ABSTRACT

Teaching programming to newbies is still an ongoing challenge. Strongly worded, either students can program when they enter the university or they never learn it. The challenge is to address as many students as possible in such a way that they get an understanding of programming and become able to complete the course successfully.

In contrary to traditional structured programming the Objects-First approach starts with classes and objects and introduces data types and programming statements whenever they are necessary to define the structure or behaviour of objects. The author has used the Objects-First approach for more than ten years. Almost the same content has been used all over the years but the way of teaching has changed. For the last two years the author has used elements of just-in-time teaching. The paper reviews the different teaching approaches based on the exams results. The author argues that teaching style has a minor influence. The social atmosphere among the students is a more crucial factor.

Keywords: teaching programming, just-in-time teaching, object-oriented programming

INTRODUCTION

How can we teach programming to new students? This question has been answered in many ways but still the question is open. The question itself has several perspectives: What is the objective to learn programming? Who are the participants in the course and do they have a background in programming? What programming paradigm or programming language is appropriate depending on the previous questions?

Introductory courses into programming have run now for decades at thousands of universities worldwide. So, why is it still a challenge?

The problem is that not all students get enough programming skills to pass the exam. It is not an exams problem only, programing skills are usually necessary for other subjects in a degree course that starts with an introduction into programming. We could not worry about students who fail the exams but still we want to get as many students as possible.

The author has run introductory courses into programming for more than 35 years using different languages. Discussions with colleagues on various conferences have underlined that teaching programming is an ongoing challenge for many lecturers.

The author has followed the object-first approached as outlined in (Barnes, et al., 2016) since 2005. In this time the author followed three slightly different strategies in teaching the same content. Now, it can be asked if the different approaches have an influence on the academic performance of the students. For two years, elements of just-in-time teaching have been included that give students a
feedback in a short time. This is a special question if these approaches result in a better performance of students. The exams results are used to answer these questions. The result is not very promising. The author comes to the conclusion that teaching methods have a rather small impact on the results. The paper is structured as follows: chapter 2 introduces the introductory course while chapter 3 focusses on the different types of teaching used. Chapter 4 analyses the exams results that build the basis for further conclusions given at the end of the paper.

**THE MODULE ON INTRODUCTION INTO PROGRAMMING**

The module ‘Introduction into Programming’ is a mandatory module for all students in the undergraduate course Business Informatics. Business Informatics\(^1\) is a discipline and a degree programme focussing on the application of Informatics (Computer Science) in business related areas, especially in business administration. The Bachelor programme in Business Informatics takes 6 semesters, the last semester includes a small thesis that has to be defended in a colloquium at the end. Students can continue for the Master in Business Informatics, it is an M. Sc., which takes another 4 semesters including one semester for a master thesis.

Programming skills are seen as basic skill for graduates in Business Informatics although it is well known that some graduates wont program at all in their future career.

Introduction into Programming is a one semester mandatory course. According to the European Credit Transfer Systems a student gets 5 credits on successful completion. A full time student has a load of modules worth 30 ECTS per semester; see (European Commission, 2015) The module offers 4 contact hours per week: a lecture of 2 academic hours (90 minutes) and lab session of 2 academic hours. According to the work load calculations in (European Commission, 2015) we expect that students spend another 4 hours per week on the topic.

For decades an introduction into programming has started with the basic lexical elements like identifiers, key words as well as variables and constants of various types. Based on that a very small programme has been developed that shows the general structure of a program written in that chosen programming language. Step by step expressions, statements, sub-routine concepts and parameters will be introduced until students are able to write more complex programs. Object orientation has been handled as an add-on at the end. The author has followed such a common approach for many years, teaching Pascal or Java.

Especially in an applied science like Business Informatics the main question is: How can we map the real world into the world of a computer? We have to map real-world objects or subjects and their actions into data and algorithms. Using an object-oriented programming language makes it easier to develop such a mapping than by using other programming languages. Therefore, the author follows the objects-first approach as outlined in (Barnes & Kölling, 2003) or later editions.

Topics addressed in the module are:

- Objects and classes defining objects,
- Collections of objects,
- Abstract Data Types as stack, queue, tree,
- Searching and sorting,
- Inheritance.

After a general introduction into the module, its organisation and structure, the first topic focusses on objects. Pre-defined objects are used to introduce the concept of an object as a container of attributes (fields or instance variables) and actions (methods). Students learn that an object can have

\(^1\) Those who are not familiar with the notion Business Informatics will find acceptable information in Wikipedia: https://en.wikipedia.org/wiki/Business_informatics
different states and that many objects can be created based on one definition: the class. The class concept typical for object-orientation, the so called blueprint class, is addressed using real-world examples like person, book, account etc. Obviously these classes follow a common and simple structure: instance variables, constructor methods, access methods, and possibly methods for processing or output. Such classes do not cover real programming, but this object-oriented approach makes it easier for the students to make first steps in a programming language.

Step by step interactions between objects and between classes are introduced leading to a divide-and-conquer approach to programming. For managing many objects collections are necessary and as in (Barnes & Kölling, 2003) two techniques are used: First, collections based on the pre-defined Java class ArrayList and second, the traditional array technique. Working with collection needs loop-statements which are introduced in this part of the module. Thus structured programming is addressed step-by-step whenever it is needed.

The section on abstract datatypes like stack, queue, list or tree is used to train various implementations of collections including structured statements like if-then-else or various loop statements. Besides this training, students become familiar with these concepts well known and applied in computer science.

Searching and Sorting are typical tasks in business applications. Students learn about searching and sorting algorithms in general and the topic is used again for training of handling arrays or ArrayLists. This part is real programming in the traditional programming sense. Here static methods as elements of so called library classes are used.

Finally, the concept of inheritance is discussed as a speciality of object orientation.

Figure 1: BlueJ: Class diagram and state of an object

All students usually work on their own notebook. Most of them use Windows operating systems but some students have Apple computers or some prefer Linux as an operating system. The free software BlueJ² is used as the Java development Environment and works on all systems mentioned above. From the teacher’s perspective BlueJ offers a few important advantages:

- Objects can be created and its behaviour can be watched interactively. This avoids the introduction of main methods and executable classes as well as writing lots of output statements.

² BlueJ, see http://bluej.org
• BlueJ does not offer dozens of icons that create hundreds of code lines automatically: users can concentrate on pure Java code while writing classes.

• Classes and their dependencies are shown in a UML class-diagram style.

TEACHING APPROACHES

Although the object-first approach was quite new in 2005 the author followed a rather traditional style of teaching. In a lecture (90 minutes) a set of slides were used as a backbone to discuss the topics. To illustrate the theory example programs were developed by the lecturer using BlueJ right in the class. Students should listen to the talk, think about the content and ideally should come up with questions. For the lab class a set of exercises had been provided. The programming exercises should be accomplished in the lab class with the help of the lecturer or tutor.

Not satisfied with previous exams results the author introduced weekly assignments. Students could collect points that later were added to the exams results. Although this is not completely in line with the examination regulation it encourages students to work continuously from the beginning until the end of the semester. Moreover, the collected points could help the students to pass the exam. In the weekly lab class students shortly presented their results. In such a way it could be roughly checked that an assignment was a student’s own work. At least he had to show that he could present the programme that is he understood the programming.

After a few years the author dropped the extra weekly assignments for reason outlined in the following Result chapter. Students just attended lectures and had the chance to solve exercises in the lab classes. The latter was supervised by the lecturer or a tutor. The written exam at the end of the semester was then the one and only serious feedback for students.

The notion Just-in-Time teaching was already coined in the nineties; see (Simkins & Maier, 2010). Nevertheless the author got to know the concept in 2014 only and in 2015 he included elements of the Just-in-Time teaching into the programming module. Weekly lectures are not used any more for presenting content and developing programs. Instead at least half of the time, sometimes even more than 80% of the time has been used for the discussion of students’ questions and problems that have occurred while students were working on a programming exercise on their own outside the lecture.

/**
 * Write a description of class Student here.
 * @author Author
 * @version DD.MM.YY
 * @Difficulty of the task:
 * (0-simple, 1-solveable, 2-difficult (lot of time), 3-insolvable)
 * @Especially difficult for me ...
 * @I did not understand ...
 * @I have the following questions: ...
 */
public class Student {
    // instance variables
    private int x;
    ...

Figure 2: BlueJ: Class header template for Java exercises
After the discussion of all remarks and questions a short introduction into the next topic is given and the next task is explained in the lecture. Students try to solve the programming assignment on their own. Students have to read about some new elements or techniques in order to solve the task. The students are forced to express their problems: A solution is only accepted if the provided class header is filled in; see figure 2. It is more important that students honestly come up with their rating of the exercise and their question than a completely correct Java class.

Beside this different structure of a lecture the lab class are run in the same way as before. There is one exception: neither the lecturer nor the tutor will help to solve the assignment exercise.

RESULTS

The following analysis is based on the results of the written examinations at the end of a semester. Exams results from 2006 till 2016 are available. A maximum of 100 points is possible. Thus, points can be seen as percentages as well. German marks are outlined in the tables as well; see table 1.

Table 1: German marks

| mark | meaning | mark | meaning | mark | meaning | mark | meaning |
|------|---------|------|---------|------|---------|------|---------|
| 1.0  | excellent | 1.7  | good +  | 2.7  | satisfactory + | 4.0  | pass |
| 1.3  | very good | 2.0  | good | 3.0  | satisfactory | 5.0  | fail |
| 2.3  | good – | 3.3  | satisfactory – |

Table 2 gives a general overview on the results putting together the number of students in that year, the average number of points and the resulting average German mark.

Table 2: Exams results: average points and marks

| year | students | points | mark |
|------|----------|--------|------|
| 2006 | 52       | 55.0   | 3.6  |
| 2007 | 32       | 50.3   | 3.6  |
| 2008 | 51       | 51.2   | 3.4  |
| 2009 | 33       | 56.9   | 3.3  |
| 2010 | 39       | 60.5   | 3.3  |
| 2011 | 31       | 52.0   | 3.6  |
| 2012 | 28       | 42.8   | 4.2  |
| 2013 | 15       | 54.1   | 3.6  |
| 2014 | 39       | 58.5   | 3.6  |
| 2015 | 31       | 75.1   | 2.5  |
| 2016 | 24       | 69.8   | 2.9  |

The average mark is only one perspective. The original idea behind weekly assignments was to reduce the number of students who fail the exam. In the years 2006 till 2008 only up to three students profit from the assignments. Profit means, they would have failed without the bonus points collected during the semester. An average effect was different: Most of the students improved their marks using the bonus points. We could also argue: if someone regularly works on assignments during the semester he is well prepared for the final exam and in average wont fail.
Table 3: Exams results: fail rate

| year | students | assignm. | fail* | rate   | fail** | rate   | minus |
|------|----------|----------|-------|--------|--------|--------|-------|
| 2006 | 52       | yes      | 24    | 46.2%  | 21     | 40.4%  | 5.8%  |
| 2007 | 32       | yes      | 17    | 53.1%  | 14     | 43.8%  | 9.4%  |
| 2008 | 51       | yes      | 19    | 37.3%  | 18     | 35.3%  | 2.0%  |
| 2009 | 33       | no       | 13    | 39.4%  | 13     | 39.4%  | 0.0%  |
| 2010 | 39       | no       | 12    | 30.8%  | 12     | 30.8%  | 0.0%  |
| 2011 | 31       | no       | 13    | 41.9%  | 13     | 41.9%  | 0.0%  |
| 2012 | 28       | no       | 19    | 67.9%  | 19     | 67.9%  | 0.0%  |
| 2013 | 15       | no       | 6     | 40.0%  | 6      | 40.0%  | 0.0%  |
| 2014 | 39       | no       | 15    | 38.5%  | 15     | 38.5%  | 0.0%  |
| 2015 | 31       | yes      | 3     | 9.7%   | 2      | 6.5%   | 3.2%  |
| 2016 | 24       | yes      | 8     | 33.3%  | 8      | 33.3%  | 0.0%  |

fail*: number of students who got less than 50% in the written exam
fail**: number of students who finally failed, i.e. exams result plus bonus from assignment is less than 50%.

The fact that the assignments helped only a very few number of students to pass the exam who would fail otherwise supported the decision to cancel weekly assignments. After six years without any assignments and any possibilities to gain bonus points for the exam the author implemented assignments again: Since 2015 assignments have been organised differently including elements of Just-in-Time teaching as well as elements of the flipping the classroom (Bennett, et al., The Daily Riff, 2013). The intensive weekly discussion of students’ feedback ended up in the best exams result ever: A fail rate under 10%.

CONCLUSION

The article reflects the experience of the author in teaching programming in a first year undergraduate course. The course is a mandatory module in the degree programme Bachelor in Business Informatics. The author has changed the way of teaching three times using almost the same content. Unfortunately, there is no relation between the way of teaching and the exams results. The use of elements of Just-in-Time teaching in 2015 yielded very promising results including the lowest fail rate. Unfortunately, the next year shows a completely different picture: Formally the fail rate (33%) is still under the average (38%) but it is not a clear argument for Just-in-Time teaching.

The average fail rate of the years without any assignments 2009 – 2014 and bonus points we get 43.1% while for the assignment years 2006 – 2007, 2015 – 2016 we get 31.8%. The conclusion that weekly assignments reduce the fail rate is not valid: If we ignore the spikes in both lists the results are almost equal 38.2% and 38.1%. Thus, the figures do not argue for a specific way of teaching.

The author argues that students or the group of students in a year are more important than the way of teaching. It is important that students communicate. If students communicate with the lecturer and among themselves than fruitful discussions are more likely and better exams results can be expected. Thus, the challenge is not teaching a programming language but open students’ minds for a productive learning environment.
REFERENCES

Barnes, D. J., & Kölling, M. (2003). Objects First with Java - A Practical Introduction using BlueJ. Edinburgh Gate: Pearson Education.

Bennett, B. E., Spencer, D., Bergmann, J., Cockrum, T., Musallam, R., Sams, A., . . . Overmyer, J. (09. July 2013). The Flipped Class Manifest. retrieved on 21. 04 2017 from The Daily Riff: http://www.thedailyriff.com/articles/the-flipped-class-manifest-823.php

European Commission. (2015). ECTS key features. retrieved on 10. 04 2017 from http://ec.europa.eu/education/ects/users-guide/key-features_en.htm#ectsTop

Simkins, S., & Maier, M. H. (2010). Just-in-Time Teaching. Sterling: Stylus Publishing.