УНИВЕРЗИТЕТ „ГОЦЕ ДЕЛЧЕВ” - ШТИП
ФАКУЛТЕТ ЗА ИНФОРМАТИКА

ISSN 1857- 8691

ГОДИШЕН ЗБОРНИК
2012
YEARBOOK
2012

ГОДИНА 1 **VOLUME I**

GOCE DELCEV UNIVERSITY - STIP
FACULTY OF COMPUTER SCIENCE

УНИВЕРЗИТЕТ „ГОЦЕ ДЕЛЧЕВ“ – ШТИП
ФАКУЛТЕТ ЗА ИНФОРМАТИКА

ГОДИШЕН ЗБОРНИК
2012
YEARBOOK
2012

ГОДИНА 1 МАРТ, 2013 VOLUME I

GOCE DELCEV UNIVERSITY – STIP
FACULTY OF COMPUTER SCIENCE

ГОДИШЕН ЗБОРНИК
ФАКУЛТЕТ ЗА ИНФОРМАТИКА
YEARBOOK
FACULTY OF COMPUTER SCIENCE

За издавачот:
Проф д-р Владо Гичев

Редакција и администрација
Универзитет „Гоце Делчев“-Штип
Факултет за информатика
ул. „Крсте Мисирков“ 10-А
п. фах 201, 2000 Штип
Р. Македонија

Address of the editorial office
Goce Delcev University – Stip
Faculty of Computer Science
Krste Misirkov 10-A
PO box 201, 2000 Štip,
R. of Macedonia

СОДРЖИНА
CONTENT

JAVA IDEs FOR EASILY LEARNING AND UNDERSTANDING OBJECT ORIENTED PROGRAMMING

Aleksandra Stojanova[1], Natasha Stojkovic[2] and Dusan Bikov[3]

[1]*aleksandra.stojanova@ugd.edu.mk*
[2]*natasa.maksimova@ugd.edu.mk*
[3]*dusan.bikov@ugd.edu.mk*

Abstract: Introduction to object-oriented programming (OOP) can be difficult for beginners in programming, especially if only pure code is used. To facilitate learning and understanding the concept of OOP many Java Integrated Development Environments (IDEs), that contains a lot of visual elements, are developed. Adding the visualization make programming easier, more interesting and interactive for users. These environments help to decrease the age of programming beginners. In this paper it will be given a brief overview of some of these environments. It will be done a comparison between them emphasizing differences among them, their advantages and disadvantages.

Keywords: object-oriented programming (OOP), Integrated Development Environment (IDE), visualization, Java.

1. Introduction

The learning and teaching of programming remains a challenging topic in the field of computer science education [2]. Working environments are very important in the study of programming, especially if it is an object-oriented programming. When the environments enriched with visualization and interaction, programming introduction becomes less abstract and less theoretical [5]. Java programming language is used worldwide in universities for learning basic concepts of OOP. Therefore many IDEs based on Java, are developed. The main purpose of Java IDEs is making process of programming easier, faster with more visual elements and less code writing [1]. In this paper we will start with introduction to Java IDEs, like BlueJ[12], Alica[11] and Greenfoot[14].We will shortly describe each of these environments and their constituent integrated elements. We also will describe and their way of working and their way of creating and presenting objects and classes. We will give the reason of using these IDEs, and target age group of users. Then, we will make comparison among these tree IDEs and we will emphasize their pros and cons.

2. Examples of IDEs for easily learning of object oriented concepts

In this paper three integrated development environments are reviewed. These environments are Alice, Greenfoot and BlueJ. They have the same goal and that is to help understanding object concepts, but they use different ways to accomplish the goal.

2.1. Alice

Alice is an IDE specifically designed as a learning tool to enable young programmers to create animations and games using 3D worlds. Alice use program visualization and enables users to see how their animation programs run. Programming visualization environment offered through Alice might be highly motivating to college students, especially for today's generation of video games and animated films [1].

It uses drag-and-drop interface, and no text entry, to make the learning fun and interesting. This Alice's feature prevents users from making syntax errors, which are very common for beginners. It is intended for beginners in object-oriented programming, and it is easy and fun way to begin learning the Java language. Working with Alice provides fast visual feedback of the program and easy understanding of its object oriented structure and relationship between each programming statement and the corresponding behavior of objects in their animation [3].

Its 3D modeled classes and instantiated objects provide concrete picture of the concept of an object [16].
Alice's library contains hundreds of 3D models which are provided in a gallery of Java classes. These models help virtual world to be enriched and populated. The user can create animations using these models that can move around a virtual world.
There are two types of animations in Alice: movie (passive animations) and interactive. The users immediately can see how their animation program runs [3].

Working in Alice consists of two phases: creating a scene and scripting (editing the code). The user can easily select new objects from a gallery that contains predefined objects, and can give the object functionality by using primitive methods from drop-down menus or by writing new functions. (Figure 1)

Figure 1. Screenshot from Alice interface.

The language presented to the user in Alice, could be with the all Java syntax details but still it is drug and drop and not standard Java text-editor environment. This environment can be used as plugin in NetBeans [10] and Alice users can easily transfer their Alice project directly into Java text editor environment [3].

2.2 Greenfoot

Greenfoot is another integrated environment for learning programming. The program language that is used in this environment is Java too. Unlike Eclipse [13] or NetBeans [10] that also are Java IDEs and are using visual elements, this environment is more interactive and needs less previous theoretical knowledge for programming. But Greenfoot is also different from Alice because it needs more writing java code for making programs, and the language exposed to the users is Java. Greenfoot is Java two dimensional environment and is specialized for development of animations, simulations and games. It is designed for using in height schools and first programming courses in universities. [7]

This environment is a full interactive world of objects. It is a self-contained system that provides a full IDE, including integrated editor, compiler and debugger on source level and also contains build in classes and objects and allows creation of new classes and object [5, 6].

Its runtime environment and compiler uses standard Java. Also classes that are used there are pure Java classes and the syntax is also the same of standard java syntax.

Some of the advantages of Greenfoot are: easy way of making programming and giving visual feedback to the user. Scenarios are flexible and offer different level of complexity that makes this system usable in different age group categories. Greenfoot give very clear way of presenting object oriented concepts like (class, objects, inheritance etc.) It allows easy development of different scenarios and games with a little previous theoretical knowledge background, and also allows interaction with objects and easy control of their behaviors. Another advantage of this environment is the existence of an easy way of migration to other Java environments. Greenfoot is easy way to start programming games and simulations complete with 2D graphics and sound. But the support for text is very poor and is not using 3D [8,6].

Greenfoot user interface consist of the world (the background of the scene) and actors (Greenfoot objects). The two superclasses in the Greenfoot are World and Actor and they are always present in the scenario. All other classes are sub classes and derived from these two superclasses. One scenario of Greenfoot is presented in Figure 2.

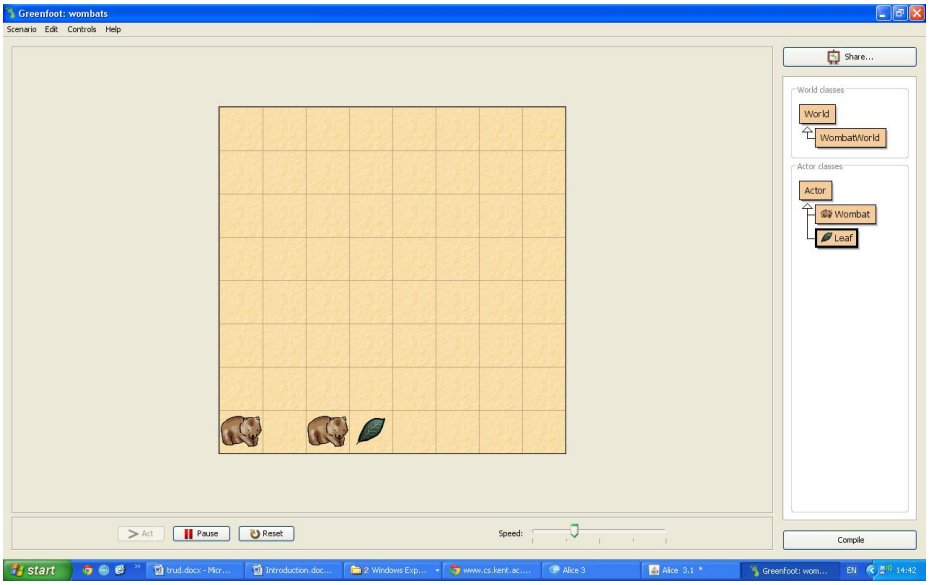

Figure 2. Screenshot from Greenfoot interface.

Greenfoot has build in predefined classes, objects and methods but also give a user a chance to create new classes objects and functions that give functionality of the instances (actors). The classes can be edited, compiled and instantiated. At any moment user can see the hierarchy structure of classes. The code of each class can be seen in code editor where the user can edit the existing code and add new code. The explanation of each class can be seen in the documentation [14].

2.3 BlueJ

BlueJ is specifically developed for the purpose of teaching object oriented programming with Java and it is free and open source software. This environment can run on all platforms supporting a recent Java virtual machine [11,13]. BlueJ is fully integrated environment. It supports graphical visualization of class structure and also a textual editing. It have built-in editor, compiler, virtual machine and debugger therefore it offers easy-to-use interface ideal for beginners.

BlueJ provides clear separation of the concepts of classes and objects. Classes and object in this environment are visually represented as UML (Unified Modeling Language) class diagrams (Figure 3) [6]. In this kind of visualization, hierarchy among classes can be clearly seen, but BlueJ does not provide direct visualization of any the object’s state or behavior.

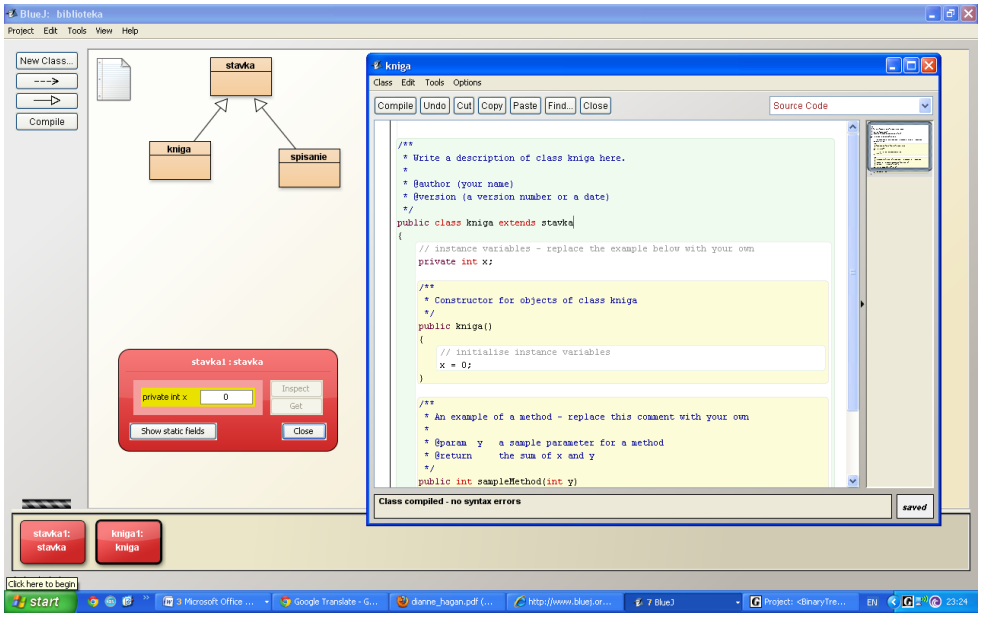

Figure 3. Screenshot from BlueJ interface

It allows easy making of classes and when the classes is added and compiled, the user can interactively instantiate objects and execute their methods through a sequence of pop-up menus. This allows users to immediately see the effect of a method invocation on that object and also simplifies the debugging process [9].

This environment uses the standard javac compiler, that gives users exposure to the exact same language, and the same compiler error messages that they might get when they leave BlueJ and move on to other tools. These kind error messages are not always helpful, especially to beginner programmers, so BlueJ provides extended help text for better understanding. This environment allows compilation and running the program without any main methods, but does not support as much features as professional IDEs, like NetBeans [10], Eclipce[13] or JBuilder[15] does.

BlueJ allows easy migration from educational tool to professional Java IDE, because in NetBeans IDE exists BlueJ plug-in that can help the migration [10].

3. Making comparison

All these three IDEs are used for easily learning of object oriented programming but still, they are different. They are different from commonly used IDEs for Java, like Eclipse and NetBeans. On one hand, these three

environments can be used for learning Java by users who have very little theoretical knowledge of programming, and on the other hand Eclipse and NetBeans are environments that are used for professional development of complex Java programs.

Target group of users of all three environments are novices programmers, but Alice is named for younger beginners. It can be used in first classes of introduction in programming. In Alice object are presented in 3D and are more realistic and more fun to animate the beginners. Greenfoot is 2D environment but still similar to Alice because it uses defined set of classes, object and methods, and all objects are presented with adequate pictures.

In BlueJ environment classes a visually presented as UML diagrams, and not as pictures. All three environments provide interactivity to user.

Alice does not use text editing at all, methods and objects are made using drag-and-drop interface and pop-up-menus. Greenfoot and BlueJ on the other side have text editors where the user can write his own code.
Language presented to the user in Alice is not standard Java allowing the user to avoid making syntax errors. In Greenfoot and BlueJ, language exposed to the user is standard Java, to make better approach to professional Java IDEs.

All three environments offer way for migration to professional IDE for programming. Alice and Greenfoot are more game oriented and might be used as a tools for easy making games and animation.

According the features they offer for learning programming Greenfoot is in the middle between Alice and BlueJ. So in the learning chain novices programmers might begin with Alice then migrate to Greenfoot and then to BlueJ. BlueJ is most closely resembled with professional IDEs and its migration to professional environment is the easiest one.

4. Conclusion

Because object-oriented programming is quite abstract and complex, beginner-students often have trouble with learning the basic concepts of object–oriented programming. Furthermore, students often do not understand the reason for learning and advantages of using the object-oriented approach to software development.

Professional IDEs are not always adequate for learning basics of object oriented programming because these environments usually are not object oriented also they can be very complex and focused on building graphical user interface[9].

Alice, Greenfoot and BlueJ, make programming easier than other commonly available tools. They easily can be used from young people, non-

engineering, undergraduates with little or no-programming experience. They use visual elements to represent the object concepts and allows interactivity to user. Using visualization and interaction provides a sense of reality for objects.

Alice, Green and BlueJ, are environments that allow users to focus on the concepts of objects like encapsulation or inheritance rather than dealing with syntax errors. So, these three environments can be used as impressive introduction to a professional IDEs.

Our purpose was to present these little-known environments for Java and to clarify their pros and cons. They might be used for adding efficiency in the process of learning.

Refernces

[1] B. Moskal, D. Lurie and S. Cooper (2004): *Evaluating the Effectiveness of a New Instructional Approac.* ACM SIGCSE '04, New York, NY, USA ©2004 pp. 75-89

[2] D. McCall, M. Kölling and P. Henriksen (2010): *Motivating Programmers via an Online Community.* Journal of Computing Sciences in Colleges, 25(3): pp. 82-93, January 2010

[3] D. Culbya, D. Cosgrove, Don Slater and Wanda Dann (2012): *Mediated Transfer: Alice 3 to Java*. In SIGCSE'12. 2012. Raleigh, North Carolina, USA: ACM. pp. 141-146

[4] D. Hagan, and S. Markham (2000) Teaching Java with the BlueJ Environment, Australian Society for Computers in Tertiary Education - ASCILITE ,

[5] M. Klling and P. Henriksen (2005): *Game Programming in Introductory Courses With Direct State Manipulation*. In ITiCSE 2005 Proceedings, pp 59-63

[6] M. Kölling and Poul Henriksen (2004): *greenfoot:Combining Object Visualisation with Interaction.* ACM 2004 Article, New York, NY, USA ©2004, pp. 73-82

[7] M. Kölling (2009): *Introduction to Programming with Greenfoot*. Pearson Education, Upper Saddle River, New Jersey, USA

[8] M. Kölling (2010). *The greenfoot programming environment*. ACM Transactions on Computing Education (TOCE) Vol 10, Issue 4,Article 14 (November 2010)

[9] M.Kölling (2008) *Using BlueJ to Introduce Programming* ,Reflections on the Teaching of Programming, Springer-Verlag Berlin, Heidelberg ©2008 pp. 98 – 115

[10] NetBeans IDE, official pages. Read on November, 2012. http://edu.netbeans.org/bluej/
[11] Official website of Alice. Read on November, 2012. http://www.alice.org/
[12] Official website of BlueJ. Read on November, 2012. www.bluej.org/
[13] Official website of Eclipse Read on November, 2012 http://eclipse.org/
[14] Official website on Greenfoot. Read on November, 2012. http://www.greenfoot.org/
[15] Official website of JBuilder, Read on November, 2012 http://www.embarcadero.com/products/jbuilder
[16] S. Cooper, W. Dann and R. Pausch (2003): *Teaching objects-first in introductory computer science*, n SIGCSE'03, New York, NY, USA: ACM.pp.191-195