available in the various Java **applets**. _____
8. I could build my own **Java applets**. _____
9. I could write a small Java program given a small problem that is familiar to me. _____
10. I could write a reasonably sized Java program that can solve a problem this is only vaguely familiar to me._____
11. I could write a long and complex Java program to solve any given problem as long as the specifications are clearly defined. _____
12. I could organize and design my program in a modular manner. _____

13. I could understand the object-oriented paradigm. _____
14. I could identify the objects in the problem domain and could declare, define, and use them. _____
15. I could make use of a pre-written function, given a clearly labeled declaration of the function. _____
16. I could make use of a class that is already defined, given a clearly labeled declaration of the class. _____
17. I could debug (correct all the errors) a long and complex program that I had written and make it work. _____
18. I could comprehend a long, complex multi-file program. _____
19. I could complete a programming project if someone showed me how to solve the problem first. _____
20. I could complete a programming project if I had only the language reference manual for help. _____
21. I could complete a programming project if I could call someone for help if I got stuck. _____
22. I could complete a programming project once someone else helped me get started. _____
23. I could complete a programming project if I had a lot of time to complete the program. _____
24. I could complete a programming project if I had just the built-in help facility for assistance. _____
25. While working on a programming project, if I got stuck at a point I could find ways of overcoming the problem. _____
26. I could come up with a suitable strategy for a given programming project in a short time. _____
27. I could manage my time efficiently if I had a pressing deadline on a programming project. _____
28. I could mentally trace through the execution of a long, complex multi-file program given to me. _____
29. I could rewrite lengthy and confusing portions of code to be more readable and clear. _____
30. I could find a way to concentrate on my program, even when there were many distractors around me. _____
31. I could find ways of motivating myself to program, even if the problem area was of no interest to me. _____
32. I could write a program that someone else could comprehend and add features to at a later date. _____