Syntax Free Software Application for Intro Computer Science Courses: Critique

Jordan Stuck & Dr. Nathaniel Lahn

Introduction

Through my experience as a tutor at the Harvey Center at Radford University, I have a noticed a lack of knowledge from students who have completed our intro courses here at Radford. The knowledge that they aren't retaining are key concepts that will help them succeed in future courses and in their career. These concepts are ideas such as: control flow, variables, functions, types, and generalization. Without the key concepts nailed down it is impossible for students to succeed in this field later in their academic and professional journey. I believe that the reason for this gap in knowledge is because students are coming in with little to no programming experience and they are trying to learn these key concepts at the same time as learning syntax for a programming language. The goal of this project was to separate the syntax of programming from the important ideas required to succeed by creating a Graphical User Interface (GUI) based software. This software would help the user visualize what code looks like without needing to type anything in. This project was intended to be added to course material to CS 101 in order to help the development of students' knowledge early in their academic career.

Context

The challenge that is being faced at Radford is that many students are coming out of the introduction computer science courses without the base knowledge required for success. The school has made great efforts to help fix this challenge through the development of new intro courses. The courses CS 118 and 119 were created in order to separate CS 120 into an easier and

slowed down two-course class. This helped in some ways but in many ways, from personal conversations with students, they felt it was too slow for them. I also noticed that students were more focused on memorizing syntax than attempting to understand the key concepts that were being taught. Other universities have had similar issues with retention of information and sought to remedy this through the creation of low-level introduction courses. At UIC (University of Illinois Chicago), after their introduction of this course they saw their success rate of students increase by 9% and retention rate by close to 22% (Sloan & Troy, 2008). Radford has sought a similar approach through the introduction of CS 101 to the curriculum. This course is intended to have a slow introduction for students with little to no programming experience. It is supposed to help build a foundational skillset for students to understand key concepts without harping on syntax of a programming language. I built this software application in order to help develop the students' understanding of these valuable ideas.

Strengths & Weaknesses

There were many strengths to this project, one of them being that it clearly created a separation between the syntax of java, and the logic of the code. It provided students with visualizations of code, as well as variables and functions (*Figure 1)*.

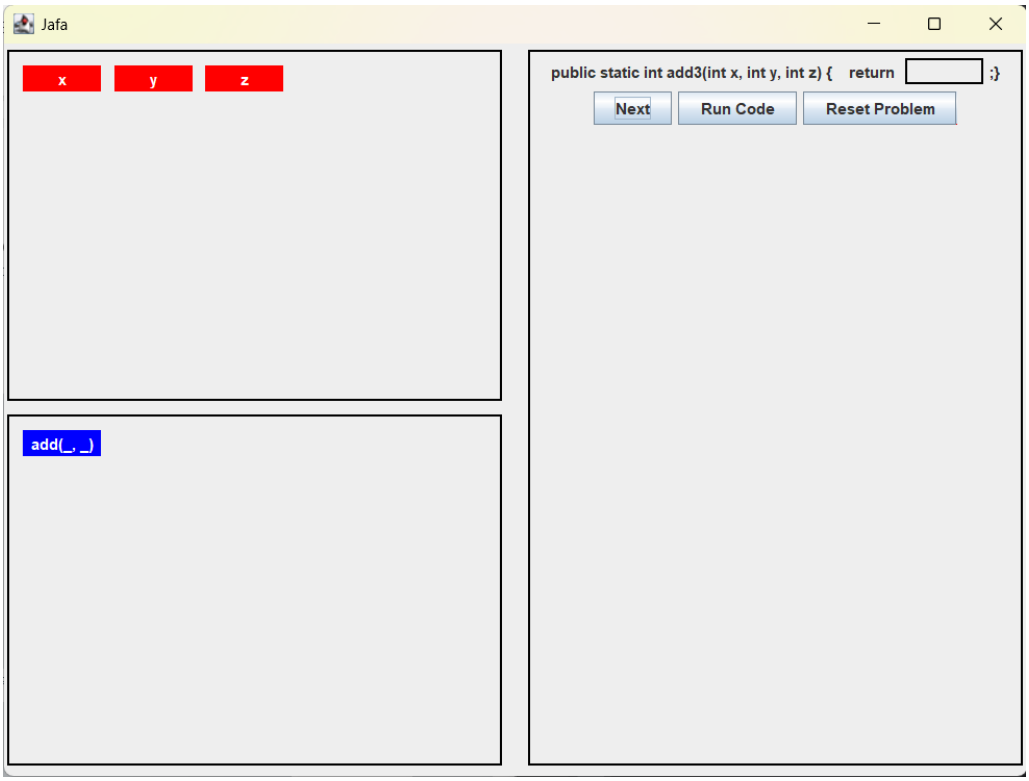

Figure 1: Layout of Software

With this application students no longer need to worry about how to write out the code but just need to understand the goal of the assignment. The students will then be able to reason their way through each problem without trying to multi-task and understand how to write code. Another strength of the project is that it is generalizable for the professors to use in any way they see fit. The professors can input their own problem sets into the program so that students can then complete the problems given by the instructor. There is no limit on the number of problems the program can have, so a professor could assign multiple problem sets focused on different key areas of programming. Finally, the last strength of the project would be video documentation for the professors to use. I provided a short video instructing the viewer on how to create their own problems and add it to the interface. This allowed it to be very user friendly for the professors to create the problems for their classes.

While my project had many strengths there were also a great deal of weaknesses and limitations regarding it. The first weakness I would like to point out is the structure of my underlying code. Dr. Lahn and I were focused on producing a presentable product over having fully reliable code. This was due to the tight time constraints on this project, as I was only able to work on it over the Spring 2025 semester. The code is quite messy and requires quite a lot of revision to make it cleaner to meet "industry" standards. The next weakness to address would be the usability of the software to the students. While it is easy for professors to create multiple problems at a time, it is probably difficult to understand the task at hand for each problem as there is no description shown for any of the problems, nor are there instructions on how to use the software. Finally, the last weakness of this project would be the structure of the code on the interface. The code does not directly emulate coding format as if the students were coding in an Integrated Development Environment (IDE). This makes the code that is displayed is hard to follow and therefore makes it more challenging for the students to complete their problems.

Next Steps

The weaknesses and limitations of the project of evident after completing this project. It is important to note the next potential steps that can be taken to improve the software, so that it can potentially be used for classes in the future. The first step would be to run through the backend code for the software and clean it up. This would include adding Java Doc comments on each of the functions and classes, and also adjusting the code it is not tightly coupled (tightly coupled simply put means when one part of the code shifts, many other changes are required in other locations in the code). Another step that could be taken would be to provide an instructions field in the problem class and display the instructions in a separate frame of the application. This

would help the students understand exactly what needs to be completed to solve each problem. This frame could also give basic instructions on how to use the application through dragging and dropping icons. This frame could also contain the buttons which navigate the code shown in the interface. The final step to develop this project further is to structure the code displayed in the interface to reflect that of an actual IDE. Overall, this project was successful and can be used in a class environment with a few refinements as mentioned above.

References

Sloan, R. H., Troy, P. (2008). Cs 0.5: a better approach to introductory computer science for majors. *ACM SIGCSE Bulletin*. 40(1). https://dl.acm.org/doi/abs/10.1145/1352322.1352230