An IDE for Java with Multilevel Hints to Develop Debugging Skills in Novices

Jesna A.A, Cochin University of Science and Technology, India
Renumol V.G, Cochin University of Science and Technology, India

The IAFOR International Conference on Education- Dubai 2016
Official Conference Proceedings

Abstract
Novices find it difficult to learn programming. In order to write a program, they have to learn the basic concepts of programming along with the syntax and semantics of a programming language. One of the important tasks in programming is debugging. To become good programmers, novices need to have good debugging skills. Typically it is difficult for novices to understand and rectify the syntax errors from the compiler-generated error messages, which they encounter during the debugging process. Professional Integrated Development Environments (IDEs) are not novice-friendly in debugging. Hence an IDE for novices has been designed and implemented for Java programming. It provides multilevel hints for the compiler-generated error messages, which will make the debugging process easier. A preliminary evaluation of the tool among engineering graduates is promising. This paper explains the design and implementation of this IDE.

Keywords: Java, Compiler error messages, Debugging, Integrated Development Environment, Novice programmer.

Introduction

Programming is a difficult task especially for novices. For writing programs a novice must acquire the programming concepts, along with syntax and semantics of a programming language. Listening to lectures helps novices to acquire knowledge, but only through practice he/she can succeed in the programming process. Novices face diverse difficulties when they start writing computer programs.

If a novice makes errors in his/her program then on compilation, the compiler generates error messages. So on seeing this error message he/she has to understand what this error message means, what is the reason for this error, how to correct this error, and learn how to avoid this error thereafter [1].

Understanding compiler-generated error messages is one of the main difficulties a novice faces during programming [2]. Because most of the times the compiler generated error messages do not match with the students level of programming ability or knowledge [1].

Different students have different ability in programming. Some students are good in syntax and semantics of programming language but they do not have an ability to solve the problem through a computer program. Some are good in problem solving but they are weak in programming language syntax. Some are average in both and some are weak in both.

A novice makes an error due to many reasons. Most of the errors are due to lack of programming experience and due to misunderstanding of the programming concepts (including syntax and semantics of programming language). Whereas the frequent errors such as, failing to close a pair of parenthesis, quotations, missing semicolon etc. are mainly due to carelessness [3]. Even though the teacher explains the error, the students repeat it. During lab sessions students spend a lot of time to rectify compiler-generated error messages for the programs that are logically correct [2].

Another difficulty faced by novice programmers is in using an Integrated Development Environments (IDEs). Most of the IDEs are developed for professional programmers [4]. Many studies have been carried out in this field to reduce the complexity of IDEs. Professional editors such as Visual Studio, Eclipse, Netbeans etc. are too complex for novices. Due to that reason many pedagogical tools are designed to make it easier for novice programmers [4]. BlueJ is one of the popular IDEs designed for novices for Java programming language [6][7][9]. It helps novices in learning object oriented concepts. In BlueJ packages are automatically created when classes are brought from different directories. Hence students use the ‘import’ statements very rarely. There is an option to add ‘main’ function in a class. Its visual environment helps students to interact with already existing classes. This may create a negative impact on students for writing programs of their own. BlueJ claims that they provide support for the compiler error message, but the help provided are not that much useful [7]. Also in this IDE user can see only one error at a time [6]. After correcting the error user has to compile again to see the next error if present, which is tedious. Studies have shown that simplified pedagogical development environments can reduce anxiety and uncertainty that novice programmers have [4].

Analysis

From a novice programmer's point of view, compiler-generated error messages are complicated and difficult to understand [2]. Sometimes these error messages are inadequate and do more harm to the debugging process [3]. For example, suppose a student misspells a variable named 'length' as 'lenght' in a java program "sample5.java", then the Java compiler displays the following error message [3]:

sample5.java:20: cannot find symbol
symbol: variable lenght
Location: class sample5

In this case, the compiler-generated error message creates enough confusion to the beginners; also it does not help them to fix the error. Another issue is that the error message "cannot find symbol" will appear in the following contexts also:

1. The variable is not declared, but is used.
2. The variable is declared, inside a conditional block and is used outside of that conditional block.
3. If there is any mistake in the spelling of the keywords like int, float, String etc.

Consider another Java error message, "sample6.java:3: class, interface, or enum expected". This error message will appear in the following cases:

1. When the programmer misspells the keyword "class".
2. When the programmer misses '}' at the end of program.
3. When there is a mismatch in '{' and '}' braces.
4. When the programmer forget to add 'class' in the program.

Consider another 'javac' error message which is experienced by almost all Java programmers, that is "; expected". This is one of the trouble-free error messages, which can be easily understandable by novice programmers. From this error message, programmer infers that the statement requires ";" at the end of the statement shown along with the error. But the same error message is displayed due to some other problems even if you have already added semicolon at the end of the statement. Some of the reasons, which novices may not be familiar with are given below:

1. If this error is occurring where there should not be a semi-colon (e.g. in the middle of a method declaration), this could be a sign that the previous method does not end properly.
2. If there is any mistake in the main function or if you forget to add '{' after main function.
3. If there is any mistake in the keywords like throws, static, public, private, protected, new, assert, for, while, if etc.
4. If you use keywords as your class name.
5. If you forget to close double quotes in the statement
6. If you forget to open a '(' or there is a mismatch in '(' and ')' in the statement shown with error etc.

This motivated the authors to add hints for the most common compiler-generated error messages in Java to help novices to understand the different reasons for the same error message. Hence, in order to find the most common errors, the authors have conducted a small survey among the 4th semester B.Tech (Information Technology) students of School of Engineering, Cochin University (CUSAT). They were asked in the "Data Structure using Java" lab, to note down the syntax errors each one encounters. After each lab session the lists were collected from the students. Based on the frequency of the same error message, the common error messages are determined.

The most common errors encountered by the students during their java lab classes are:

1. cannot find symbol
2. ;expected
3. class, interface or enum expected
4. illegal start of expression
5. incompatible types
6. invalid method declaration; return type required
7.)expected
8. <identifier> expected
9. reached end of file while parsing
10. }expected
11. not a statement
12. Possible loss of precision
13. unclosed string literal
14. Variable NAME Might Not Have Been Initialized
15. Null Pointer Exception
16. Array Index Out of Bounds Exception
17. file not found
18. Missing return statement
19. malformed floating point literal
20. illegal start of type
21. package system does not exist
22. {expected
23.]expected
24. (expected
25. inconvertible types
26. (or[expected
27. unreported exception java.io.IOException; must be caught or declared to be thrown

The above examples show that the compiler-generated error messages are not very specific and are not helpful for novices. Different studies have been conducted to find out the most common errors students/novices face during the java programming [5][6]. The studies showed that the syntax errors dominate run time and logic errors, in novice programs.

Good error messages help programmers to understand the errors easily and also help them to fix the error. Due to poor error messages, students might misunderstand the error message and lead them to make changes in the source code without

understanding the real cause of the error [1][10]. Good error messages act as learning aid for novices [1]. This motivated the authors to design and develop a better debugging tool for novices.

Design and Implementation

There are many factors to consider when designing an IDE for novices. Simplicity is a significant design consideration. Many pedagogical tools that are already available provide customized error messages, which are different from standard error messages [4]. One drawback of this approach is that students find it difficult when they have to deal with the original compiler error messages. Another issue is that, the professional IDEs like Netbeans, Eclipse etc. provide a lot of options. Typically these are not required for a novice programmer [8]. Some of them even provide coding templates, which may not help the students to learn the programming language syntax.

Figure 1 shows the interface a user sees when he/she selects the option File→New Project in the Netbeans IDE. The figure shows the package declaration, the class declaration and the main function that are automatically available when a user creates a new project. For a novice programmer these code templates create a negative impact that may hinder their learning process [8].

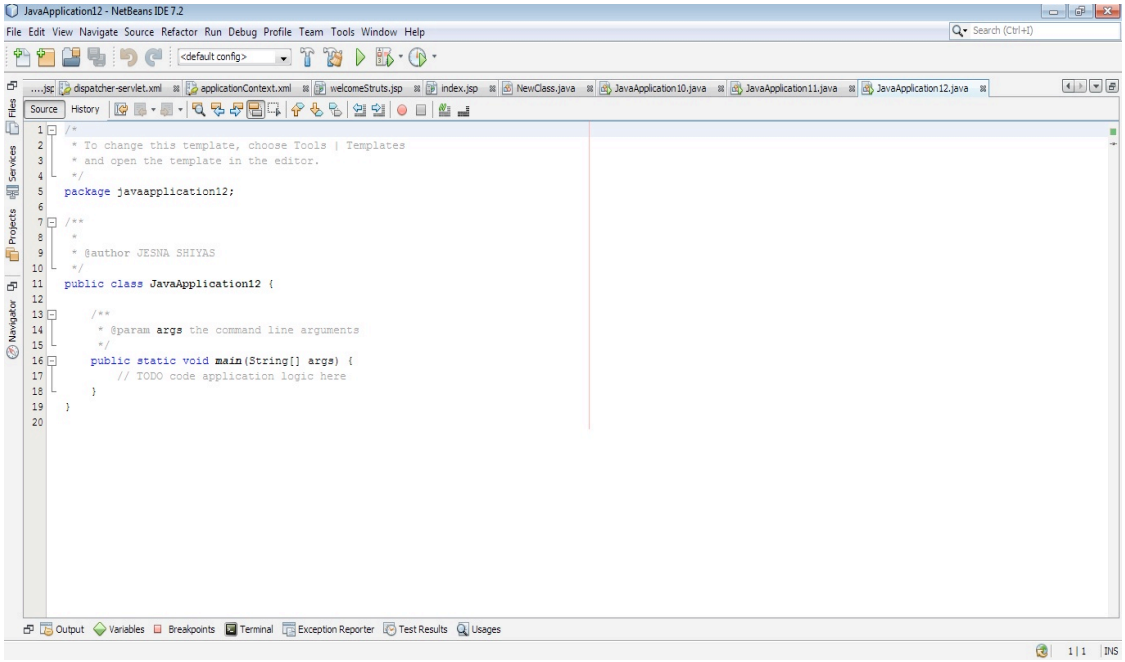

Figure 1: Screenshot1 of NetBeans IDE

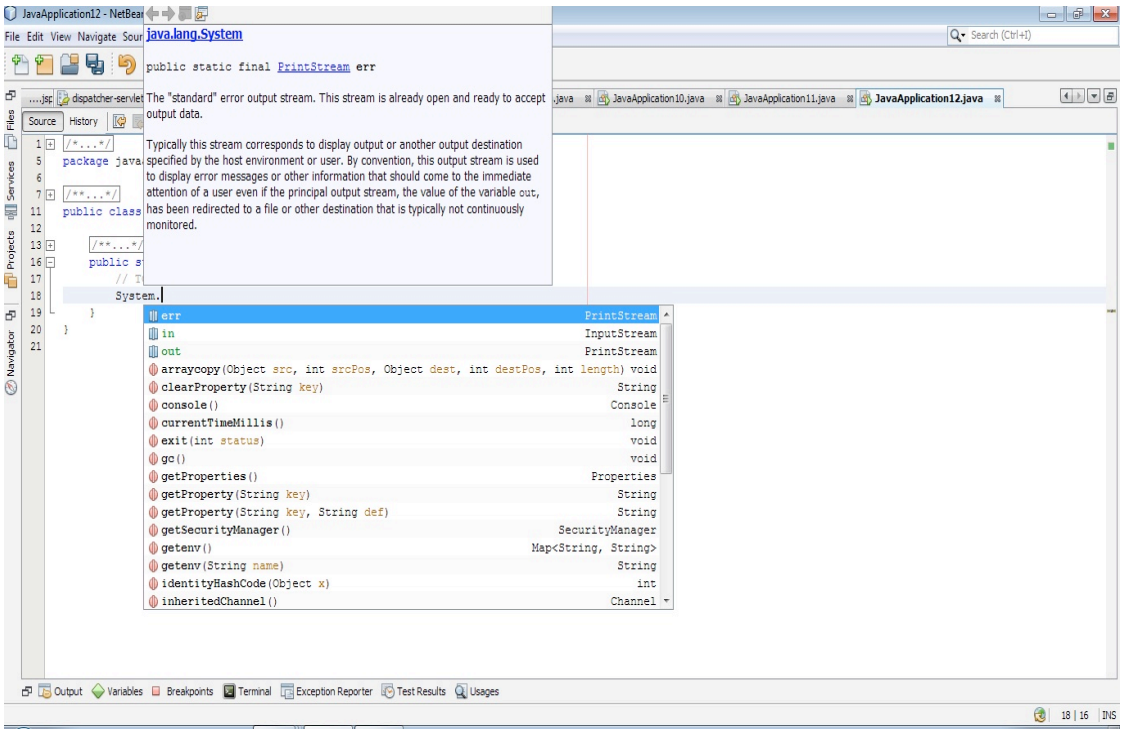

Figure 2: Screenshot2 of NetBeans IDE

Another example is shown in figure 2, which shows the suggestions provided by Netbeans, when a user types a Java program. If the novices use these suggestions blindly then they may not learn to write the program statements themselves. Understanding the syntax and semantics of a language has a central role in the programming language learning process [8]. The code templates and the suggestions provided by the IDEs are extremely useful for professional programmers, but not for novices. These features shall help the professionals to find and correct careless mistakes and develop the applications more quickly [8].

The above facts have been considered in order to design and develop an IDE for novices to do Java programming. The implementation is done using the C#.net. The IDE consists of simple interfaces and menu options to make the learning process easy. There are options for novices to type new programs, to save programs, to open already saved programs, to edit programs, to search a word, to replace a word with another word etc. It uses 'javac' compiler for compilation and a database that stores user-friendly hints for compiler-generated error messages. When a student types a program, the keywords of Java get highlighted. It helps students to check whether they type keywords correctly or not. For each student a separate folder is created in his/her username. So when a student logs in, the programs he/she has already saved will get displayed on the right side panel of the IDE. After typing a Java program, the student can compile the program by clicking on the 'compile' option. If the program contains errors, then the compiler will generate error messages as usual. But, when the student clicks on each error, the corresponding line in the program will get highlighted and a message box will pop up with user-friendly hints for the corresponding error message. If the student is not able to rectify the error with the help of the initial hint, he/she can get the next level of hint. Based on the error, 4 to 5 levels of hints have been provided. The hints will help the novice students to

understand the reasons for the error so that they can easily correct them. The reasons are provided based on the extensive observation of students' experience in the Java programming lab. The error messages and the hints provided in the IDE shall act as a learning aid for novices. On the other hand, if the compilation is successful, the student can run the program through command prompt, which is a panel inside the IDE, as shown in figure 4. The purple color window shows the output of the program.

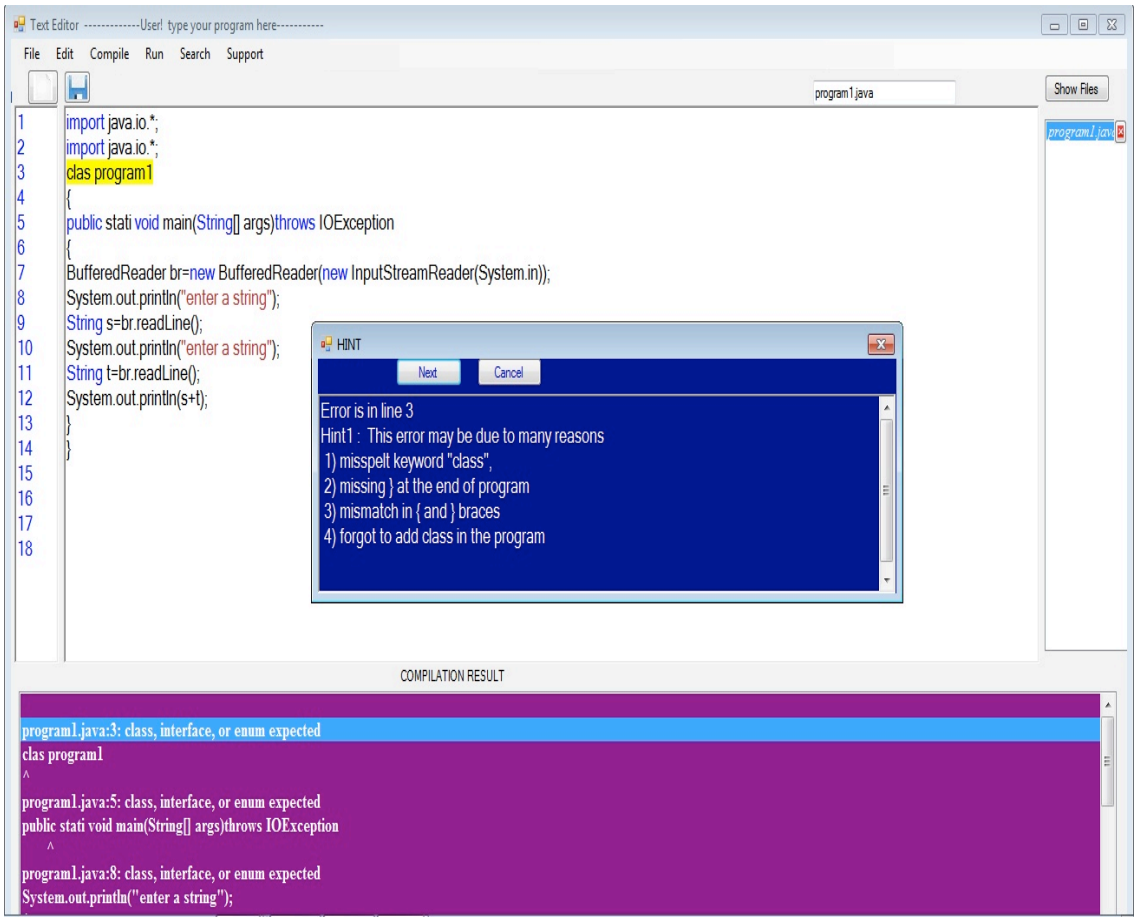

Figure 3: Screenshot of IDE (compiling phase)

Figure 3 shows the IDE interface for the compilation phase. Novices can use them to create, save, edit, compile, debug and run their programs. The yellow color shows the line in which the error occurred. The purple colored area, at the bottom of the IDE shows the original error messages generated by the 'javac' compiler. When a student clicks on each of these error messages, he/she can see the hints, in the blue message box. If the student needs more hints then he/she has to press the Next button in the message box. At the right hand side of the IDE, a list of files already saved by the student is displayed.

In some pedagogical tools, users see only the modified/simplified error messages, instead of the original compiler error messages. But this IDE provides both compiler-generated error messages and additional simple hints to rectify the error.

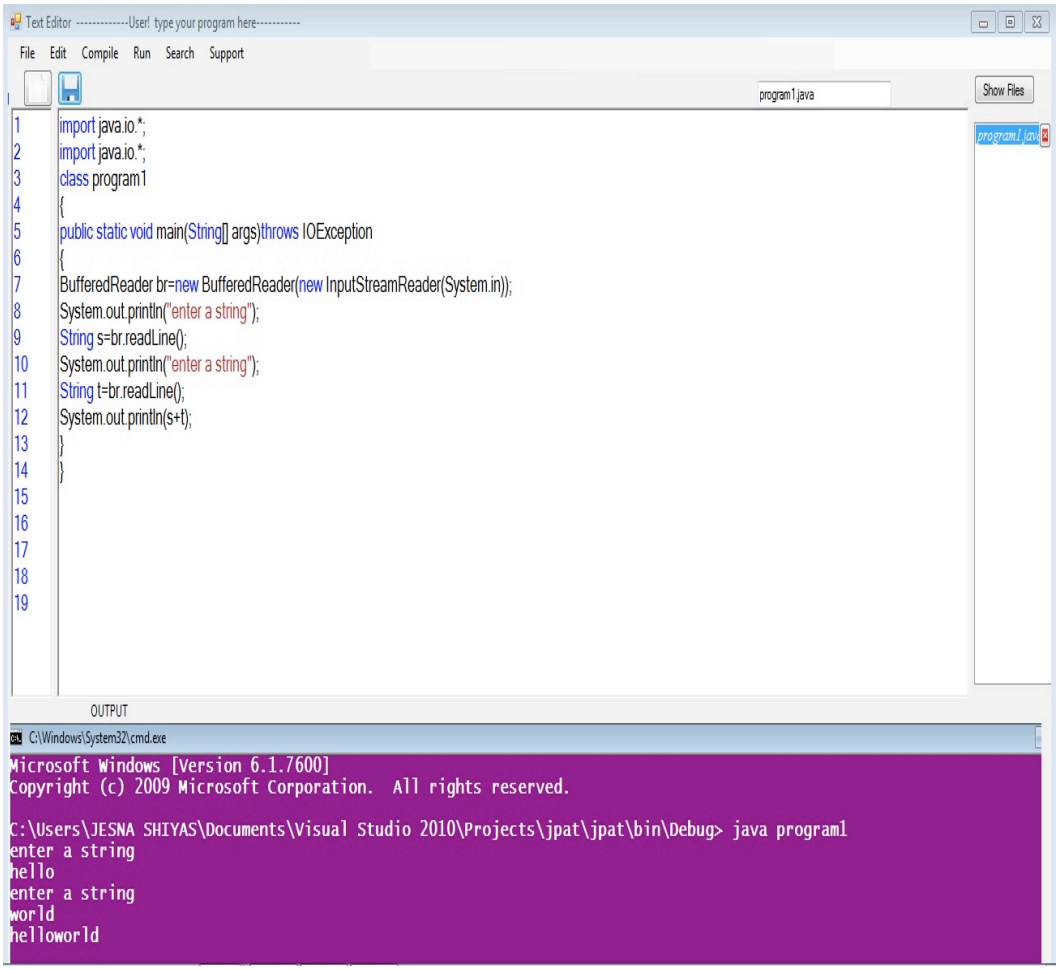

Figure 4: Screenshot of IDE (running phase)

The IDE also provides help for the various java keywords, as shown in figure 5. The green color window shows the help for the keyword which is selected. The following 53 keywords have been included in this IDE:

1. import, 2. static, 3. class, 4. interface, 5. extends, 6. implements, 7. abstract, 8. assert, 9. Boolean, 10. break, 11. case, 12. catch, 13. const, 14. continue, 15. default, 16. do, 17. double, 18. else, 19. enum, 20. final, 21. finally, 22. float, 23. for, 24. if, 25. instanceof, 26. int, 27. long, 28. native, 29. new, 30. package, 31. private 32.protected, 33. public, 34. return, 35. short, 36. strictfp, 37. super, 38. switch, 39. synchronized, 40. this, 41. throw, 42. throws, 43. transient, 44. try, 45. void, 46. volatile, 47. while, 48. false, 49. null, 50. true, 51. goto, 52. char, 53. byte

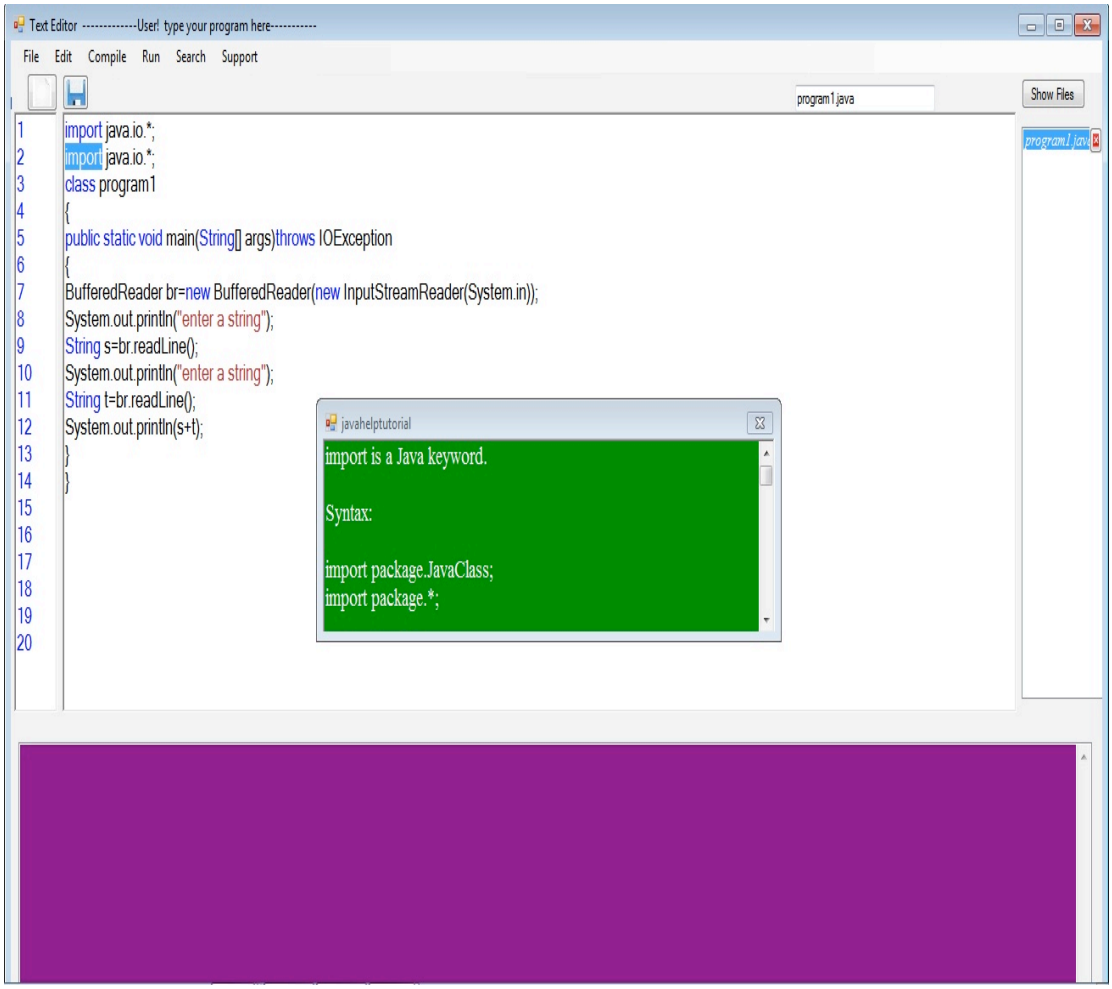

Figure 5: Screenshot of IDE (showing help for the keyword “import”).

If the students select a keyword and right click on the mouse button a window pops up with the help for the keyword selected. The help includes, what is the use of the selected keyword, what is the correct syntax of using the keyword, example showing the use of the keyword and a description about the keyword. It helps novices to clear the doubts regarding the keywords used in their Java program.

Figure 6 shows the support the IDE provides for various Java concepts, like polymorphism, inheritance and its types, input/output, and runtime errors. If a student selects inheritance from the list, then a menu with two options will appear which shows what is inheritance and the different types of inheritance in Java. If he/she selects different types of inheritance, then a menu showing single, multiple, multilevel, hierarchical, and hybrid inheritance appears. From the menu if he/she selects ‘Hybrid Inheritance’, then a window appears with the details of the Hybrid Inheritance, as shown in figure 7.

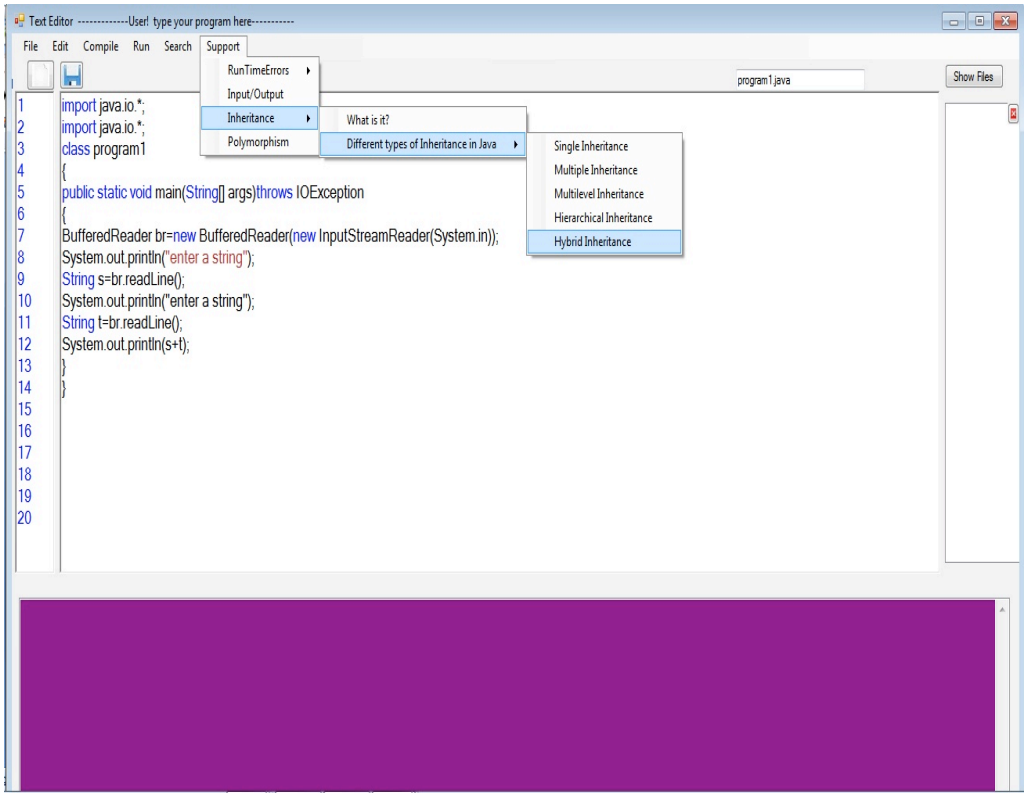

Figure 6: Screenshot of IDE (showing the support option).

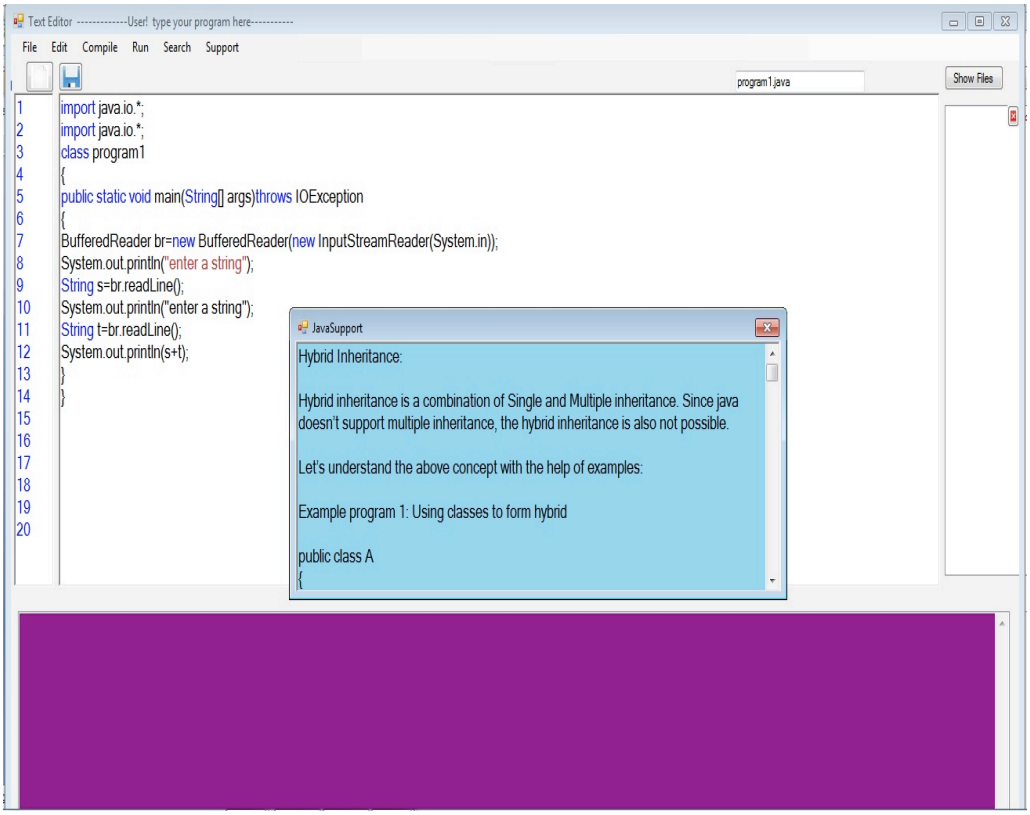

Figure 7: Screenshot of IDE (showing support for Hybrid Inheritance)

An example for the hints for the error message "incompatible types" is shown in Table 1. A database is created to store the error messages and their corresponding hints. Currently it includes 27 common error messages and their hints. The database can be extended with additional error messages when they are encountered.

Table 1. ERROR AND IT HINTS

Error	Hint1	Hint2	Hint3	Hint4
incompatible types	JAVA checks that the type of an expression is compatible with the type of the variable in an assignment statement and this error will result if they are incompatible	This error occurs when there are type issues in your program. It is possible to convert between some kinds of types; for example: you can freely convert a char to an int and vice versa, and you can also convert a double to an int with some typecasting. However, you cannot convert between primitive types and objects such as String.	A variable is being passed a data type that it is not designed to hold. Example: int i = "hello";	Check the return types of any methods you are using. If the message returns an int, for example, make sure that you're not trying to store it in another type of variable, for example String

Result

In order to validate the developed IDE, it has been given to a set of B.Tech 4th semester students in the authors' university. Students were also provided with a feedback form to note down their comments and suggestions about this IDE.
The following are some of the comments from the students:

- Easy to debug errors.
- Much easier to understand and offers better learning experience to beginners.
- Easy to find in which line the error occurs.
- Find and replace options provided help to find out the word easily and to replace the repeated words at the same time.
- Help provided for the java keywords are useful
- Hints provided for error correction is very useful.
- More helpful as it gives us more knowledge about the errors and how to correct them.

These comments warrant a further study about this IDE by allowing more students to use it in the forthcoming semesters.

Conclusion

Understanding the Java compiler error messages is very difficult for a beginner. Hence providing understandable hints to each error message will help novices to recognize the reasons for the errors and to rectify them. This paper discusses the design and implementation of a novice-friendly IDE for java programming. The easily understandable hints provided in the IDE, for the most common javac error messages helped students to overcome the difficulties they experienced before. The comments from a set of students who have used it, confirm the effectiveness of the IDE. The main attraction of this IDE is simplicity. Students can easily use this tool without any training or help.

Acknowledgement
We would like to express our sincere gratitude to the students who have participated in the testing of this IDE.

References

Traver, V.J. 2010. *On compiler error messages: what they say and what they mean*. Technical Report. Computer Languages and Systems Department. Jaume-I University.

More, A., Kumar, J., Renumol, V. G. 2011. *Web Based Programming Assistance Tool for Novices***,** 2011 IEEE International Conference on Technology for Education.

Flowers,T., Carver, C., Jackson,J. 2004. *Empowering Students and Building Confidence in Novice Programmers through Gauntlet*. 34th ASEE/IEEE Frontiers in Education Conference, Savannah GA, Oct 2004, T3H-10 – T3H13.

Truong, N., 2007. *A web-based programming environment for Novice Programmers*, Ph.D. dissertation, Faculty of Inform. Technology, Queensland University of Technology, Queensland, 2007.

Jackson, J., Cobb, M., et al. 2005. *Identifying top Java errors for novice programmers*. Proceedings of the Frontiers in Education Conference (2005), T4C–24.

Thompson,S.M.,2004. *An Exploratory Study of Novice Programming Errors and Experiences*. Unpublished MSc dissertation. Victoria University 2004.

Hagan,D., Markham,S.2000.*Teaching Java with the BlueJ environment*. Proceedings of Australasian Society for Computers in Learning in Tertiary Education Conference (2000).

Gross, S., Strickroth , S. , Pinkwart, N., Le, N. T. 2013. *Towards Deeper Understanding of Syntactic Concepts in Programming*. In The First Workshop on AI-supported Education for Computer Science (AIEDCS 2013), p. 11. 2013.

Jadud, M. C. 2006 .*Methods and tools for exploring novice compilation behaviour*, In Proceedings of the Second International Workshop on Computing Education Research, ICER '06, pages 73{84, New York,NY,USA, 2006. ACM.

Marceau, G., Fisler, K., and Krishnamurthi, S.2011. *Mind your language: On novices' interactions with error messages*. In Proceedings of the 10th SIGPLAN Symposium on New Ideas, New Paradigms, and Reflections on Programming and Software, ONWARD '11. ACM, New York, NY, USA, 2011, pp. 3–18.

Contact email: jesnaasharaf661@gmail.com
Contact email: renumolvg@gmail.com