Detection and Categorization of Errors by Novice Programmers in a First Year Java Programming Class: A Comparative Analysis

Murtala Mohammed1, Abdulwahab Lawan2, Bashir S. Galadanci3, Mijinyawa M.K

1. Department of Computer Science Northwest University, Kano, Nigeria
2. Department of Information Technology, Bayero University, Kano, Nigeria
3. Department of Software Engineering Bayero University, Kano, Nigeria

Corresponding Author e-mail abd_wahhb@yahoo.com

ABSTRACT
The increasing demand for Information Technology (IT) professionals including Java programmers is rising and hence the pressure on IT institutions to train more and more students in this technical area is becoming crucial. Although students are attracted to, and may enroll in IT related undergraduate degree programs, challenges associated with computer programming and related courses are making them to later change to other fields of study. This paper reports on a study to determine whether early tutorials could effectively reduce the programming errors made by novice students taking a Java programming class at the Meliksah University. The empirical work is based on a control experiment with a key objective of determining which of the two groups of novice students taking a Java programming class had fewer programming errors. The first group of students was given a tutorial on Java programming errors; the second was not. The preliminary result shows that the tutorial given to students in the first group had a positive impact on reducing their programming errors, and therefore improving student programming skills. The results also lead to some recommendations to improve instructor teaching and student learning of Java programming courses.

KEYWORDS
Java Programming errors, Gauntlet, Control experiment, Information Technology

1. INTRODUCTION
The high demand for Information Technology (IT) practitioners is driving the increasing need for properly trained IT professionals, and highly skilled programmers as well. This is, indirectly, increasing the pressure on IT institutions to train more and more students. Conversely, some students from the beginning are attracted in enrolling in IT related degree programs but may later change to another field of study. Prior researches have discussed various challenges associated with computer programming courses [1, 2, and 3]. Computer programming in the contemporary information age is generally seen as a useful skill, and regarded as a rewarding career. Recently, the need for skilled programmers is growing rapidly. However, the interest in programming by students appears to be declining. Arguably, first year programming students suffer from a wide range of difficulties and poorly structured programming courses, are generally regarded as difficult and are said to have the highest rates of dropout. Winslow [4] suggests that it takes about ten years for a novice student to become an expert in programming. Novice students face a lot of difficulties while learning to write computer programs. The difficulties might include the programming concepts (flow of execution, functions and parameters, etc.), the semantics and syntax of the programming language being taught, and the process of implementing a solution of a problem into a computer program. The learning task is made more challenging for a novice student when the compiler (the underlying stack of software checking the students' computer program – source-code – for compliance with particular language syntax and semantics) displays error messages (indicating that the language syntax or semantics has been violated e.g. missing semicolon). This becomes more complicated if the students are unable to correctly interpret the error messages displayed by the compiler. A novice programmer would learn to reduce,
or even learn to not make, programming errors if properly given effective programming tuition and guidance. This could reduce the time that students spend in fixing errors, and also strengthen their learning of effective programming concepts. Arguably, effective techniques that help in easier identification, correction and reduction of common errors in computer programs would enable students to focus more on problem-solving, instead of fixing errors. In this study, a tutorial was used in educating students on common Java programming errors. The work being reported in this paper is a preliminary result of a long-term study to answer the research question: will exposing students about Java programming errors through tutorial at the beginning of the semester prevent them from repeating some errors? To answer this question, a comparative study was conducted between two groups of novice students to determine their performance in terms of making minimum errors.

2. LITERATURE REVIEW

Understanding the challenges faced by novice students is an important first step in helping them to learn programming. Prior study has shown that novice students lack programming strategies and have difficulties in using selection structure [5]. Other studies [6, 7] have suggested that novice students have difficulties in expressing natural language solutions into computer programming languages. Other challenges reported in [5] include analyzing and designing mathematical expressions, naming variables and assigning suitable data types and structures to these variables, correctly evaluating output statements, arithmetic expressions and relationship expressions. Prior study has suggested that novice students have difficulty in debugging/fixing loop conditions, conditional logic, arithmetic errors, update, and data initialization [8].

A further study conducted by Flowers, Carver, and Jackson [09] in the United States Military Academy with the objective of exploring the first year intake in an introductory programming skills using Java. The study found that instead of focusing on problem-solving, students taking programming classes often end up trying to fix errors shown by the compiler. Flowers, Carver, and Jackson [09] identified nine (9) common errors made by students in the context of their research. To help students conducting programming assignments, a program that pre-processes source code called 'Gauntlet' was introduced. The program explained, in simple terms, each of the syntax errors a student encountered while conducting programming assignments. It also explained, in detail, the identified errors, giving suggestions on how to fix them. It also provided students with ideas about typical causes of the errors. The use of Gauntlet benefits the novice programmers in reducing instructor load using Gauntlet help students in requiring less assistance from their instructors [9]. Jackson et al. [10] suggests that further to the findings by Flowers, Carver, and Jackson [09], errors such as incompatible types and syntax errors (e.g. missing “class” or “interface” keywords and illegal expression statements) were also observed. They argue that students had a good understanding of the learning objectives and that lack of programming experience and simple syntax errors were primarily responsible for student programming errors. Pillay, Nelishia, Vikash and Jugoo [11] conducted a study to identify the causes of students programming errors and the difficulties students encountered while learning to program in two South African institutions. The participants investigated in the research were students enrolled in a first year Java programming course. Sixty one students from the University of KwaZulu-Natal and thirty (30) students from the Mangosuthu Technikon were the participant covering the two institutions. The researchers used both qualitative and quantitative approaches during practical sessions and the examinations. Errors made by the students and learning difficulties encountered during the practical sessions were recorded by the researchers. Also, the programming errors made by students in tests and examinations were traced and recorded. Pillay, Vikash and Jugoo [11] classified the errors based on the difficulties and occurrences of particular
programming concepts, as percentages: variables and typing – 10%; arithmetic operations – 3%; logical operations – 11%; input and output – 9%; conditional control structures – 12%; iterative control structures – 23%; and modularization – 32%. Errors for each of the topics included both syntax and semantic errors. The researchers established that the main causes of errors made by students in the examination were poor problem solving ability and the lack of knowledge and understanding of the programming language.

3. METHODOLOGY

A control experiment was conducted during students' Java programming workshop. The key objective of the experiment was to compare the performance between two groups of novice students and to identify which group has fewer programming errors. The data and observations from the experiment would inform the impact of the tutorial on Java programming errors given to students in the control group, in relation to their Java programming performance.

3.1 Participants

The participants in this study were first year computer engineering students at Meliksah University that were enrolled in the spring semester, 2013/2014 session. The students registered for two (2) programming courses that were conducted using the Java programming language. The courses included: CSE (101) (an introductory programming course) and CSE201. The CSE101 was taken in the first semester, and forty-five (45) students were registered. The CSE102 was taken in the second semester, with the composition of fifty-five (55) students in the class. Students in the latter group had previous experience in Java programming by taking the pre-requisite CSE101 course. The students took a weekly programming workshop on the topics covered during their programming lectures and classes. Each workshop run for three (3) hours. For this, each student sat on a computer, and each computer had a Macintosh Operating System (Mac OS) and a recent version of Java NetBeans Integrated Development environment (IDE) Installed.

The participants were forty (40) in number, and were assigned to two groups. Twenty (20) students were assigned to the control group (Group A), and were given a tutorial on common difficulties and errors in writing a Java program. The other twenty (20) students were assigned to the uncontrolled group (Group B), and they were left unattended.

3.2 Data Collection

The data collection was based on a programming workshop comprising of ten (10) questions that tested the participating students' knowledge of basic concepts of Java programming including loop structure, array declaration, assignment of value, nested 'if' and 'if then else', method calling, and objects. This particular workshop was completed within an hour. The Java codes of each participant were collected and analyzed in the Java NetBeans IDE. The analysis sought to identify the programming errors and the number of their occurrences in each participant's codes. As illustrated in Table 1, all errors identified were categorized as 'syntax' error, 'runtime' error, or 'logical' error.

A syntax error indicates a failure to compile a particular code. The particular syntax error and its location are indicated by the compiler (an underlying software for checking code error and compliance with programming language semantics) in the Java NetBeans IDE as illustrated in Table 1. The indicated syntax error and its category are recorded. A runtime error indicates a failure to 'run' and execute a compiled code. This is observed when a program halts during execution (in the process of running). Then, the researcher conducts a 'trace' in the Java code to identify the location of the errors. The runtime errors identified were also recorded, as illustrated in Table 1. A logical error indicates bad programming logic, leading to unexpected
output when the program is compiled and executed. This is observed when the Java code is executed and the actual output is different from the desired and required output. The logical errors observed were also recorded, as shown in Table 1.

4. RESULTS AND DISCUSSIONS

The total number of errors recorded was two hundred and two (202). As illustrated in Table 1, eighty-one (81) were recorded from the control group (Group A). This is equivalent to 40.10% of the total number of errors. With respect to the uncontrolled group (Group B), 121 errors were recorded, accounting for 59.90% of the total errors.

The error data in Table 1 suggests an improvement of fewer errors in the control group (Group A) that were given a tutorial on Java programming and errors. And Table 2 shows the occurrences of Errors made by Control Group from the least to highest. On the other side, Figure 1 shows a comparative error chart as percentages of the two groups. It shows that there has been an improvement in eight (8) of the ten (10) error types being studied. However, no improvement was recorded for two (2) of the error types, these types are categorized as logical errors. Each error and the related improvement and changes as illustrated in Table 1 and Figure 1 are now discussed.

Table 1 Categorizations of Errors and Comparative Result of the Two Groups

<table>
<thead>
<tr>
<th>S/N</th>
<th>Error</th>
<th>Category</th>
<th>Total</th>
<th>N</th>
<th>%</th>
<th>Over-All %</th>
<th>N</th>
<th>%</th>
<th>Over-All %</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Off by one</td>
<td>Logical</td>
<td>42</td>
<td>16</td>
<td>38.10</td>
<td>7.92</td>
<td>26</td>
<td>61.90</td>
<td>12.87</td>
</tr>
<tr>
<td>2</td>
<td>Multiple if without else</td>
<td>Logical</td>
<td>37</td>
<td>21</td>
<td>56.76</td>
<td>10.4</td>
<td>16</td>
<td>43.24</td>
<td>7.92</td>
</tr>
<tr>
<td>3</td>
<td>Array index out of range</td>
<td>Runtime</td>
<td>27</td>
<td>9</td>
<td>33.33</td>
<td>4.46</td>
<td>18</td>
<td>66.67</td>
<td>8.91</td>
</tr>
<tr>
<td>4</td>
<td>Missing print out</td>
<td>Logical</td>
<td>22</td>
<td>9</td>
<td>40.91</td>
<td>4.46</td>
<td>13</td>
<td>59.09</td>
<td>6.44</td>
</tr>
<tr>
<td>5</td>
<td>Code outside a method/main method class</td>
<td>Runtime</td>
<td>17</td>
<td>2</td>
<td>17.76</td>
<td>0.99</td>
<td>15</td>
<td>88.24</td>
<td>7.43</td>
</tr>
<tr>
<td>6</td>
<td>Missing [ ] in array</td>
<td>Runtime</td>
<td>16</td>
<td>6</td>
<td>37.50</td>
<td>2.97</td>
<td>10</td>
<td>62.50</td>
<td>4.95</td>
</tr>
<tr>
<td>7</td>
<td>Incorrect swapping of values</td>
<td>Logical</td>
<td>13</td>
<td>5</td>
<td>38.46</td>
<td>2.48</td>
<td>8</td>
<td>61.54</td>
<td>3.96</td>
</tr>
<tr>
<td>8</td>
<td>Incorrect printout placement</td>
<td>Logical</td>
<td>11</td>
<td>4</td>
<td>36.36</td>
<td>1.98</td>
<td>7</td>
<td>63.64</td>
<td>3.47</td>
</tr>
<tr>
<td>9</td>
<td>Forgot to create object in accessing a non-static method</td>
<td>Runtime</td>
<td>10</td>
<td>3</td>
<td>30.00</td>
<td>1.49</td>
<td>7</td>
<td>70.00</td>
<td>3.47</td>
</tr>
<tr>
<td>10</td>
<td>Incorrect accumulator</td>
<td>Logical</td>
<td>7</td>
<td>6</td>
<td>85.71</td>
<td>2.97</td>
<td>1</td>
<td>14.29</td>
<td>0.50</td>
</tr>
<tr>
<td></td>
<td>TOTAL</td>
<td></td>
<td>202</td>
<td>81</td>
<td>40.10</td>
<td>40.12</td>
<td>121</td>
<td>59.9</td>
<td>59.92</td>
</tr>
</tbody>
</table>

Table 2 Occurrences of Errors by Control Group from the least to highest

<table>
<thead>
<tr>
<th>S/N</th>
<th>Error</th>
<th>Category</th>
<th>Occurrences</th>
<th>% Errors</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Code outside a method/main method class</td>
<td>Runtime</td>
<td>2</td>
<td>17.76</td>
</tr>
<tr>
<td>2</td>
<td>Forget to create object in accessing a non-static method</td>
<td>Runtime</td>
<td>3</td>
<td>30.00</td>
</tr>
<tr>
<td>3</td>
<td>Array index out of range</td>
<td>Runtime</td>
<td>9</td>
<td>33.33</td>
</tr>
<tr>
<td>4</td>
<td>Incorrect printout placement</td>
<td>Logical</td>
<td>4</td>
<td>36.36</td>
</tr>
<tr>
<td>5</td>
<td>Missing [ ] in array</td>
<td>Runtime</td>
<td>6</td>
<td>37.50</td>
</tr>
<tr>
<td>6</td>
<td>Off by one</td>
<td>Logical</td>
<td>16</td>
<td>38.10</td>
</tr>
<tr>
<td>7</td>
<td>Incorrect swapping of values</td>
<td>Logical</td>
<td>5</td>
<td>38.46</td>
</tr>
</tbody>
</table>
The second error type is “Multiple if without else”, and was observed thirty-seven (37) times. Group A accounted for 56.76%, while Group B had 43.24% of this error type. Unexpectedly, the control group had more errors than the uncontrolled group. This indicates that for this error type, the control group had a worse performance. However, this negative change is relatively small, compared to the much bigger change observed in the tenth error type discussed below. The next error type in Figure 1 is “Array index out of Range”. This error was recorded twenty-seven (27) times. For the control group (Group A), 33.33% of these errors were recorded, while 66.67% of these errors were recorded for the uncontrolled group (Group B). Comparatively, the data suggests an improvement in fewer errors for the control group. The fourth error type, “Missing printout statement”, was recorded twenty-two (22) times. Of this many times, the control Group A accounted for 40.91%, while the uncontrolled Group B had 59.09% of this error type. Again, comparatively, Figure 1 shows an improvement in fewer errors for the control group. The next error type in Figure 1 is “Code outside a method or outside a main method”. This was recorded seventeen (17) times. The control group (Group A) accounted for 17.76%, while the uncontrolled group (Group B) accounted for 88.24% of this error type. Comparatively, the control group shows an improvement with fewer errors. Noticeably in Figure 1, of all the error types studied, this error type has the widest margin between the errors recorded from the control group and those from the uncontrolled group. Consistently, the over-all error comparison chart in Figure 2 also shows that this error type has the least percentage of errors recorded for the control group. Both of these observations indicate that this error type has seen the most improvement in this study.
Figure 1. Error Comparison Chart of the two groups
The sixth error type, “Missing []/{} in array” was observed sixteen (16) times. Group ‘A’ (control group) accounted for 37.50%, while Group ‘B’ (uncontrolled) had 62.50% of this error type. Also for this error type, there was an improvement in fewer errors in the control group, relative to the uncontrolled group. The next error type in Figure 1 is “Incorrect swapping of values”. This error type was observed thirteen (13) times. The control group (Group A) accounted for 38.46%, while the uncontrolled group (Group B) accounted for 61.54% of this error type. Again, an improvement in fewer errors was observed in the control group, relative to the uncontrolled group. The eighth error type is “Incorrect printout placement”. This was recorded eleven (11) times. The control group (Group A) accounted for 36.36%, while the uncontrolled group (Group B) had 63.64% of this error type. For this error type, the students in the control group were observed to gain performance with respect to having fewer errors relative to their counterpart in the uncontrolled group. The ninth error type is “Forgot to create object in calling a non-static method”, occurring ten (10) times. The control group (Group A) accounted for 30.00%, while the uncontrolled group (Group
B) accounted for 70.00% of this error type. This comparison suggests that students in the control group had an improvement in fewer errors: less than half as many errors recorded for the students in the uncontrolled group.

The last error type in Figure 1 is “Incorrect Accumulator”. It was observed seven (7) times in this study. The control group (Group A) accounted for 85.71%, while the uncontrolled group (Group B) had 14.29%, of this error type. Comparatively, the control group had a lot more errors and therefore performed worse, relative to the uncontrolled group. Noticeably, in Figure 1, of all the error types studied, this error type appears to be least in improvement for students in the control group. In fact, the wide margin between the control and uncontrolled groups suggests the worst performance for students in the control group.

The results presented above shows that in eight (8) of the ten (10) types of errors studied, the novice students in the control group (Group A) made fewer Java programming errors, compared to their counterparts in the uncontrolled group (Group B). However, the data on two (2) particular error types showed no improvement for the control group (Group A). Unexpectedly, novice students in this group had a worse performance and more programming errors than their counterparts in the uncontrolled group (Group B). This negative change might be attributed to students being confused by the new under-development tutorial material. As shown in Table 1, the two particular error types were categorized as ‘logical’ errors.

Prior research suggests that logical errors are difficult to identify and to fix. For instance, Kolikant and Mussai [12] noted that novice programmers would consider a particular code to be correct, if it worked in general but not for all valid inputs. Logical errors mostly indicate the usage of an incorrect algorithm in a program [13]. In a study by [14], students perceived that a logical error was more difficult to detect and to fix than a syntax error. A logical error requires a mental focus, logical thinking and strategy in finding and fixing it because the compiler does not usually indicate if a logical error occurred, it is only known when the output is different from the expected result. The fact that logical errors are more difficult to detect and that they require better instructional and learning strategies beyond simple tutorials, the results as depicted in Table 1 are sorted by % and errors in Group A (the controlled group). Another interesting finding as in Table 2, shows that runtime errors come on top of the table while logical errors come at the bottom. This clearly suggests that the tutorials had a stronger significant on guiding students to avoid runtime errors than logical errors.

5. CONCLUSIONS

The main objective of this research is the detection and categorization of errors made by Meliksah University students taking first year programming courses. The findings from this study suggest that educating novice students, through Java programming tutorials, would help them in reducing, and even avoiding, diverse Java programming errors. However, the non-improvement observed in the novice student programming control experiment, in relation to logical errors, are consistent with previous research suggesting that logical errors are difficult to detect and fix, often requiring a mental focus, logical thinking and learning strategies. Suggestively, future studies should seek to develop instructional and learning strategies to assist novice student in overcoming the learning difficulties and programming errors associated with logical errors. Such strategies and tutorials may include providing novice students with a reference material document that contains a list of errors with detailed examples on how to avoid them. Furthermore, Java programming instructors could generate an error-rules template and documentation that can be integrated into the Java integrated development environment (IDE). This will serve as a plug-in tool for detecting errors and suggesting possible solutions to the particular error types, such as those identified and...
reported in this study.

REFERENCES