BlueJ as the *NetBeans* Plugin

Rudolf Pecinovský
Department of Information Technologies
University of Economics, Prague
Prague, Czech Republic
rudolf@pecinovsky.cz

***Abstract* — One of the best IDE for introduction courses of object oriented programming is the *BlueJ* IDE. This IDE was developed with respect to needs of the absolute beginners. It offers almost everything what we need for teaching according the *Architecture First* methodology. However later, when the students go over to some professional IDE, they lose many of the *BlueJ* excellent features, especially the ability to design the program and its architecture in the interactive mode. The paper shows, how the *BlueJ* IDE was modified to work as fully functional plugin for *NetBeans* IDE and which new possibilities we obtain with it.**

Keywords—OOP; Architecture First; BlueJ; NetBeans; Plugin; Programming education;

I. Introduction

A. BlueJ Integrated Development Environment

The *BlueJ* IDE ([1]) is at present probably the best development environment for using in introductory courses of programming. It has the following strengths:

- It uses a class diagram for primary depiction of the developed program and thus it pushes the students to think about the developed program at the architecture level, not at the code level.
- The above mentioned class diagram is interactive and enables to involve the required changes of architecture very simply without being distracted by certain graphic details that take the attention off when using other UML diagram editors.
- Editing the class diagram is closely connected with editing the source codes. All important changes in class diagram are immediately included into source codes of influenced data types (interfaces, classes, enums …) and, on the contrary, all changes in the source code are almost immediately shown in the class diagram.
- The *BlueJ* environment is equipped with a simple code generator so that within first few lessons it is possible to design functioning programs without presenting the created source code to students.
- The environment includes a simple and natural support for test class creating that makes teaching the students more effective, and enables that the unit test creating becomes a natural part of the program development for them.

B. Architecture First methodology

The abilities of development tools as well as used frameworks are further improved. The programmer's focus goes currently from creating the code to designing the architecture. Contemporary trends even show that the time when various code generators will take over coding of the architecture design is a near future.

Unfortunately the methodologies currently in use do not react to these trends and continue to prefer teaching of programming that, to certain extent, concentrates only how to code the explained programming constructions and that postpones the explanation of architectural design to follow-up courses.

This approach is, however, contrary to an important *Early Bird* pedagogical pattern, which says ([2], [3]): *"Organize the course so that the most important topics are taught first. Teach the most important material, the "big ideas", first (and often). When this seems impossible, teach the most important material as early as possible."*

The *Architecture First* methodology ([6], [7] and [8]) respects the crucial importance of knowing the architecture design for future programmers and, according to the above stated pedagogical pattern, it lines up the teaching of the architecture design at the beginning of teaching process.

So that we could test the designed architecture, it is necessary to code the designed program. In case we would not like to distract the students with the explanation of syntactic rules and we would like to focus on the architecture design, we need certain code generator at our disposal.

At this time *BlueJ* with its code generator comes to help us. Using it, we can focus to the architecture design with students and coding of the relevant program will be provided by this generator.

Thus *BlueJ* with its code generator enables to divide the *Architecture First* methodology into required phases as follows:

1. Basics of object oriented programming are explained together with basics of object architecture design, whilst the

The paper was processed with contribution of long term institutional support of research activities by the Faculty of Informatics and Statistics, the University of Economics, Prague.

coding of created programs is in charge of the program generator.

2. After exhausting all possibilities of the used code generator the programs designed in the first phase are assessed once again. But nowadays the source code created by the generator and the necessary syntactic rules are explained to students at these examples.

3. Next item involves an explanation of constructions and architectural principles, which are already behind the possibilities of the used code generator.

II. Where are the problems

BlueJ offers a number of properties which we can use in applying the *Architecture First* methodology (and we also use them), however, some of its properties need further improvement as follows:

A. Small capability of the code generator

The current code generator has really only limited abilities. One of the activities, we are dealing with, is the significant extension of these abilities. Detailed information concerning this problem is described in e.g. [4].

B. No graphical view of the code

The *BlueJ* environment depicts graphically only the program architecture in the form of a class diagram. When teaching we would consider as very useful when it would be able to show also the used algorithms – for example through kopenograms. This topic is discussed in details in [5].

C. Bad cooperation with professional IDEs

BlueJ IDE is optimized for the introductory courses with the very beginners. Therefore its simplicity of using is especially important. However, this simplicity is redeemed by limitation of its functionality in certain areas.

On the other hand this limitation of functionality can be a good pretext for coming to the work with professional IDE in the moment when students reach certain level of their knowledge. However, the problem is in this case, that the possibility to work in the interactive mode as well as the close connection between the editor and the code generator is lost.

There is a plugin for *NetBeans* environment, which enables to open the project originally developed in *BlueJ* environment, and to open it in such way so that they could be open also in *BlueJ* environment again, but that is all what this plugin offers. In other areas the students have to fully submit to practice of this new environment.

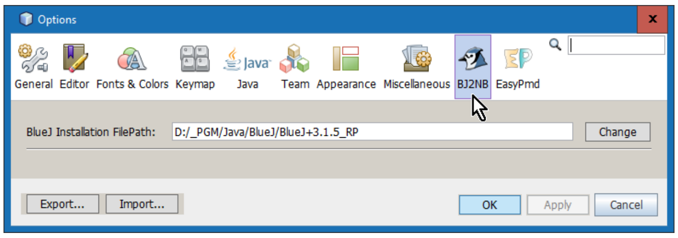
We used the fact that *BlueJ* environment is distributed as an open source and we decided to modify the *BlueJ* environment in such way so that it itself becomes the plugin of *NetBeans* environment and, if need be, also the plugin of other development environment. This paper deals with this extension.

III. Requirements and their solution

A. Easy installation

The first requirement for the created plugin was an easy installation available even for the beginners.

The plugin is installed in a standard way in which only the `nbm` file containing full necessary information for installing the relevant plugin is selected. During its installation the plugin defines its own card in the Options dialogue. This card allows setting the *BlueJ*'s location (see Fig. 1). If the user forgets it, plugin will points him.

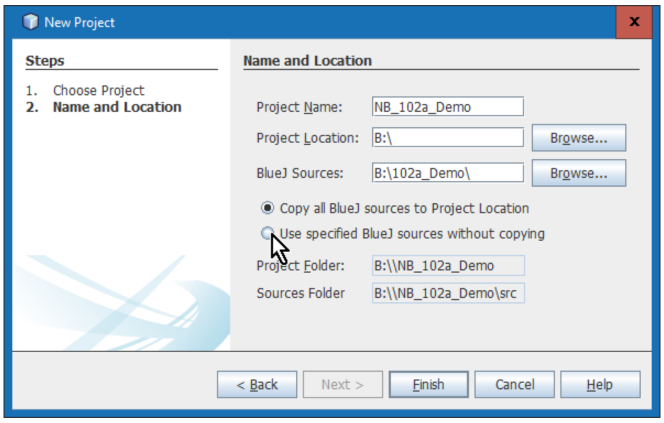


BlueJ Installation FilePath: D:/_PGM/Java/BlueJ/BlueJ+3.1.5_RP

Fig. 1. The Option dialog for setting the *BlueJ* location

B. Simple creation of new Project

Another requirement was the possibility to create easily the new project which would overtake the source codes of certain existing project created in *BlueJ*. Thus Import BlueJ IDE Project has been included into the menu of the project types. After its entering the dialogue from Fig. 2 opens. Besides the name and the location of the created project it requires also entering the location of source codes of the transferred original *BlueJ* project. The user can choose if these source codes will be copied into the created *NetBeans* project's folder or if he will work with original files.



Project Name: NB_102a_Demo
Project Location: B:\
BlueJ Sources: B:\102a_Demo\
[✓ Copy all BlueJ sources to Project Location]
Use specified BlueJ sources without copying
Project Folder: B:\\NB_102a_Demo
Sources Folder B:\\NB_102a_Demo\src

Fig. 2. Settings for the created project

This solution enables that the current *BlueJ* project could be easily converted into *NetBeans* project without losing the possibility to be used by independently operating *BlueJ* environment. With the other words, it enables that the project would be operating in both environments (and even simultaneously with certain limitations).

C. *Simple activation of BlueJ window*

BlueJ project is presented in the project window similarly as any other *NetBeans* project and we can work with it as with any current *NetBeans* project.

If we want to open some of the packages of the given project in *BlueJ*, we can only enter the command Show in BlueJ in the context menu. Then *NetBeans* will open the *BlueJ* window with the class diagram of the given package (see Fig. 3).

D. *Window Layout*

Another requirement was so that the appearance of the window in *NetBeans* tab would be closely similar to that one of the *BlueJ* application window, so that the students could be familiar with the new environment. As can be seen from Fig. 3, the only difference is that the buttons from the upper part of the button panel moved to their own panel.

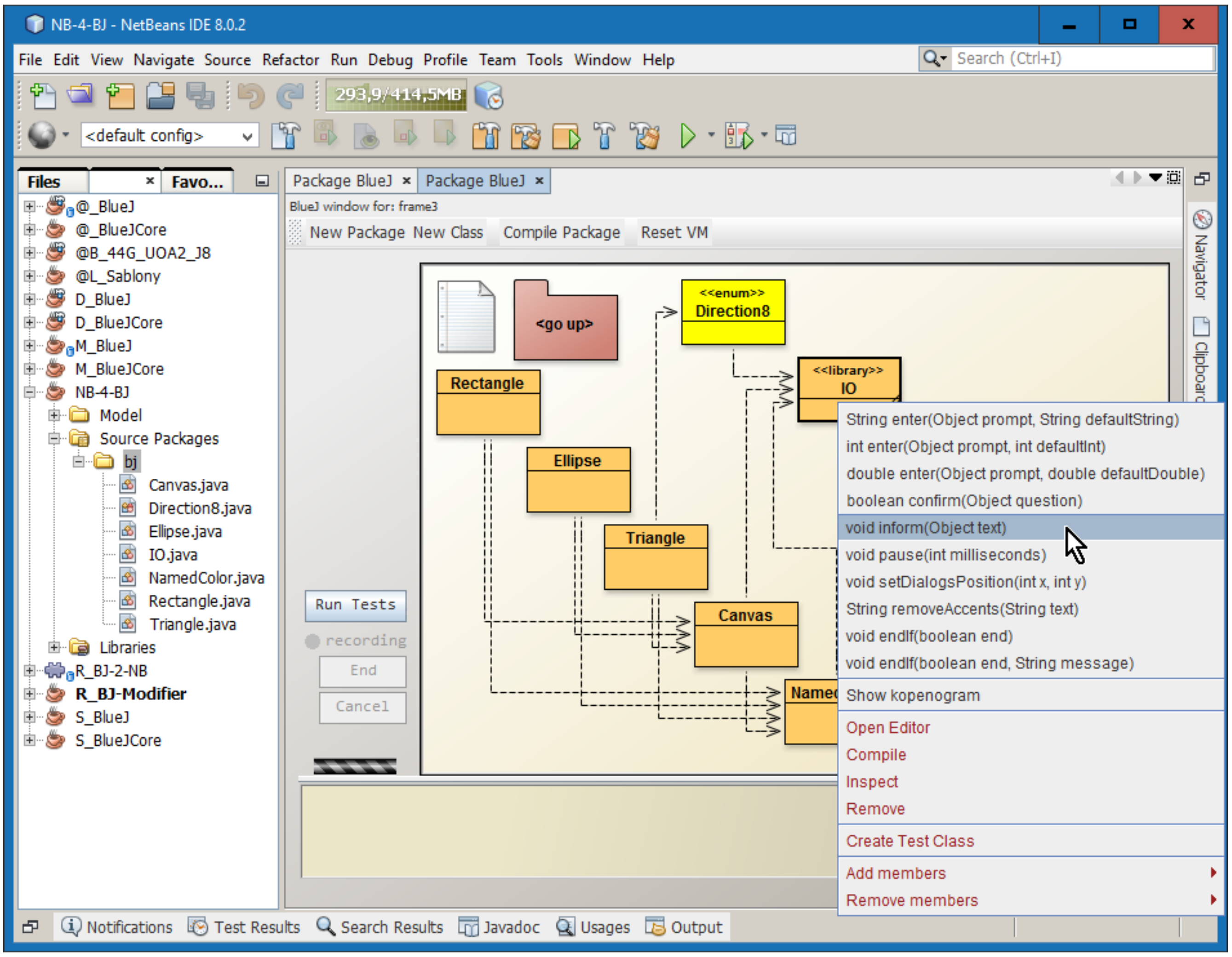


Fig. 3. The *NetBeans* application window with the Project window and *BlueJ* window with the opened package

E. *Functionality*

The key requirement for the final product was to mediate the functionality of *BlueJ* environment in *NetBeans* environment, above all the ability of working in the interactive mode and close connection of the source code and the class diagram. The user should have, in optimal case, an access to all functionality of *BlueJ* environment.

As is shown in Fig. 3, the *BlueJ* window provides all functionality of *BlueJ* environment including the possibility to work in the interactive mode.

F. *Editor*

Despite asking a maximum compliance of *BlueJ* functionality it was clear that it would be advantageous to prefer *NetBeans* functionality in certain areas. Such area was the source code editing. Therefore we asked *BlueJ* to use a comfortable editor inbuilt in *NetBeans* for editing the source code.

NetBeans editor will open independently to whether we ask for editing the source file from the *NetBeans* project window or from *BlueJ* window. Thus we can have an access not only to comfortable editor of the source code, but also to its other functionality, as is e.g. sophisticated refactoring, searching and replacing text using regular expressions, searching and replacing text in multiple files and folders etc.

IV. Summary

The first experience with the described plugin is very good. Students can significantly easier come from the environment in which they started their learning into a more comfortable, but at the same time more demanding environment in which they develop more complex programs used in further phases of studies.

Thus, using of this plugin is more advantageous also for teachers because they do not have to switch between the environments in which the programs for teaching are designed and the environment in which they are subsequently verified. They can get quickly the feedback and can far more operatively verify the applicability of proposed procedures.

References

[1] Page About *BlueJ* [online]. at http://bluej.org/about.html

[2] BERGIN, J.: Fourteen Pedagogical Patterns. *Proceedings of Fifth European Conference on Pattern Languages of Programs.* (EuroPLoP™ 2000) Irsee 2000.

[3] BERGIN, J.: *Pedagogical Patterns: Advice For Educators*. CreateSpace Independent Publishing Platform 2012. ISBN 1-4791-7182-4.

[4] BOBUSKY S.: *Enhancing BlueJ interactive mode*. Master thesis at University of Economics, Prague. 2015.

[5] CHADIM M., PECINOVSKÝ R.: Kopenograms and their implementation in BlueJ. SDOT 2015.

[6] PECINOVSKÝ Rudolf: Principles of the Methodology Architecture First. *Objekty 2012 – Proceedings of the 17th international conference on object-oriented technologies*, Praha 2012. ISBN 978-80-86847-63-4.

[7] PECINOVSKÝ, Rudolf, KOFRÁNEK, Jiří. The Experience with After-School Teaching of Programming for Parents and Their Children. Las Vegas 22.07.2013 – 25.07.2013. In *FECS'13 – The 2013 International Conference on Frontiers in Education: Computer Science and Computer Engineering.* http://worldcomp-proceedings.com/proc/p2013/FEC4207.pdf

[8] PECINOVSKÝ R.: *OOP – Learn Object Oriented Thinking and Programming*. Eva & Tomas Bruckner Publishing 2013. ISBN 80-904661-8-4.

.