Learning System FRIMAN

Jozef Kostolny and Monika Vaclavkova

Department of Informatics,
University of Zilina
Zilina, Slovak Republic

(Jozef.Kostolny, Monika.Vaclavkova)@fri.uniza.sk

Abstract. Education of professional managers who are able to lead teams or companies developing information systems is a specific process. On our Faculty of Management Science and Informatics, we want to modify the teaching of relevant courses in such a way that more challenging themes and techniques will be distributed to smooth the learning process. For this purpose, we have prepared a proposal of a teaching system that allows focusing on the basis of algorithms in object-oriented programming environment in the first part of the considered courses. More advanced concepts of the programming that are considered difficult or problematic by beginners will be shifted to the later parts of the courses.

Keywords: Learning System, Object-Oriented Programming, Language Syntax, Unified Modelling Language, Information Management

1 Introduction

Informatics for managers can be defined as the application of informatics and information technology in management. Checkland [1] describes this as follows: technology works with data, people interpret them as information bearing significance, which becomes a stimulus for further negotiations. The process of interpretation of information is a cognitive process whose results can be interpreted as knowledge. It is emphasized frequently that good managerial work in various fields requires using information and communication technologies. Information and communication systems in information management represent useful support tools for work acceleration. Therefore, it is important for managers to understand the basics of information and communications technologies. Today, this is taken as the necessary computer literacy. In addition to basic knowledge of operating systems and office applications, workers and potential workers in the management of information technology should also have knowledge of the design, creation and maintenance of information systems. They should have also basic knowledge of software engineering. At the Faculty of Management Science and Informatics of University of Zilina, two courses dealing with principles of Computer Science are taught in bachelor study program “management”: Informatics for Managers 1 and Informatics for Managers 2. The content of these courses is introduction to creation of information systems.

The core of courses Informatics for Managers 1 and Informatics for Managers 2 is development of application using object-oriented paradigm. Teaching beginners' courses at our faculty is carried out in Java. Therefore, educational materials for demonstration of the principles of object-oriented programming are created in Java. The principles of object-oriented programming are taught using the Java programming language. This language was chosen due to its high popularity in creation of real applications.

Most of our students in management have no knowledge of programming and designing information systems before the courses and, therefore, many students have usually problems to understand them.

In the first part of the courses, we deal with the basics of algorithm design and language syntax. During the teaching these courses, we found that one of the biggest obstacles for passing the courses successfully by beginning students is to understand the principles of syntax and compliance with the Java language. For these reasons, we have tried to find a solution that would simplify initial lessons of the courses and, at the same time, allow grasp and understand the basic concepts of object-oriented programming and software development. For this purpose, we have designed a learning system for teaching managers in the bachelor study program. This system was named FRIMAN, and it has been developed in the frame of Project course, which is taught in the master study program "informatics" at our faculty.

2 Teaching Software

Currently, environment BlueJ [2] is used as a support tool for teaching software development in courses Informatics for Managers 1 and Informatics for Managers 2. The graphical interface of this tool is shown in Figure 1. The environment has been designed at the University of Kent. This tool is intended not for development of professional applications but for teaching the principles of object-oriented programming in the Java programming language.

BlueJ is an easy development environment that is suitable for beginners. It incorporates a lot of user friendly modules, such as:

- syntax validator of the code in created classes,
- debugging module for compiled classes,
- color distinction between individual sections of the code.

BlueJ environment allows users to create instances of implemented and compiled classes with a help of the graphical user interface. The created instances have an ability to receive messages in a graphical way. The environment also allows you to view the internal state of the instance by inspecting values of the attributes. These and other features provide full support for beginner-level students learning object-oriented programming. Therefore, it can be stated that the graphical environment of BlueJ is in full compliance with abilities of beginners and acts as a useful support tool for them.

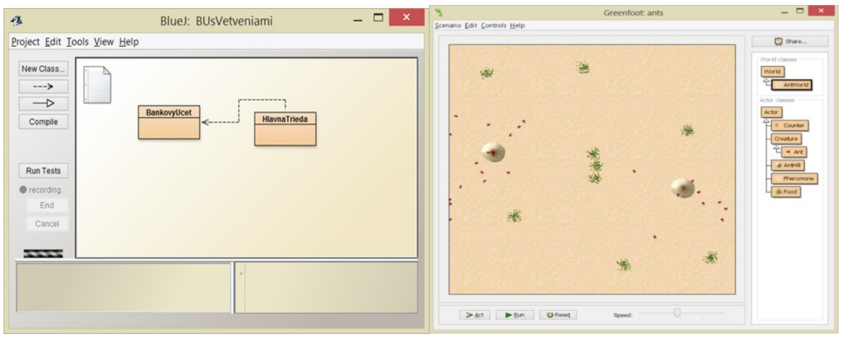


Fig. 1. Graphical user interface of BlueJ and Greenfoot

However, after several years of teaching the course, we concluded that the biggest problem for our beginning students is to understand the principles of Java syntax. One of the main problems is correct positioning brackets especially in blocks or nested blocks of the code.

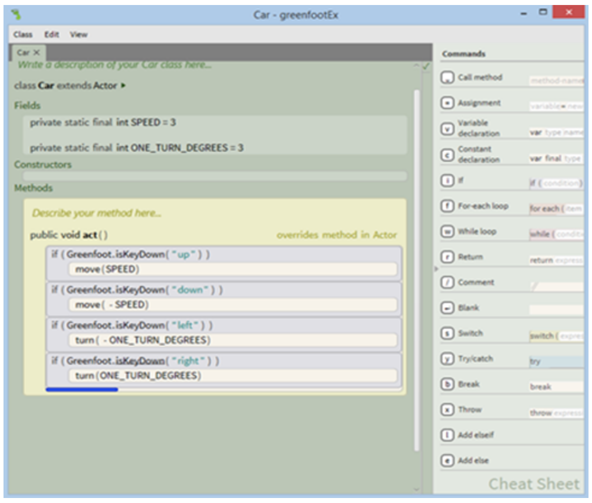
Another problem of our students is using the output terminal window. The current design of application graphical interface is good, but use of more complicated constructs, such as "`System.out.println(parameter)`" or formatted output, seems to be very confusing for beginners. Entering input data using class Scanner is also difficult to understand for beginners. The inputs and outputs handled by more advanced way, e.g. by class `JOptionPane` from package `javax.swing`, means even much more challenging task.

Due to the previously mentioned problems with syntax and others we do not have enough space for further dealing with important terms and concepts of object-oriented programming during the course. For these reasons, we have tried to find a solution that would eliminate the problems with the syntax and that would navigate students in introductory lessons to learn syntax and, at the same time, it allows grasping the essence of the explained concepts and knowledge. The required solution should have features equivalent to those in BlueJ, while it should minimize the need to use the syntax of the Java programming language. Our visions are as follows:

- In the first step, the aim is to familiarize students with the basic principles of programming and object-oriented paradigm without using the Java syntax. This part of teaching results in a common understanding of the basic principles necessary for the courses.
- In the second step, we plan to teach students considering constructions of the Java language. In this step, a tool suitable for development more complex Java applications will be used. Using this approach, we want to achieve a gradual consolidation of the most important knowledge.

3 Looking for Appropriate Solution

At the beginning of the process of finding solution to the aforementioned problems, we looked for a new environment that would satisfy our requirements. The most suitable one was Greenfoot [3], whose design, as we can see in Figure 1, is very similar to BlueJ. This tool has also been developed at the University of Kent. In Greenfoot, students acquire knowledge by creating simple computer games. This approach can be very interesting and attractive for students. The latest versions of Greenfoot contain a pseudo language called Stride, which allows creating simple game applications in more professional way. A big benefit of this tool is a possibility to select individual statements and language elements from the menu placed in the right column (Figure 3). After the selection, the user fills the fields of the statement, e.g. the condition and executed code in case of if-statement. The code created using Stride is then automatically converted into Java. Furthermore, this tool also allows developing applications by direct use of the Java programming language.



```
Write a description of your Car class here...
class Car extends Actor
Fields
    private static final int SPEED = 3

    private static final int ONE_TURN_DEGREES = 3
Constructors
Methods
    Describe your method here...
    public void act()                          overrides method in Actor
        if (Greenfoot.isKeyDown("up"))
            move(SPEED)
        if (Greenfoot.isKeyDown("down"))
            move( - SPEED)
        if (Greenfoot.isKeyDown("left"))
            turn( - ONE_TURN_DEGREES)
        if (Greenfoot.isKeyDown("right"))
            turn(ONE_TURN_DEGREES)
```

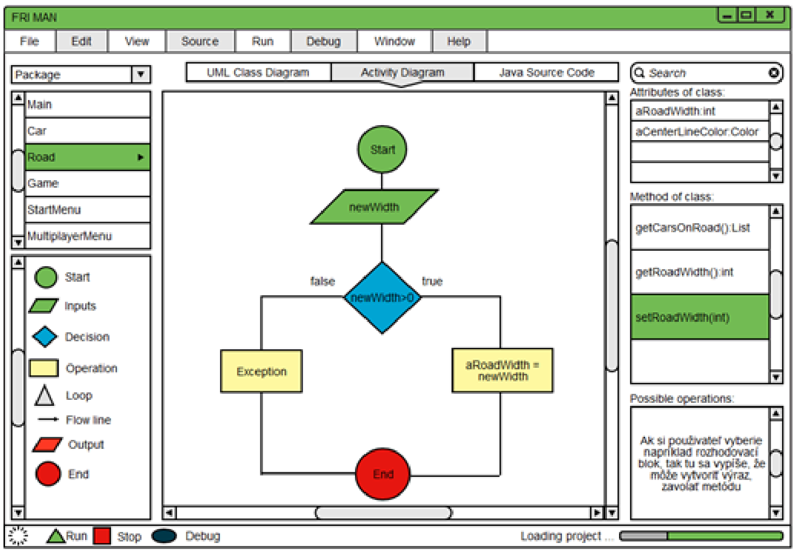
Fig. 2. Development environment of Greenfoot

However, the main drawback of the solution based on Greenfoot is a necessity to learn another programming language, i.e. Stride. Students would have to learn two languages and that could complicate the situation for beginners and bring confusion into acquired knowledge. Another ground for refusal of Greenfoot was the fact, that Greenfoot environment uses a construct of inheritance, which can be very difficult to understand by beginners. For these reasons, we had to look for other solutions.

4 Proposal of Learning System FRIMAN

In the courses for managers, every student has to create a semester project. The project is a simple application in Java, and it should present the knowledge gained during the semester. The semester project includes a documentation that contains class diagrams of the created application. Our experiences show that work with UML diagrams and tools for its creation is an important for students to gain knowledge. Usually, students do not have a problem with drawing the diagrams. Generally, understanding and creation of activity diagrams is much simpler for them than the programming itself. Because of that, the FRIMAN will address translation to Java automatically on request. Later lessons in other courses will be able to familiarize students with Java and will be focused on re-development of algorithms proposed using diagrams in Java. Our reflections include expectation that many of our students will be work as managers with different models and diagrams [6] but not as programmers.

Since only a minimum of these students will deal with implementation of software in the future, we do not teach a specific programming language in the majority of courses. For these reasons, the most important thing for students is to understand the importance of management direction, especially, design and development of applications using class and activity diagrams. The issue of implementation, which depends on a specific programming language, will be considered in other courses created for students that are interested.



Ak si používateľ vyberie napríklad rozhodovací blok, tak tu sa vypíše, že môže vytvoriť výraz, zavolať metódu

Fig. 3. Graphical user interface of FRIMAN

The system designed with respect to the previously mentioned requirements could be extended by other functionality that allows transformed the created project into other object-oriented programming language. This way, students can obtain valuable awareness on significance of the analytical phase of software development [7]. At the same time, they will be able to realize the fact that the model of the system is independent of the implementation language, which is important from a pedagogical point of view. By applying this extension, the system could also be simply adopted in future change of the programming language used in both courses. This language is currently Java, but it can be exchanged for any other programming language without changing the basic curriculum of the courses. The development of programming languages is a permanent process and more appropriate programming language may appear in the future [8].

The FRIMAN is connected with the UML FRI. The diagrams used in the FRIMAN are in fact XML documents generated by UML FRI. Graphical interface of the FRIMAN is user friendly to be easy to use and attractive for students. A view on the graphical interface of the FRIMAN is in Figure 4.

In the middle of the window, you can switch between the different modes of the system, namely:

1. complete class diagram of a selected class;
2. activity diagram of a selected method;
3. source code of a chosen method of a selected class in Java language.

In the left part of the window, basic graphic components for creating activity diagrams are located. In the right part, elements of a selected class, such as attributes and methods, are displayed. Further, in the bottom right corner, information relevant to a selected program component is present for additional help.

Using the FRIMAN, students can develop applications based on the following steps:

(a) designing classes including relationships between them using class diagrams provided by UML FRI tool;
(b) the FRIMAN creates basic representation of the classes, their attributes and headers of the methods and constructors by parsing the appropriate XML files obtained from the UML FRI;
(c) students will propose algorithms as the bodies of the generated constructors and methods with the help of graphical components provided by the FRIMAN;
(d) the FRIMAN transforms diagrams into Java language;
(e) debugging the final application using graphical representation of instances created from implemented classes.

Furthermore, the FRIMAN will be capable to debug the generated code, and the debugger will be connected with the graphical components. This allows students to see their logical errors directly in the diagrams.

The steps described above cover a lot of functionality. Because of that, the FRIMAN is divided into several modules with specific functionalities (Figure 5). The first module is XML parser that is designed to parse an XML document containing a UML class diagram. As mentioned above, the XML document is an output of the UML FRI. XML parser module is connected to the module responsible for inserting and processing class diagrams in a project controlled by the FRIMAN. The prepared files are then processed

and modified by the core modules of the application, which are connected to the main window of the application. The main window (Figure 4) permits performing several modes of processing, and therefore, it closely cooperates with the module responsible for editing programming constructs.
Editing individual programming constructs will be performed through graphical components. Examples of graphical components can be viewed in the left bottom part of the window shown in Figure 4. Individual graphical components must provide help for their operation. Every component that can work with variables (attributes, parameters and local variables) has to offer access to them. In the case of reference variables, the component has to be able to show interface of the referenced object.

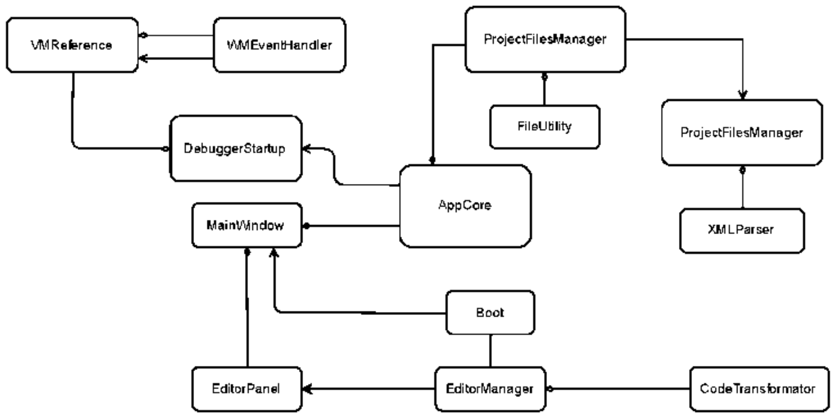


Fig. 4. Proposal of FRIMAN modules

The editing module is connected to the module whose function is transformation of graphical components into Java and vice versa. This module is `called Code-Transformator`. Another module called Boot runs the entire application and is connected to the main module called Main. After starting this module, the main window is shown. These core modules are connected with others that allow compiling the source code into Java bytecode. The compilation is done using the classic Java compiler that is part of Java virtual machine. Another module that has access to the compiler module is debugger. Java code can be debugged after each command, and every Java command will be represented in the FRIMAN editor as a graphical component. Graphical components can correspond not only to one command in Java but also to several. In this case, it is necessary to ensure that the debugging module will correctly interpret line numbers of the various graphical components. Further, if it is necessary, the module has to allow identifying the fault line. This problem is not trivial, and its solution requires finding correct transformation.

Learning system FRIMAN and its architecture brings a number of issues that needs to be resolved. Therefore, we plan to implement it utilizing the latest technology of object-oriented programming and using the JAVA language. One of the technologies that will be used is reflection.

5 Conclusions and Future Work

Ensuring the acquisition of comprehensive knowledge in the field of applied informatics for specialists in management who are educated on bachelor studies is very necessary nowadays. This requires from different educational institutions to prepare relevant courses that contribute most to that. In our courses we prepare professionals who will be managers and will be able to control teams or companies engaged in the creation of information systems. For that reason, acquiring knowledge in the field of design and creation of information systems plays a key role in our bachelor study.

Based on years of experience, we have identified which parts of teaching makes the biggest problem for our students. Based on them, we prepared a proposal of a learning system that will benefit from the positive aspects of the existing and previously used development environments. The proposed system will allow giving the emphasis to understanding the basic principles of object-oriented programming and allow students to avoid problematic parts of programming. The parts that beginning students consider problematic will be discussed in the later parts of the course. This way we try to ensure the equitable distribution of knowledge that should be acquired by the students during the course. We expect that the proposed learning system will increase percentage of students that successfully pass courses Informatics for Managers 1 and Informatics for Managers 2.

System FRIMAN, which was presented in this paper, allows creating projects and classes in the projects. In every class, it also allows declaring methods and creating their bodies through graphical components. This permits dealing with basis of algorithm design. Furthermore, the system allows generating code from the diagrams, creating instances of the compiled class, and debugging code.

Currently, we work on modification of the initial proposal in such a way that the drawing relevant diagrams will also be performed using the FRIMAN. After solving this issue, there will be no necessity of connection to the UML FRI, and the FRIMAN will be more consistent and less dependent on other projects.

References

1. Checkland, P., Scholes, J.: Soft Systems Methodology in Action. Wiley (1999).
2. University of Kent: BlueJ hompage, http://bluej.org.
3. University of Kent: Greenfoot hompage, http://greenfoot.org/.
4. Parso, E., Suroviak, D., Ondrejak, J., Dragula, M., Jesensky, M., Sensel, P., Sedlacek, P., Vaclavkova, M.: FRIMAN. Cent. Eur. Res. J. 2, 70–75 (2016).
5. Sadlon, L., Janech, J.: UML FRI hompage.
6. ŘEPA, V.: Analýza a návrh informačních systémů. Ekopress, Praha (1999).
7. Horstmann, C.S.: Big Java 3rd Edition. John Wiley & Sons, Inc. (2008).
8. Václavková, M., Nedeljaková, I., Kovalík, Š., Boháčik, J., Kopecký, J.: Informatika pre manažérov – základy programovania v jazyku Java. Žilinská univerzita v Žiline EDIS (2016).