JAFA: Java Formative Assessment – Development of a formative assessment tool to support learning of introductory computer programming

Dr. Kevan Buckley (K.A.Buckley@wlv.ac.uk)
Matthew Green
School of Computing and IT

Background and rationale

This project contributes to the long term goal of improving the Learning and Teaching of Computer Programming. This is a core topic in The School of Computing and Information Technology (SCIT) that the majority of students must study, but historically pass rates of around 60% at first attempt have been common. This has an obvious impact on progression and retention.

The work follows on from a successful project which concentrated on the assessment of Numeracy, Mathematics and Statistics (Thelwall, 1998), the resulting product of which has been used extensively in the School, but has now reached retirement only because of technical reasons. The primary focus of this stage of the project is to develop a formative assessment tool that is conceptually similar in some respects to the earlier system, but is specifically tailored to Programming and uses up to date technology to maximise effectiveness. The product will be tested with a large student cohort to ensure that it functions correctly and is fit for purpose, with the intention that a follow-on project will embed the resulting product in the curriculum.

The outcomes will be twofold. Firstly, it will enable a move towards a learning style for which there is anecdotal evidence that indicates students like and overcomes issues related to the transition from their prior learning styles. Secondly, it will enable research into the behaviour of students learning to program, and will lead to enhanced provision of support for learners.

Introductory Computer Programming

Computer Programming is a core discipline that all Computer Science graduates must be proficient in. It often causes students difficulty because it is conceptually demanding, drawing many abstractions from mathematics as well as involving problem solving and creativity skills. It is easy for students to fall behind quickly as the concepts build incrementally, thus if a student falters on one topic it impedes their access to later topics. In other subject areas, particularly at lower levels, students may progress by adopting little more than a surface learning approach involving temporarily memorising of facts that are not retained post assessment. However, Programming involves learning a set of concepts, a formalised language for expressing ideas and instructing the computer, as well as being able to use development tools and processes. Students must be able to practically apply knowledge and skills in a problem solving environment, which requires higher level skills. To complicate matters, computer languages bear little resemblance to human written languages, are riddled with peculiar punctuation and the unforgiving nature of computers requires a high level of precision resulting in students becoming easily frustrated. An example of a

program, taken from an assessment held in teaching week 6, is shown in Appendix 1. Any spelling errors or misplaced punctuation will force the program to be rejected by the computer. Syntax errors are cited by Weinberg (1971) as a key factor in the dampening of enthusiasm of beginner programmers. Even if the computer does accept the syntax of the program, there is no guarantee that the program logic is sound and appropriate for solving the given problem.

The approach adopted at Wolverhampton aims to mirror aspects of learning a human language; firstly students will acquire a reading knowledge of programs by learning the keywords and syntax that form building blocks, then tracing the behaviour of programs and being able to state their outcomes. Secondly, students move on to develop strategies for solving new problems, formulating algorithms, then preparing programs to communicate with the computer to instruct it how to solve the problems. Both the first aspect and the second contain complex concepts that are difficult to understand all at once, thus this project aims to decouple key concepts and enable students to concentrate on them almost in isolation.

In a parallel study[1] involving a subgroup of the same cohort, student responses at focus group meetings indicated that some students were reluctant to read recommended literature. Those students perceived that a task involving reading 20 pages from the (carefully chosen, "student friendly") course text over the period of a week posed a great challenge. This may be a result of the cultural shift that has been reported on by Prensky(2001) and the methods of learning experienced in schools and further education. As books still remain the primary dissemination medium for the majority of mankind's knowledge, guided reading must remain an important part of undergraduate study, thus students' reading skills must be nurtured. One use of the outputs of this project will be to support the decomposition of larger reading tasks into smaller, "bite size" chunks, separated by formative activities. One aim is to encourage reflection and re-reading if feedback indicates poor understanding or relay a sense of reward if the feedback shows that the material has been understood.

The Innovation

The product is called Jafa (Java Formative Assessment) and consists of 2 components. Student access the system through a Java applet, which runs inside a Web browser. Students find a link to the applet on Wolf, the University's virtual learning environment and when loaded gives the impression that they are still inside Wolf. Elsewhere on the Internet a specialised server controls access to the system, records data and issues questions and answers to the student's applet. Figure 1 shows a typical screen shot of the students' main view. A tree structure allows topics to be grouped. Different icons show which tasks have been successfully completed, unsuccessfully completed and not attempted. The frame towards the bottom gives the student information about the selected question, in this case a question that has been successfully completed and the time taken for completion (included to add an element of challenge and self satisfaction similar to highscores tables recommended to encourage repeated play in computer games (Rouse, 2001)).

The learning outcome of being able to read and understand a program is assessed in a traditional examination environment using a set of small programs that students must analyse and state the outcomes of. These start off very small, requiring only a single concept to be understood and

[1] TLRP/EPSRC funded project : Learning and Teaching for Social Diversity and Difference

escalate to larger programs in which multiple concepts are entwined, thus enabling the different levels of understanding to be differentiated. To prepare for this, Jafa allows students to be repeatedly presented with questions similar to those they will be assessed on, but each time the question will be slightly different. This is not achieved by the usual method of using a bank of questions (Penfold, 2001), which are time consuming to generate and inherently limited leading to questions being repeated. This software uses sophisticated algorithms to automatically generate questions and answers that meet the tutor's specification, i.e. the tutor defines the template for a group of questions and a set of parameters to constrain variability. With this question type a small number of questions are grouped together to form an exercise. Students are given questions (Figure 2) one at a time and are asked for their solutions to be typed at the keyboard, until the exercise is complete. Feedback is then given for the whole of the exercise. With Programming there is no need for the system to check against a variety of valid responses, as is commonly done with free-text questions, because precise answers are expected. Feedback is brief because the intention is for students to use Jeliot (Ben-Bassat Levy, 2003) to help them understand programs. Jeliot is a program visualisation application that animates how a program is interpreted. Code for incorrectly answered questions can be pasted into Jeliot for explanation.

As well as generating questions, testing and giving feedback the system records all transactions with users. This includes times of issue of questions and return of answers along with the location on the Internet of the user. This will allow detailed analysis of student participation and progress.

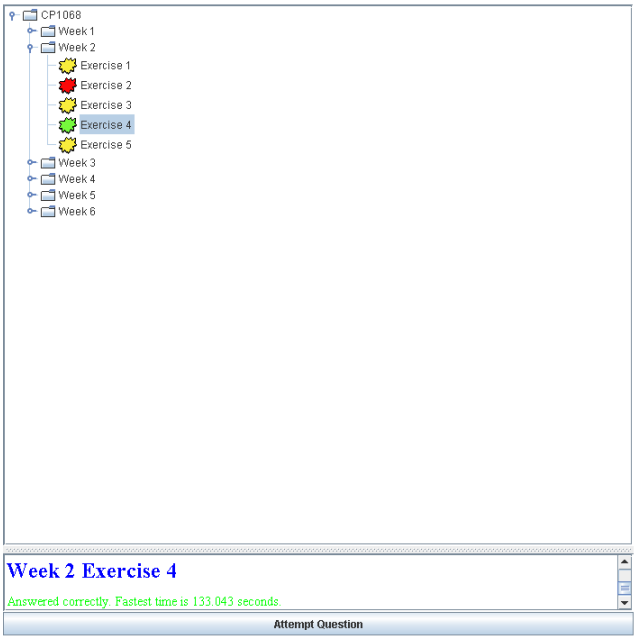

Figure 1 - Exercise selection screen

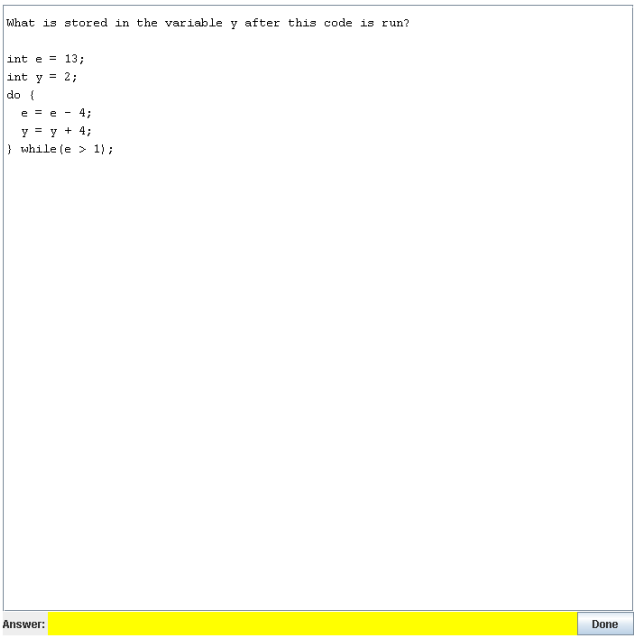

Figure 2 - Computer generated free text questions

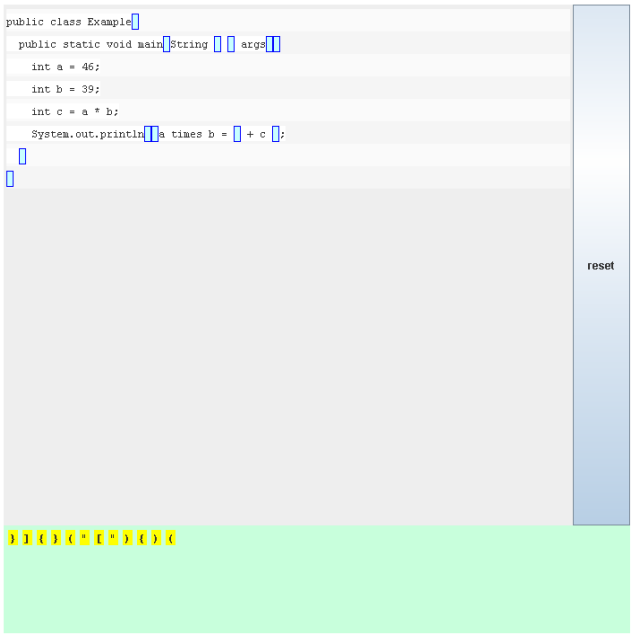

Figure 3 - 'Micro' code magnets problem

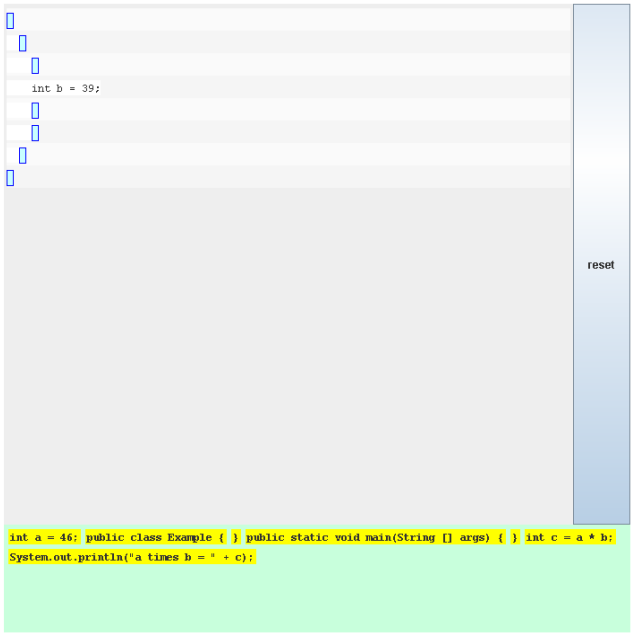

Figure 4 - 'Macro' code magnets problem

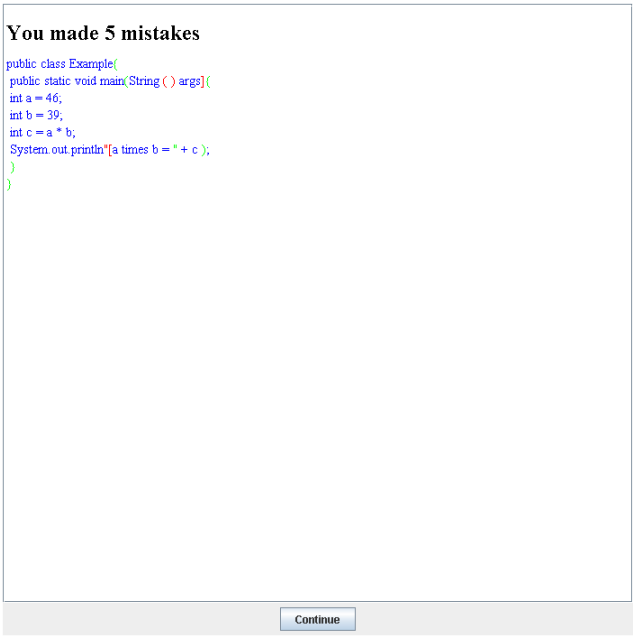

Figure 5 - Feedback from code magnets

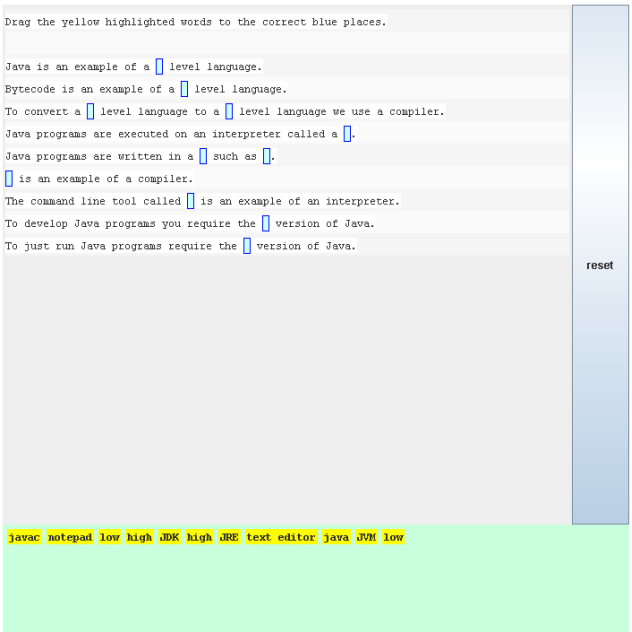

Figure 6 - Missing words exercise

To support the learning of program writing, Jafa provides Code Magnets questions. This is a similar idea to fridge magnets where each one contains a snippet of program code and when arranged correctly forms a working program. The idea stems from exercises for the O'Reilly and Associates Head First series of books (Sierra, 2005) and the idea has been experimented with using

sets made from cardboard previously on the module. A Code Magnets computer simulation is novel to this project.

Experience using Code Magnets in class has resulted in the identification of two distinct ways of using them, termed micro and macro approaches here. Figure 3 shows an example of the micro approach, where most of the code is in place, except for certain elements of punctuation. The places where punctuation is missing are shown as boxes in the program. Magnets containing the required punctuation can be dragged from the bottom of the screen and dropped into the appropriate boxes. By doing these micro exercises, based on the placement of small tokens, students will reinforce their knowledge of program syntax, but without the complication of having to think about designing an algorithm to solve the problem. Figure 4 shows the macro approach, where the student does not have to think about micro syntax issues and is left to think about the overall structure of the program, its logic and algorithm design. The two approaches complement each other and prepare students for the goal of writing programs from scratch, which combines the micro and macro approach, but with the added complication there is not a restricted choice of magnets to use, but students may draw from the experience of magnets to remember how the building blocks of a program are formed.

A third use of the Code Magnets application was only realised when the aforementioned focus groups identified students' problems managing their guided reading. It was thought that Code Magnets could be used to provide Missing Words exercises to break up and provide feedback on reading task in an interesting way. Figure 6 shows an exercise used to support a reading task.

Results

The prototype system consisting only of the Code Magnets component was made available for three weeks during the module. The aims of the pilot test were twofold. Firstly, to prove the core of the system worked, in particular the communication between clients and the server, authentication and data logging. Secondly, to find out if students liked the idea and identify any refinements. In hindsight, it has been recognised that students' opinions should have been elicited via structured survey, however only informal tutor-tutee conversations took place. Feedback from students was only positive and their enthusiasm to use the system was exhibited by the fact that there were in excess of 2000 logins recorded.

As the user interface for the randomised, free-text questions had not been integrated into the client applet in time, the question/answer generator was used to produce worksheets for students. 5 worksheets, each consisting of 85 questions were made available through Wolf. Students requiring help were guided through worksheet 1, given help where required with worksheet 2 and were left to do the other sheets on their own. Questions on all versions were generated by the same template, e.g. question 10 on the first sheet was generated from the same template as question 10 on the other worksheets, thus students could work with tutors on solving a particular question, then when understood had more variants to practise on. Students that took advantage of these exercises, said they had greatly benefited from them. The complete prototype including both question types was trialled at a revision summer school, where it was shown to operate to specification in a student environment.

Evaluation

The system has been shown to operate in a student environment and does not suffer from any loading problems with a large cohort. Its effectiveness as a learning tool has yet to be evaluated, as most project resources were invested in software development and subject content developed was minimal, thus could only be used for a limited part of the module.

Several human-computer interaction (HCI) problems have been identified, such as small fonts on Code Magnets and the use of colour coded icons on the question selection screen and colour coded text on feedback screens that are not suitable for all users. These will be ironed out for the next release and student opinion of HCI issues will form part of a user survey.

Students need to learn the syntax and structure of a programming language and be able to apply it to solve problems. Jafa provides a means to this end, but will certainly fail to capture students' enthusiasm if used in isolation. The *drill* type exercises that Jafa provides make a valuable contribution, but they must be used in conjunction with more appealing activities that contain the elements of Keller's model of motivation (Keller, 1987): attention, relevance, confidence and satisfaction.

One limitation is that the tutor writing the questions has to include a small amount of program code to perform the question-answer mapping. This is not a problem when used in the current scenario as it is assumed that tutors teaching programming can program themselves, but it does restrict the transfer to other subject domains, principally mathematics and statistics, where the approach seems applicable. A prototype system, which allows questions to be defined in an XML (eXtensible Markup Language) based language is currently being investigated, and may ultimately lead to the development of a support application for authoring questions, thus enabling wider use.

Future Developments

In Semester 1 2007/8 Jafa will form an important part of first year students' study in SCIT. Firstly, it will be fully embedded in the Fundamentals of Programming module in the way it was originally intended. It will also make a significant contribution to pastoral support and counselling. In a school-wide drive to improve retention, personal tutors will be informed regularly of the progress of their students. The data from Jafa will be combined with data from in-class voting systems, wiki usage, Wolf tracking and traditional attendance records to form student profiles for personal tutors to use. Most importantly the information will inform the individual meetings that personal tutors have with their tutees at the end of teaching week 4. The new courses that will be delivered have an "escape route" to allow students who would be better suited to less technical modules to progress on a different course. This has been enabled by having a common first 4 weeks for the technical and non-technical modules, which subsequently diverge to deliver different learning outcomes.

Further research will include the analysis of Jafa records combined with student achievement to identify any correlations between student behaviour patterns, formative assessment results and module outcomes with a view to building predictive models. There is a belief that attendance and success are closely coupled. This research will quantify the relationship and extend it to focus on

participation as opposed to attendance. Many students now have busy lifestyles, often working long hours to support their studies and claiming to do University work at peculiar times. This study will inform us whether full participation is actually taking place in non-traditional study patterns and may estimate its effectiveness compared to students able to follow more usual study patterns.

With a view to maximise student participation, the theory learnt via Jafa will be applied in student-motivating activities such as programming their own mobile phones and a team of Java-controlled robots.

References

Ben-Bassat Levy, R., Ben-Ari, M., Uronen, P.A. (2003) The Jeliot 2000 program animation system, *Computers & Education*, 40(1), 2003, 1-15

Keller, J.M. (1987) Strategies for stimulating the motivation to learn, *Performance & Instruction*, 26(8), 1-7.

Penfold, B. (2001), Automatic Marked Summative Assessment *IN Learning and Teaching Projects 2000/1* (Eds. Moore, I. *et al.*), University of Wolverhampton Press, ISBN 0-9542116-0-x

Prensky, M. (2001) Prensky Digital Natives, Digital Immigrants. *On the Horizon*, NCB University Press, 9(5)

Rouse, R. (2001) *Games Design: Theory and Practice*. Wordware Publishing, ISBN 1-55622-735-3

Sierra, K., Bates B. (2005) *Head First Java.* O' Reilly, ISBN 0-59600-920-8

Thelwall, M. (1998) A unique style of computer assisted assessment. *Alt-J* 6(2) 49-57.

Weinberg, G.M. (1971) *The Psychology of Computer Programming*. Van Nostrand Reinhold

Appendix - Example Java Program from Semester 1 Week 6

```
public class Solution {
  public static void main(String[] args) {
    int k;
    int cubed, squared;

    for (k = 0; k < 5; k++) {
      if (k > 0) {
        System.out.print(k + ".");
        squared = k * k;
        cubed = squared * k;
        if (squared >= 10) {
          System.out.print(squared);
        } else {
          System.out.print("0" + squared);
        }
      } else {
        System.out.println("-------");
      }
    }
  }
}
```