Asian Journal of Information Technology 11 (6-12): 194-200, 2012
ISSN: 1682-3915
© Medwell Journals, 2012

An Integrated Development Environment for Blocks Creation

Marlinawati Djasmir, Sufian Idris, Marini Abu Bakar and Abdullah Mohd Zin
Center for Software Technology and Management,
Faculty of Information Science and Technology, Universiti Kebangsaan Malaysia, Bangi, Malaysia

Abstract: Block Based Software Development is a software development approach that supports end-user software development. In this approach, end-users can develop applications by selecting, customizing and combining software blocks. In the current implementation, a block is developed by using the Java programming language and is packaged as a JAR file which may consists of the following files: the class files, text and html files, images, audio and video clips, configuration data and any other files required for the block to operate. In order to help block developers to develop software blocks, a special purpose Integrated Development Environment (IDE) need to be provided. This special purpose IDE can help to improve the quality of blocks, reduce cost and time and hence increase the productivity of programmers. In this study, researchers describe the design and development of this special purpose IDE. The study is done in three phases. During the first phase, requirement analysis process was carried out by analysing existing IDEs, especially those related to components development. Based on this requirement, the IDE is designed and implement. The last phase includes an evaluation process to determine the effectiveness of the software tool.

Key words: Integrated development environment, block based software development, end user programming, software components, Malaysia

INTRODUCTION

Block Based Software Development (BBSD) is a software development environment that supports end-user software development. The concept of BBSD is a combination between end-user programming and component-based software engineering (Zin, 2011).

End-user programming is a term that refers to computer programming carried out by end users (for example teachers, accountants, scientists, engineers and parents) who are not trained as programmers (Goodell, 1998). Since, most of end-user programmers do not have programming skills, it is very difficult for them to write programs by using conventional programming languages. In order to support end-user programming, a number of software tools have been developed. These software tools enables end-users to create applications from scratch or to modify a certain part of existing applications. Software tools to support end-user development must fulfill a number of criteria. The most important one is that the tools must be flexible and adaptable to various user environments. These tools must also easy to understand, to learn, to use and to teach (Lieberman *et al*., 2006).

Component-Based Software Engineering (CBSE) has become an increasingly popular approach to facilitate the development of evolving systems since it enables software to be developed by using reusable components (Bennett, 1995). The objective of CBSE is to take elements from a collection of reusable software components and build applications by simply plugging them together. Hence, the main aim of this technology is to produce high-quality software systems with shorter and more cost-effective development cycles. By reconfiguring components, adapting existing components or introducing new components it is hoped that applications could be adapted to changing requirements of real-world software systems more easily and address the problems of object-oriented development approaches (Cheung, 2004).

In BBSD, end-users develop applications by selecting, customizing and combining software blocks. In BBSD, it is assumed that many blocks are available for a given problem domain. The task of creating these blocks is done by professional programmers (also known as block developers). The block developers need to have a very strong technical expertise in order to be able to produce a very high quality and reliable blocks.

In order to support Block-Based Software Development, a number of software tools need to be provided. One of the most important tools is the one for creating blocks. The availability of such a tool will provide

Corresponding Author: Marlinawati Djasmir, Center for Software Technology and Management, Faculty of Information Science and Technology, Universiti Kebangsaan Malaysia, Bangi, Malaysia

a convenient way for block developers to develop blocks. This study describes a work that has been carried out in order to develop a specialized integrated development environment for blocks creation.

Literature review

Software components: A software component is generally defined as a piece of executable software with a published interface which must be delivered in a ready to use form Hopkins (2000). In CBSD, these software components can be integrated to form a functioning system. It is always assumed that components must be correct and reliable and that they must be represented by accurate and complete specification.

A number of component models have been proposed, such as CORBA (OMG, 1996), Microsoft COM (Rogerson, 1997), JavaBeans (Sun Microsystems, 1997) and Enterprise JavaBeans (Monson-Haefel, 2000). CORBA is the acronym for Common Object Request Broker Architecture. It is an open, vendor-independent architecture and infrastructure that enables computer applications to work together over networks. Microsoft COM (Component Object Model) technology enables software components to communicate within the Microsoft OS environment. COM is used by developers to create re-usable software components, link components together to build applications and take advantage of Windows services. Object-oriented languages such as C++, provide programming mechanisms that simplify the implementation of COM objects. The family of COM technologies includes COM+, Distributed COM (DCOM) and ActiveX® Controls. JavaBeans are reusable software components in the form of classes written in the Java programming language that conform to a particular convention. They are used to encapsulate many objects into a single object so that they can be passed around as a single bean object instead of as multiple individual objects.

Blocks: Within the context of Block Based Software Development, a block is a Single-Layer Software component that can be used to perform a specific task. Single-layer implies that a block cannot have sub-blocks and it cannot be a sub-block of another block.

A block consists of four elements: attributes, behaviour, GUI elements and interfacing (pin in and pin out). It is possible for a block to have more than one GUI.

In order to support block-based development in a particular domain, sufficient number of blocks for the domain needs to be provided. This task is carried out in four stages as follows:

- Identify the programming blocks required for the domain
- Design the blocks
- Implement the blocks
- Test the blocks

The process of block identification and design has been described in Afiza and Rokiah.

A block can be implemented by using various programming languages. However, currently most of the blocks that have been developed are implemented by using the Java programming language. The block is packaged as a JAR file which may consists of the following files: the class files, text and html files, images, audio and video clips, configuration data and any other files required for the block to operate.

Figure 1 shows an example of a block for learning letters. This type of block is useful for developing courseware for early childhood education. The block's package consists of:

- A class file for displaying text
- A class file for playing audio
- An audio file
- An attribute file

This block consists of seven attributes:

- Name
- Text
- Text colour
- Text size
- Text font
- Background colour
- Audio file name

This block has three behaviours:

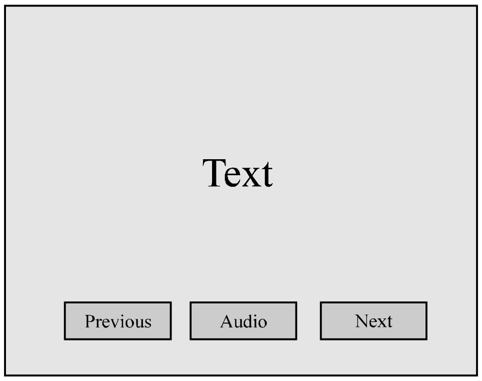


Fig. 1: A block for learning combination of capital and small letters

- Set audio on and off
- Move to the next block
- Move to the previous block

The attribute file is a simple text file in the following format:

- Name: LearnLetter
- Text: Aa
- TextColor: Black
- TextFont: Arial
- BackgroundColour: Brown
- Next: Nil
- Previous: Nil

A user can customized the block by changing values of the attributes to create an instance of a block. For example, Fig. 2 shows an instance of the block showed in Fig. 1. In this case, the text has been replaced by Aa. The user can give a name of this particular instance of the block.

Another example of a block is shown in Fig. 3. This block can be used for learning to link alphabets with words. The block's package consists of:

- A class file for displaying text and image
- A class file for playing audio
- An audio file
- An image file
- An attribute file

This block consists of fourteen attributes:

- Name
- Text1
- Text1 colour
- Text1 size
- Text1 font
- Text2
- Text2 colour
- Text2 size
- Text2 font
- Background colour
- Audio file name
- Image file name
- Image file width
- Image file height

Similar to the previous block, this block also has three behaviours. An instance of the block in Fig. 3 is shown in Fig. 4. In this case, the two text boxes are replaced with Chicken, C and the image is replaced with an image of a chicken.

Implementation of block: The pseudo-code for the class file for displaying text for the block for learning letters is shown below.

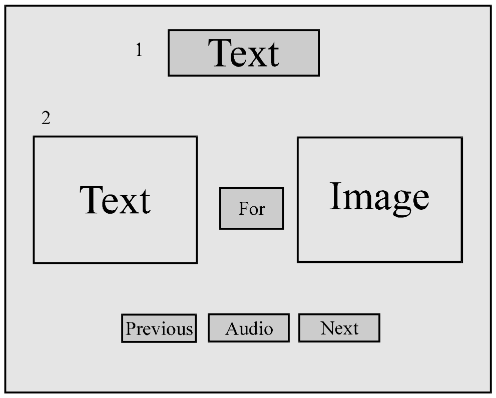


Fig. 3: A block for learning alphabets and words

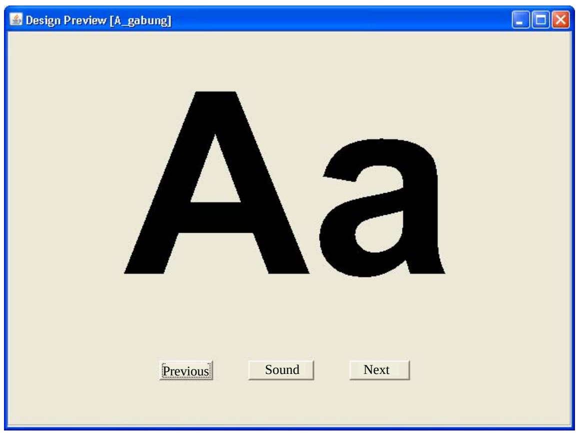


Fig. 2: An instance of a block for learning combination of capital and small letters

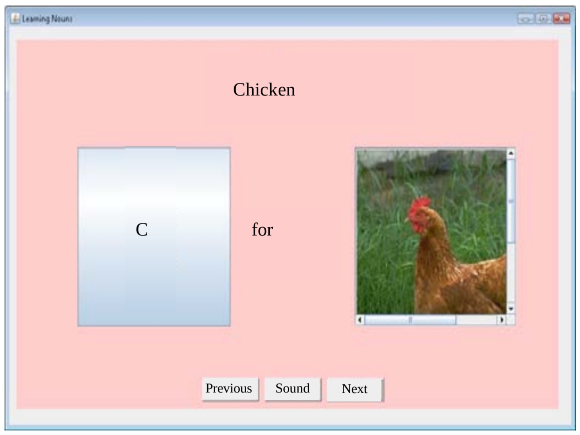


Fig. 4: An instance of a block for learning alphabets and words

```
package LearnLetters;

// import relevant Java libraries

 public class Letters extends
 javax.swing.JPanel{

 //variable declarations

   public Letters() {
     Initialize Components
     ReadAttribute File
      Set Attribute Described in the File
  }

  private void initComponents() {
   jLabelFont = new javax.swing.JLabel();
   prevButton = new java.awt.Button();
   nextButton = new java.awt.Button();

   Set Background Colour
   Set Border
   Set Text Colour
   Set Text Font
   Set Horizontal Allignment
   Set Text(A);
 }

  Private ReadAttributeFile() {
    // read file line by line
  }
}
```

MATERIALS AND METHODS

Requirement specification: An Integrated Development Environment (IDE) is a software tool that can help programmers to develop applications. The main objective of IDEs is to make programmers more productive. This is achieved by providing tools which are intuitive to use and cover more of the software engineering cycle.

The requirement of an IDE for block creation is shown as a use case diagram in Fig. 5. From the use case diagram, it is clear that the IDE should provide support for following functions:

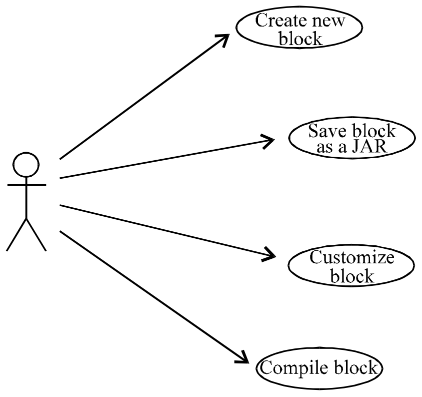


Fig. 5: Use case for using the integrated development environment

Create new block container: The first step in creating a new block is to create a new block container that enables a block developer to drag components into it.

Save block: A block can be saved temporarily as a java file. Once the block is completely developed, it must be saved as a JAR file for distribution purposes.

Customize the block: A block developers can customizes a block.

Compile the block: A block should be compiled to ensure that sure that it does not contain any errors.

Analysis of available tools: Since, a block is based on Java, block creation can basically be done by using any Java IDEs such as Netbean, JCreator and BlueJ IDE.

Eclipse (www.eclipse.org) platform is designed to serve as a common base for diverse IDE-based products, providing open APIs to facilitate this integration. Eclipse IDE is a professional IDE and contains advanced features and it is currently used widely. It is an open source software technology where users can download the software for free together with the source code. Eclipse is a platform-centric IDE with various development tools can be integrated within it (Chen and Marx, 2005). It is a universal IDE for anything and for nothing in particular (Chen and Marx, 2005). Eclipse provides code wizards to make it more productive. Eclipse does not only support extensions by allowing plug-in but user can modify the core code to solve problems encountered when extending the platform. Eclipse IDE is not Java specific where the menus and procedures are too generic and user need a documentation and tutorials about this IDE.

BlueJ (www.bluej.org) is an Integrated Java environment specifically designed for an introductory programming course in Java. This IDE is suitable for beginners and someone who have no experience in programming. It is better suited to introductory teaching than other environments for variety of reason: the user interface is simple and the environment provides important teaching tools that are not available in other environments. One of the most important strengths of the BlueJ environment is the user's ability to directly create objects of any class and then to interact with their methods.

NetBeans (www.netbeans.org) is a platform to create professional desktop, enterprise, web and mobile applications by using the Java programming language. This IDE is designed with reusability and extensibility in mind and thus it allows users to add new features easily or even to build their own applications by reusing NetBeans core classes as a framework. NetBeans IDE provides an amazing development and it a free and its

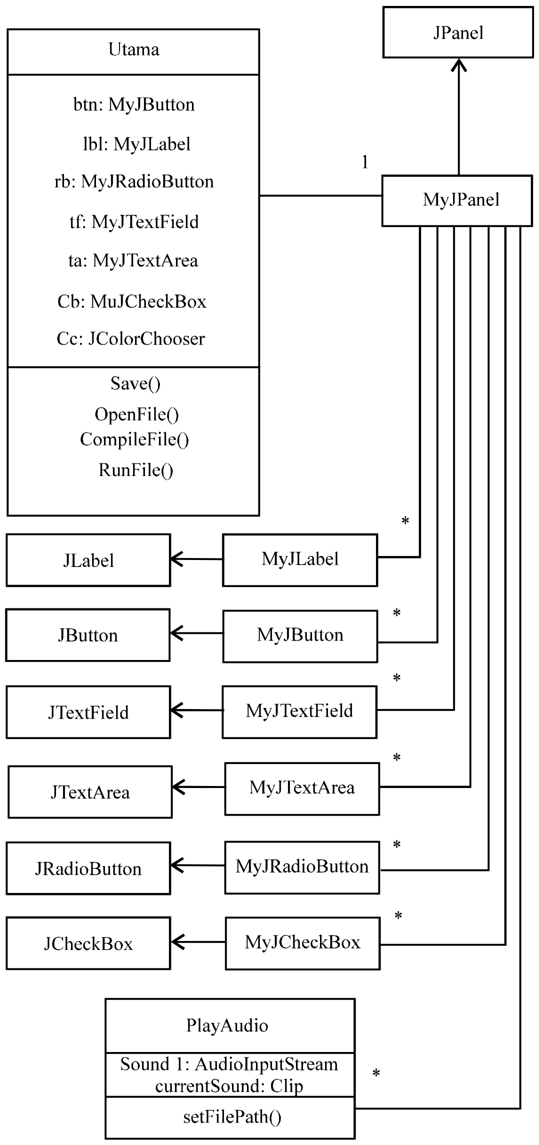


Fig. 6: Class diagram

code is freely reusable whether for developing commercial or non-commercial software. NetBeans technology is standards-based and it is also an open source software.

Although, these three IDEs provides a lot of facilities to support block creation, analysis of these tools indicates that they do not support all of the required functionalities. Thus, there is a need for us to develop a special purpose IDE for block creation.

Software design: There are nine classes needed to develop the proposed IDE. These classes are Main class, PlayAudio, MyJButton, MyJLabel, MyJRadioButton, MyJTextField, MyJTextArea, MyJComboBox and MyJPanel. The class diagram for the IDE is shown in Fig. 6.

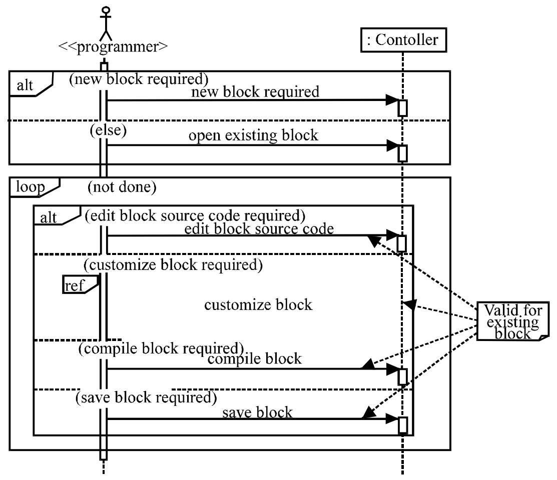


Fig. 7: Sequence diagram for creating a new or opening an existing block container

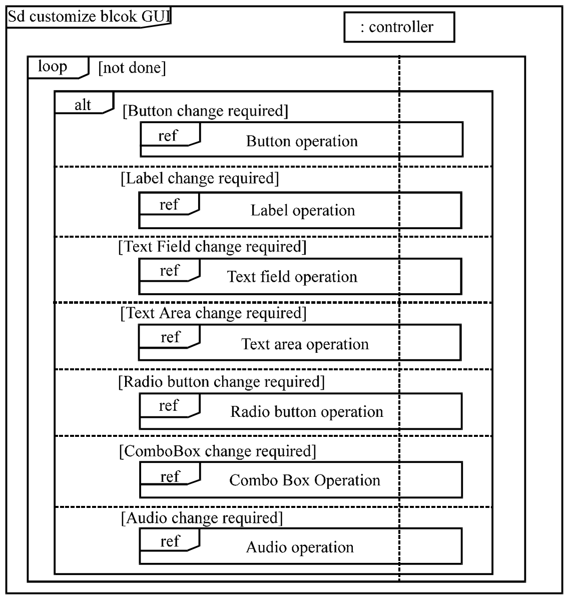


Fig. 8: Sequence diagram for block customization

The interaction between these classes can be shown by using three main sequence diagrams: a sequence diagram for create or open existing block container, a sequence diagram to describe block customization and a sequence diagram to describe customization of block GUIs. The first sequence diagram is shown in Fig. 7. As shown in Fig. 7, activities involved in creating a new block container include edit block source code, customize block, compile block and save block.

The second sequence diagram is shown in Fig. 8. There are three types of customization: customization of the block GUI, customization of the block behavior and customization of the block interfacing.

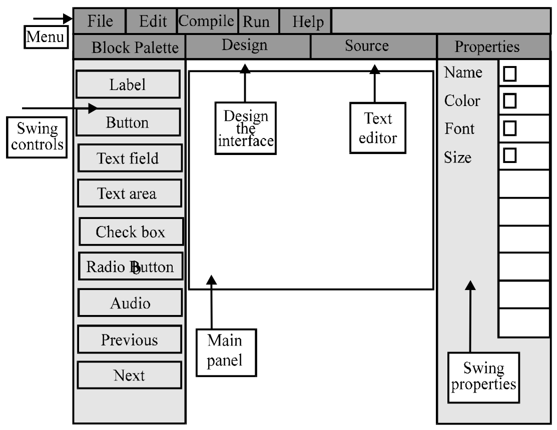


Fig. 9: User interface for the IDE for block creation

User interface design: Figure 9 shows the user interface of the IDE for block creation. As shown in figure, the user interface consists of six sections: the menu section, swing controls section, design section, text editor section, main panel section and swing properties section.

There are five pull-down menus provided to support the functionalities of the IDE. Menu files consist of a five sub-menus, menu edit consists of three sub-menus. The other two menus are compile and run. The swing controls section consists of nine elements: label, button, text field, text area, check box, radio button, audio, previous and next. The working area consists of two views: design view and code view. For the design view, a block developer can drag and drop swing components into the design area in order to create the block user interface. The block developer can also edit properties of a component. The last section, swing properties consists of attribute of the instance that is generated from each component. The developers can specify attributes of the instance for the block. These properties can be modified by the end user.

Implementation OF IDE: The implementation of the IDE was carried out by using Java programming language. The overall implementation is done by using Netbeans 6.8 and Java JDK. JDK stands for Java Development Kit and it is a free software development package from oracle to implement the basic set of tools we need to write, test and debugging Java applications.

RESULTS AND DISCUSSION

The evaluation of the IDE is done by using case study approach. Five programmers have been asked to use the IDE to create blocks by going through the process step by step as shown in Fig. 10-14.

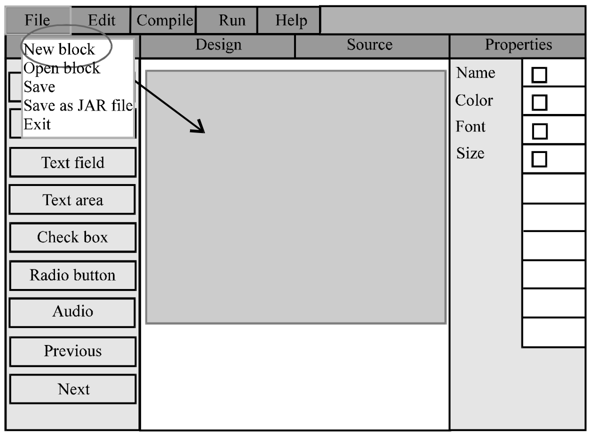


Fig. 10: Create a new block

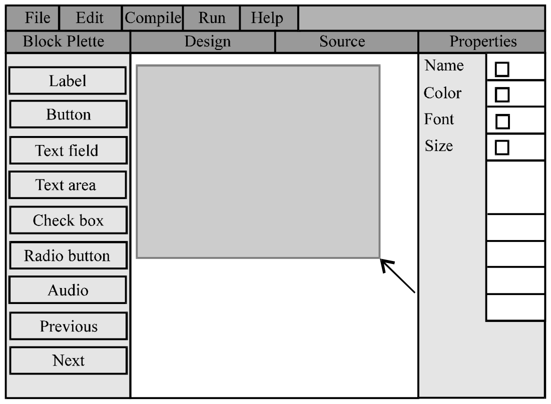


Fig. 11: Resize the block

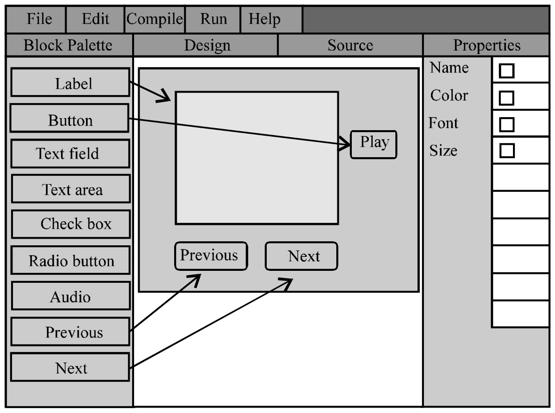


Fig. 12: Create the instance of element

As shown in Fig. 10, the first thing that needs to be done in order to create a block is to select new block in the menu file. A block container will then be displayed in the design area. The block developer can then modify the size of the block container as shown in Fig. 11.

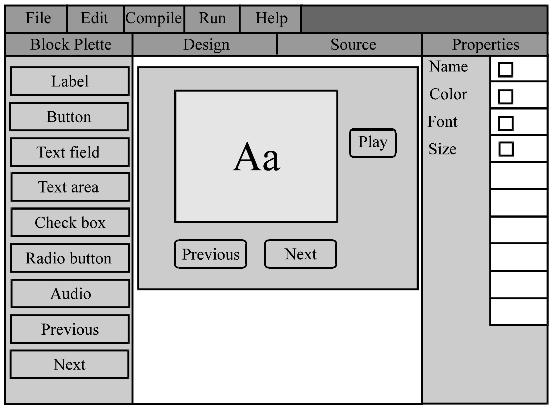
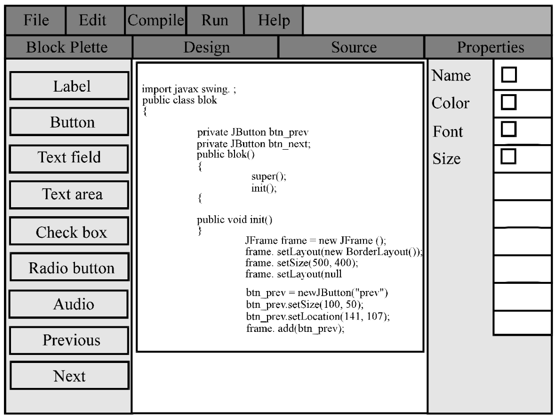


Fig. 13: Block has been produced



```
import javax swing. ;
public class blok
{

          private JButton btn_prev
          private JButton btn_next;
          public blok()
          {
                    super();
                    init();
          {

          public void init()
          }
                    JFrame frame = new JFrame ();
                    frame. setLayout(new BorderLayout());
                    frame. setSize(500, 400);
                    frame. setLayout(null

                    btn_prev = newJButton("prev")
                    btn_prev.setSize(100, 50);
                    btn_prev.setLocation(141, 107);
                    frame. add(btn_prev);
```

Fig. 14: Code editor environment

To insert an instance of a component into the block container, a block developer can click on the selected component on the left side and the instance of this component will appear in the design area. Attributes for the instance will then be displayed. The values of these attributes value can be changed if necessary.

Figure 13 shows the block that has been completed. This block consists of the labels Aa, button Previous and Next and button Play to play the audio. This block must then be compiled by clicking the compile button on the menu file to generate a class file. It can then be executed by clicking the run button. The block will be saved as class file and the properties will be saved as txt file. The block can then be packaged as a jar file by selecting Save as JAR file. The code view allows a developer to view the Java code of the block as shown in Fig. 14.

CONCLUSION

In the study, researchers have described the design and implementation of an IDE for the block creation. The purpose of this IDE is to support the block based software development approach. Although, blocks can be created by using currently available Java IDE such as Netbean, the availability of specialized IDE for block creation enables blocks to be created faster and with better quality.

REFERENCES

Bennett, D.W., 1995. The promise of reuse. Object Magaz., 4: 5-68.

Chen, Z. and D. Marx, 2005. Experiences with eclipse IDE in programming courses. J. Comput. Sci. Colleges, 21: 104-112.

Cheung, H.W., 2004. The impact of component-based technology on the role of user in traditional software development. 21st Comput. Sci. Seminar.

Goodell, H., 1998. End user programming. Proceedings of the IEEE Symposium on Visual Languages and Human-Centric Computing, September, 1998, Dallas, Texas, pp: 215-222.

Hopkins, J., 2000. Component primer. Commun. ACM, 43: 27-30.

Lieberman, H., F. Faterno, M. Klann and V. Wulf, 2006. End-User Development: An Emerging Paradigm. Kluwer Academic, The Netherlands.

Monson-Haefel, R., 2000. Enterprise JavaBeans. 2nd Edn., O'Reilly and Associates, California, Pages: 472.

OMG, 1996. The Common Object Request Broker: Architecture and Specification. Object Management Group, Salt Lake City, UT, USA., Pages: 177.

Rogerson, D., 1997. Inside COM: Microsoft's Component Object Model. Microsoft Press, Bellevue, Washington, Pages: 376.

Sun Microsystems, 1997. JavaBeans specification. Sun Microsystems Inc., http://www.cs.vu.nl/~eliens/documents/java/white/beans.101.pdf.

Zin, A.M., 2011. Block-based approach for end-user software development. Asian J. Inform. Technol., 10: 249-258.