

# Strategies for Improving a Java-based, First Year Programming Course

Michael Blumenstein

School of Information Technology, Griffith University-Gold Coast

PMB 50, Gold Coast Mail Centre, QLD 9726, Australia

E-mail: [m.blumenstein@mailbox.gu.edu.au](mailto:m.blumenstein@mailbox.gu.edu.au)

## Abstract

*This paper describes the evolution of a first year Java course at Griffith University - Gold Coast since Semester 1, 2000 to the present day. The course was updated to emphasise program design and to implement and evaluate an "objects-as-needed" approach to first year programming. A number of strategies were tested to increase consistency amongst teaching staff, improve delivery of course resources, successfully cater to a wide variety of students and to enhance the learning experience in general. The success of the revised course has been measured by evaluating student feedback and performance. Currently, a focus group-based strategy of evaluation is being adopted to determine students' attitudes to the most recently implemented changes.*

## 1. Introduction

In the last few years, the literature has been inundated with papers describing the difficult transition from the use of procedural to object-oriented languages in the teaching curriculum [3]. Many institutions have favoured the adoption of Java as their language of choice for their first year programming course. At this stage, some universities are still pre-occupied with the question: "Not whether Java but how Java" [14]. There are a number of contentious, and some say unresolved issues that have plagued the minds of educators across the globe when dealing with the above question. Some of these issues have been tackled in the last two years whilst revising the first year Java course at Griffith University.

In particular, one of the major issues that presented itself during the revision of the Programming 1 course was the choice of methodology for teaching object-oriented programming. This paper discusses the dramatic changes and resulting experience obtained in revising the Programming 1 course. Particular attention is paid to the areas of: 1) Teaching resources, 2) Delivery of teaching materials, 3) Teaching methodology and 4) Assessment.

The remainder of this paper is divided into 4 Sections. Section 2 addresses the challenges that were considered whilst evaluating the Programming 1 course. Section 3

deals with the various strategies that were used to modify the course and Section 4 gives a description of the effectiveness of the newly revised curriculum. Finally, Section 5 offers conclusions and discusses future developments relating to Programming 1.

## 2. Background of the Course and Challenges Faced

This section addresses challenges that were tackled whilst revising the Programming 1 curriculum. In particular, factors that were specific to the Gold Coast campus will be addressed as well as general issues that are faced by all educators dealing with Java as a first language.

### 2.1 Campus Demographics

The first year programming course at the Gold Coast campus attracts a wide variety of students from different disciplines and backgrounds. For Multimedia and Information Technology students, Programming 1 is a core course in the first year of their degree. In Semester 1, the majority of students (80-90%) are Information Technology students. However, in Semester 2, the Multimedia students dominate the Programming 1 course's demographics. In both semesters the remainder of students enrolled are from other disciplines.

Whilst reviewing the course, the previous convenor indicated that there was a peculiar trend with regards to the performance of students over the two semesters. Specifically, the students enrolled in Semester 1 usually outperformed the students in Semester 2 in terms of academic achievement. It seemed that this trend might have correlated with the fact that the majority of students enrolled in Semester 2 were Multimedia students. Through their own experiences Allen and Bluff [1] note that disparities between these two groups arise due to the different expectations that each has. Specifically they mention that many Multimedia students are led to believe that their degree will be centred on more visual aspects of interface and application design rather than the more technical aspects of application development.

## 2.2 General Challenges

A more common issue that was addressed included whether to teach Java "objects first" or to continue along the lines that it had been taught in 1999 i.e. "structured programming-first" in a console-based environment.

Finally, an on-going challenge faced by the previous course convenor was to determine the best assessment strategies for the Java course. There were many problems with the assessment pieces that were set in 1999 including the sheer number of deliverables and a lack of individual assessment under "exam conditions".

## 3. Revision of the P1 Course

This section details the evolution of the Programming 1 course with particular attention to four areas: 1) Teaching Resources, 2) Delivery of Teaching Materials, 3) Tutor Support and Communication and finally 4) Assessment.

### 3.1 Teaching Resources

**3.1.1 Objects "gently" and Textbook Choice.** Prior to revising the Java course, a brand-new textbook [8] was considered as a replacement for the one used previously. Upon reviewing the text, it seemed that the question of objects was handled in a "gentle" and hence favourable manner. More specifically, it did not take the stance of either of the radical methods i.e. "objects-first" [6] or "structured programming-first" [4]. Instead, the text took the approach of introducing object-oriented concepts "as needed" [8].

The textbook was also attractive for another reason. It facilitated a shift away from entirely console-based applications and presented an opportunity to embrace GUI ones. However, rather than plaguing the students with the complexities of AWT, they were able to make use of the BreezyGUI package that was included with the textbook. The authors of the textbook, along with a number of other educators, are convinced that students are far keener to learn programming when they are able to produce applications with easy to build interfaces [7], [10]. It was also hypothesized that it would be appropriate for teaching the Multimedia students in Semester 2, as they would be able to develop applications more relevant to their field of interest.

**3.1.2 Console-based Applications.** Although BreezyGUI provided an excellent strategy for motivating students from most disciplines to commence and enjoy programming, it was felt that students would benefit from obtaining a more balanced view of programming in Java. It was therefore decided that students be introduced to console-based programs in the first few weeks of the course. The problems associated with Java and its

complicated I/O operations are well known [2], [11]. The solutions to teaching these difficulties vary, however the method chosen for Programming 1 was to share resources from Griffith University's Nathan campus. Specifically, custom-built classes for input and output were adopted: SimpleReader() and SimpleWriter() [13]. Classes of this nature have been adopted by a number of educators [2], and have allowed students to focus on the task of performing input and output rather than dealing with the complexities of Java's stream classes.

**3.1.3 Design Paradigm.** Upon commencing the course evaluation process, it was evident that emphasis on application design had not been prevalent between 1998 and 1999. This was an area of concern and would need to be investigated. As the proposed course structure would at times follow the textbook closely, the design paradigm would have to match this structure. It was therefore decided to adopt the Structured Design Chart (SDC) paradigm for teaching design. SDCs are based on Nassi-Schneiderman diagrams [12] and have one distinct advantage over other design techniques: not only do they provide the student with the final algorithm, but they also display the steps that were taken to get it.

### 3.2 Delivery of Teaching Materials

The method of material delivery chosen went along similar lines to previous semesters. There were four hours of contact time per week including one two-hour lecture and a single two-hour computer laboratory. It was felt that the class size, although reasonably large, could still benefit from material delivery in a lecture situation. Outside of these times, students were able to attend consultation times with their tutors or the course convenor.

To complement the contact time described above, the Programming 1 webpage and the School network became the centre pieces of "after-hours" material delivery. In previous years, little or no emphasis was placed on the webpage as a teaching aid. It was the task of the convenor to alter this state of affairs so that the focus could be reassigned towards that of "flexible delivery". All lecture material, tutorial exercises, assessment items, hints, course outline, staff contact details and announcements were hence placed on an easy to navigate, rapidly accessible page.

### 3.3 Tutor Support and Communication

Tutor instruction and communication was of particular importance with regards to the student numbers. It is for this reason that a close rapport was maintained (in the form of fortnightly meetings) between the convenor and each tutor to ensure consistency and quality with regards

to delivery of weekly teaching material in computer laboratory time.

### 3.4 Assessment

Student assessment prior to 2000 was in the form of 11 short, take home laboratory modules and one major project. Upon reviewing the nature of these assessment items more closely, it was clear that this method of student performance evaluation had many disadvantages. Firstly, the sheer number of assessment items that needed to be collected from students over the course of a semester was overwhelming. As a result of this excessive assessment load, many were lost, and it was then difficult to track them down at a later time. The assessment load also increased the marking load for the convenor and the tutors. Finally, due to the fact that the assessment items were not to be completed under "exam conditions", students had the benefit of working with others, using code from the textbook or notes and finally the possibility of plagiarism.

Due to the problems discussed above, the assessment methodology for the course was investigated. Although it was agreed that the concept of frequently assessing the students was beneficial, it was necessary to incorporate assessment pieces that could evaluate students' individual performance. This necessity led to the introduction of a mid-semester exam and a final exam into the course in addition to practical assessment such as laboratory assessments and a project. It was later found that this assessment structure was challenging for the students and seemed to evaluate their performance well on all topics in the course. However as will be seen in later sections, it would not remain static throughout 2000 and 2001.

## 4. Outcomes and Discussion

The following sections will relate some of the experience obtained from teaching students at the School of Information Technology over four semesters. The subsections presented below testify to the fact that the course structure agreed with certain student groups but was substantially more difficult for others. To dynamically address the needs of students with different backgrounds, minor changes to the course and assessment structure were made to increase student learning and to evaluate student performance more effectively.

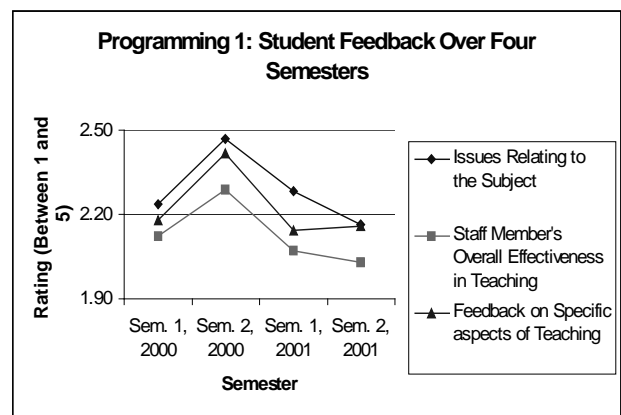
### 4.1 Programming 1: Semester 1 & 2, 2000

Initially, the delivery of the course in Semester 1 proved to be quite challenging due to the added novelty and embellishments already discussed. Regardless, the new course structure proved to be reasonably successful. The main evidence for the course's success was sourced from student performance and student feedback. Both

were satisfactory in Semester 1 as may be seen in Table 1 & Figure 1.

**Table 1. Profile of Grades (Gr.) for Programming 1 from Semester 1, 2000 up to and including Semester 2, 2001. The failure rate in brackets includes those students that did not submit the majority of assessment items.**

Semester 1, 2000		Semester 2, 2000		Semester 1, 2001		Semester 2, 2001	
Gr.	%	Gr.	%	Gr.	%	Gr.	%
HD	11.8	HD	8.11	HD	18.1	HD	5.5
D	12.2	D	8.11	D	13.2	D	15.1
C	15.1	C	10.4	C	20.6	C	12.4
P	20.4	P	16.7	P	13.2	P	21.1
F	14.7 (28.2)	F	26.6 (37.8)	F	12.8 (23.5)	F	12.39 (26.1)



**Figure 1. Programming 1: Student Feedback. The y-axis represents average student ratings on a scale between 1 and 5 (1 signifies 'Outstanding' and 5 signifies 'Very Poor')**

**4.1.1 Student Performance in Semester 1.** It might be useful to look at the profile of grades obtained by students at the end of Semester 1. Nearly 40% of students obtained a grade of a Credit or above. The failure rate obtained (whilst including only those students that submitted a majority/all of the assessment items) was an acceptable level: 14.69%. Unfortunately, the failure rate for students that only submitted 1-3 pieces of assessment out of a possible 6 was higher: 28.2%. This demonstrates the fact that some students did not withdraw from the course at the appropriate time and did not complete their studies. Others simply did not attend lectures and laboratories or had fee problems.

**4.1.2 Student Feedback: Implications for Semester 2.** At the conclusion of Semester 1, there were a number of issues brought up by students in their written feedback that was helpful for the upcoming semester. The main issues were with regards to assessment. It was generally felt that

the number of assessment items was excessive. It was for this reason that the curriculum was altered in Semester 2, 2000 so that one set of lab assessments (practical programming problems) was removed to enable students to focus on their end of semester projects.

**4.1.3 Evaluation of Student Performance in Semester 2.** The minor improvements in the organisation of the course, and many positive written statements and feedback from students and tutors in Semester 1 attested to the general level of satisfaction from students with respect to teaching and learning in the course. To a large extent, the profiles of student grades in Semester 2 seemed to be similar to those in the previous semester (aside from a small increase in the number of failures).

An investigation of student demographics and performance found that a high proportion of students that did not perform well in the course were undertaking a Multimedia or Commercial computing (Business) degree. Aside from the above evidence, written student feedback correlated well with the poor performance of the groups mentioned. Specifically, a number of comments from Multimedia students were evaluated and reflected upon. The comments were all along the same lines, focussing on the fact that "...we (Multimedia students) will not need programming for our prospective jobs and therefore do not see the point of doing it". This particular view seems odd as many Multimedia students will be required to develop webpages that will in turn require knowledge of Java and applets. Nevertheless, it seems that this attitude is not an isolated view and it may be the reason for a lack of effort and enthusiasm on the part of certain student groups undertaking first year programming.

## **4.2 Programming 1: Semester 1 & 2, 2001**

**4.2.1 Student Motivation and the Importance of Practicing Programming.** With regards to motivation, most tutors, found that in Semester 2, 2000 students were not attending laboratories frequently and in many cases were not completing their assigned programming exercises. As an initiative to tackle the above problems, it was decided that students would be asked to submit their weekly exercises for marks. Students would therefore be "obliged" to, at the very least, attempt the weekly exercises and could work at a constant pace throughout the semester. This is supported by Duke *et al* [3] who agree that "...it is only through practice that a computer language, like any language, can be mastered". As may be seen in Table 1, the use of these exercises seemed to have a positive affect on student performance in Semesters 1 & 2, 2001. To complement the above evidence, Figure 1 displays all time highs in positive student feedback for both Semesters 1 and 2, 2001.

**4.2.2 Issues Relating to Semester 2, 2001.** In Semester 2, a different approach for assessing students' practical programming performance was tested. Rather than completing the lab assessments as a solely take-home exercise, it was decided that they be divided into two parts. Part one would be an in-class assessment to take place in laboratory time and the second part would remain a take-home component to complement the former. This assessment piece was executed very successfully with little or no difficulties across all labs.

## **4.3 Programming 1: Semester 1, 2002**

**4.3.1 Teaching Objects Early.** As may be seen from the student performance and subject evaluations from 2000 and 2001, the newly instituted Programming 1 curriculum proved to be quite successful. However, upon reflection and a thorough evaluation of students' work, it was noted that there were a large number of students who were still finding the concepts of object-orientation difficult to master. The objects-gently approach, although successful, had its drawbacks. Specifically, students seemed to struggle at project time whilst attempting to master user-defined classes.

It was due to this observation that the Programming 1 course was guided through an extra evolutionary step. In Semester 1 2002, the approach to teaching objects became less "gentle" and slightly more inclined to the "objects-first" approach [6]. At approximately the same time, the 2<sup>nd</sup> edition of the Lambert & Osborne text [9] was released. It provided good support for this approach and hence it was adopted for Semester 1, 2002.

With the advent of this less "gentle" approach to teaching objects, the following were the main changes that were implemented: 1) Basic concepts and terminology of Object-Oriented programming were dealt with in lecture 1, 2) Object instantiation and message passing were cursorily covered in lecture 3, 3) The application object and other O-O issues were covered in more detail in lecture 4.

**4.3.2 Focus Groups and Programming 1.** With the advent of the course modifications described in Section 4.3.1, one of the initiatives considered important was to undertake an evaluation of student opinions and attitudes towards the course. The feedback obtained, would provide a reasonable idea of how the new, less "gentle" approach to teaching objects was being received by students.

The methodology chosen for evaluating the students' attitudes was: The "focus group" approach [5]. In this approach, groups of students representative of the population are chosen to respond (in written form) and reflect on issues pertaining to the course at various intervals throughout the semester. The first set of focus group "meetings" were convened during laboratory time in

week 6 of the semester. This coincided with the completion of lectures dealing with the more assertive object-related material. Three labs consisting of 20 students each were randomly chosen. The second "meeting" is to take place in Week 12 to determine how students are coping with advanced O-O concepts after being exposed to objects early on in the semester.

The student demographics in each lab suggested an even spread of IT students and those from other disciplines. The following questions were given to students:

1. What are your current feelings towards the course?
2. What do you like about the course so far?
3. What do you dislike about the course?
4. What would you change about the course?

From the preliminary study, the feedback is very positive. As would be expected, students that are not undertaking an IT program have been finding it difficult, however they have expressed their enjoyment. On the other hand nearly all IT-based students are finding the course challenging and informative.

As this is only the preliminary stage of the focus group study, conclusions will be deferred until the end of the course. However, from the evidence sourced, it may be observed that in general, students find the less "gentle" approach to learning objects challenging in the first few weeks.

## 5. Conclusions and the Future of P1

This paper has described various challenges that were faced prior to and during the re-design of the Programming 1 course at Griffith University. Following the implementation of various changes to the course, the learning outcomes of students along with student feedback were measured over four semesters. It was shown that the learning outcomes for Semester 2, 2001 were the highest of all four semesters. This may be attributed to initiatives that were adopted to continually monitor student progress by encouraging the completion of weekly exercises. The amount of positive student feedback was at its highest in Semesters 1 and 2, 2001 and had increased significantly from the earlier semesters of the course.

Another adjustment to the course curriculum has been implemented in Semester 1, 2002. In a preliminary focus group study, written feedback suggests that students are coping well with the new "objects early" approach to programming. Further focus group meetings will be held again at the conclusion of the semester. In future, it may also be necessary to further investigate the applicability of Programming 1 to Multimedia students. Student feedback

and performance (based solely on Semester 2, 2000 and 2001) suggests that the course might benefit from further modifications to incorporate topics that are relevant to Multimedia students as well as those of other disciplines.

## 6. References

- [1] Allen, R. K. and Bluff, K., *Jumping into Java: Object-Oriented Software Development for the Masses*, ACSE '98, (1998), 165-172.
- [2] Clark, D. and MacNish, C., *Java as a teaching language-opportunities, pitfalls and solutions*, ACSE '98, (1998), 173-179.
- [3] Duke, R., Salzman, E., Burmeister, J., Poon, J., Murray, L., *Teaching Programming to Beginners-choosing the language is just the first step*, ACSE '00, (2000), 79-86.
- [4] Gibbons, J., *Structured programming in Java*, CTI Computing - Monitor 9, *Java in the Computing Curriculum II*, (Spring 1998), <http://www.ulst.ac.uk/cticomp/gibbons.html> (downloaded 22/10/01).
- [5] Goodrum, D., Hackling, M. and Rennie, L., *The status and quality of teaching and learning of science in Australian schools*, A Research Report prepared for the Department of Education, Training and Youth Affairs, (2001).
- [6] Kölling, M. and Rosenberg, J., *Guidelines for Teaching Object Orientation with Java*, ITiCSE 2001, (2001), 33-36.
- [7] Lambert, K., and Osborne, M., *Easy GUIs with Java in the Computer Science Curriculum*, 30th Annual SIGCSE Technical Symposium, New Orleans, (March 1999).
- [8] Lambert, K. A., Osborne, M., *JAVA: A Framework for Programming and Problem Solving* (1st Edition), Brooks/Cole Publishing Company, Pacific Grove CA, 1999.
- [9] Lambert, K. A., Osborne, M., *JAVA: A Framework for Programming and Problem Solving* (2nd Edition), Brooks/Cole Publishing Company, Pacific Grove CA, 2002.
- [10] Martin, F. and Williams, P., *Java: teaching and learning in large classes within a modular scheme*, CTI Computing - Monitor 9, *Java in the Computing Curriculum II*, (Spring 1998), <http://www.ulst.ac.uk/cticomp/fmartin.html> (downloaded 22/10/01).
- [11] Martin, P., *Java, the good, the bad and the ugly*, CTI Computing - Monitor 9, *Java in the Computing Curriculum II*, (Spring 1998), <http://www.ulst.ac.uk/cticomp/pmartin.html> (downloaded 22/10/01).
- [12] Robertson, L. A., *Simple Program Design* (2nd Edition), Thomas Nelson, Australia, 1993.
- [13] Rock, A., *SimpleReader and SimpleWriter I/O package (abr.srw)*, (2001), <http://www.cit.gu.edu.au/~arock/p1.01.2/> (downloaded 29/04/02).
- [14] Wallace, C., Martin, P. and Lang, B., *Not Whether Java but how Java*, CTI Computing - Monitor 8, *Java in the Computing Curriculum*, (1997), <http://www.ulst.ac.uk/cticomp/not.html> (downloaded 22/10/01).