Novice Helper: Eclipse Integrated Development Environment Plugin to Support Novice Programmers

Caroline Berger

***Abstract*—Novice Helper, an Eclipse Integrated Development Environment (IDE) Plugin aims to teach young students about computer science and programming by translating Java's compiler error messages into age appropriate debugging advice embedded with computer science concepts. Unlike other programming comprehension tools targeted towards children that abstract a programming language into drag and drop blocks, Novice Helper does not intend to alter the Java programming language. Instead, Novice Helper works inside the Eclipse IDE and interacts with the Eclipse Problems View. Novice Helper is intended to be used in conjunction with a set of educational exercises that introduce a beginner to variables, types, loops, scope and control flow.**

Keywords*—Novice programmers, error message comprehension, Eclipse IDE Plugin

I. Introduction

Programming literacy has become increasingly important to primary school educators that want to prepare students for a future professional life immersed in technology. Tools like Scratch are created to introduce young students to programming concepts by reducing keyboard typing and supporting a colorful user interface. By abstracting programming languages such as Java and Python into blocks, students can create interactive and visual projects. While eliminating the need for typing is attractive for young students, Scratch also eliminates the informative and educational process of debugging. Scratch users do not encounter errors in their programs therefore miss the opportunity to learn from debugging [1].

After using solely drag and drop programs, a student may enter her first undergraduate computer science class, forced to relearn concepts that Scratch abstracted and thrown into the daunting task of debugging from foreign and cryptic error messages [1, 2]. Instead, students could program in text based editors that require debugging at a young age. Novice Helper is a proposed solution to this problem.

The Eclipse IDE is widely used by professional software developers and provides users with a graphical user interface and debugging tools. Eclipse is brimming with resources that aid development and, in turn, can overwhelm a novice programmer [3]. Yet, novice programmers may be able to work within the professional environment with the correct forms of support. By scaffolding the learning process, the student can incrementally use more and more of Eclipse's tools to transition from programing predefined exercises to more complex independent software projects. Novice Helper aims to provide support for novice programmers in understanding their error messages while working within the Eclipse IDE.

II. Relevant Research

Prior research pertaining to the topic of novice programming falls into two categories research on undergraduate students programming in textual languages and children programming in visual languages. Successes and draw backs in current programming tools were used as a guide in the design and implementation of the Novice Helper.

While there is little to no research that solely analyzes children programming in a text based language in a professional development environment, there are studies comparing text and visual programming languages. These studies highlight the pitfalls of teaching children to code solely with a visual based language and the necessity for more text based tools and research.

In a study that compares textual and visual programming languages for primary school children, students using visual programming languages were observed to have increased motivation scores during the course while the students using textual languages were observed to have no change in motivation scores [2]. While these findings highlight the importance of learner motivation, there is a significant limitation to this study. The results of this study were judged on motivation alone but, when assessing the effectiveness of a curriculum or learning tool, it is important to discuss the learning outcomes alongside with satisfaction. A different study discusses the challenges of a student who is transitioning from visual programming language to a textual programming language. The study explains the unlearning that must occur when a user transitions from Scratch to Python. Concepts, such as comparison are represented differently in each language [1]. Unfortunately, the study does not examine in depth the transition between little to no debugging in a visual language to tedious debugging in Python. Based on studies that analyze current programming tools for children, the results suggest the need for development and trial of new novice programming tools.

There is a wealth of studies and tools to assist undergraduate students in their debugging. While this age group is older than Novice Helper's intended audience, the findings from these studies influenced the design and development of Novice Helper. One study suggests against directly solving the users error and instead trying to explain the error in a general manner [3]. Current tools, such as BlueJ, Alice, and Expresso, define their own editors and are designed to help the user while she is coding. Alice adapts the Java programming language to be used by beginners. Unlike the other preexisting tools, Novice Helper is designed for novice programmers in a professional development environment. The tool is not designed to change the Java programming language and is meant to support the

user in a non-obstructive fashion when errors are encountered.

III. Solution

The Novice Helpers advice is created to be used with "5 Introductory to Java Programming Exercises for Children aged 9 through 12", a group of problems designed to introduce the user to print statements, variables and loops. The exercises are included in the Appendix.

A. The Error Dictionary

To form a comprehensive and appropriate error dictionary, findings from past studies on novice programmers' common errors was adapted to the context of the "5 Introductory to Java Programming Exercises for Children aged 9 through 12" problem set [4]. In addition, the error translations were reviewed by Dr. Taciana Pontual da Rocha Falcao, a Post Doctorate in Education at McGill University to help ensure that the error translations were appropriate for users aged 9 through 12 in their level of vocabulary and relevant prior knowledge.

The main difficulty while forming the Error Dictionary was deciding how far in depth to explain concepts. For example, String is capitalized because it is an object and not a primitive type. For a novice programmer, mistaking capitalization is easy because all the other types they will have seen (int, boolean, double, etc.) are lowercase. After discussing whether to teach the notion of objects to students through error messages, it was decided that this was too complex of a topic to be condensed into one line. Instead, this error message offers a quick fix of "String must be capitalized".

B. Software Implementation

Novice Helper is an Eclipse Plugin created in the Eclipse Plugin Development Environment. Classes of the Novice Helper extend the View eclipse extension and depend on the Problems View. There are three classes in the project: the Activator class, the NoviceHelperView class and the Translator class. The Activator is responsible for maintaining the plugin's lifetime. The entirety of the Activator's code is inherited from its parent AbstractUIPlugin. The NoviceHelperView class has a SelectionListener that listens to the clicks in the Problems View. The NoviceHelperView calls Translator and is responsible for setting up the positioning of the text. The Translator uses the ErrorDictionary.properties file to look up the different errors.

C. Design Patterns and Information Hiding

Eclipse Plugins are highly object-oriented in their implementation as well as their interaction with internal Eclipse concepts and structures. The Singleton, Observer, Adapter and Decorator Design Patterns are used in the implementation of the Novice Helper Plugin. The Singleton and Observer were used in the design and implementation of the Novice Helper Plugin. The Adapter and Decorator was used to work with inner IMarker resources and ViewPart that are predefined by the Eclipse Architecture. The UML class diagrams can be found in the Appendix.

The Singleton Design Pattern is implemented by the Activator class which has a private instance of the plugin and a public getDefault method ensuring that there is only one instance of the plugin. The Observer Design Pattern is implemented with the NoviceHelperView and the ProblemListener. The NoviceHelperView registers the ProblemListener to observe the Eclipse Problems View. When the ProblemListener observes a change in selection, it calls the showSelection method of NoviceHelperView, changing what is shown in the NoviceErrorView. The NoviceHelperView is a concrete observer as well as a manager of the ProblemListener because if the NoviceHelperView is closed, then the listener will be deleted. The ProblemListener should not exist without the NoviceHelperView therefore the ProblemListener is composed inside the NoviceHelperView class.

The Adapter Design Pattern is used inside the NoviceHelperView class in the showSelection method. A selection to the Problems View is of type ITreeSelection. Through reflective program, it was discovered that the actual IMarker representing the error is the second segment of the first path (paths[0].getSegment(1)). To access the message associated with IMarker, it must be cast into an IAdaptable then call the getAdapter(IMarker.class) which returns the IMarker object. The Decorator pattern is used in the NoviceHelperView to add design and text to the ViewPart.

Design Patterns are a large part of the inner workings of the Eclipse Architecture. To access IMarker objects and create an Eclipse View, it is essential to understand how to leverage design patterns. Novice Helper is modular in design because of its use of design patterns.

To perform information hiding, methods that could be private were declared as private. Methods in the Activator class and NoviceHelperView that were declared public by their parent class were declared public to ensure proper plugin functionality. The Activator class hides the instance of the plugin and uses a getDefault method. While programming an Eclipse Plugin there are restraints on what is public based on which part of the Eclipse Architecture the plugin is extending, nonetheless, it is imperative to use proper design patterns that help encapsulate the Plugin's attributes and actions.

IV. Conclusion

The Novice Helper is designed to allow for the full expression of the Eclipse IDE and increase the understanding of error messages for novice programmers. The Novice Helper differs from other current programming tools because it does not alter the Java language. Instead, the Novice Helper supports a user's debugging process and aims to enhance the user's understanding of computer science concepts.

A. Software Improvements

Before deployment, the Novice Helper should undergo improvement. The Novice Helper could be improved to have a better ErrorDictionary.properties implementation by adding opening and closing double quotes to the untranslated error messages before they are sent to the properties file. This would improve the readability of the ErrorDictionary.properties file.

Another improvement would be to use a browser to display Novice Helper advice instead of using a supplemental Eclipse View. Checkstyle, an Eclipse Plugin throws errors when used within the Eclipse Plugin Development Environment with this specific plugin. Although, it has not been proven why this clash occurs, it could be because both Checkstyle and Novice Helper use some of the same extension points. In the future, this error should be further investigated in order for Checkstyle to help improve the code quality.

B. Future Research

Once the software has been improved, the logical next step for this research project is to perform a human study that tests the tool on children. First, a pilot study would take place to filter any blatant errors. The software and exercise set would be then improved upon for a formal human study to occur.

ACKNOWLEDGMENT

Thank you Dr. Taciana Pontual da Rocha Falcao for consultation throughout the project.

REFERENCES

[1] W. Robinson. From Scratch to Patch: Easing the Blocks-Text Transition. In Proceedings of the 11th Workshop in Primary and Secondary Computing Education, WiPSCE '16 2016, pages 96-99, New York, NY, USA, 2016. ACM. ISBN 978-1-4503-4223-0. doi 10.1145/2978249.2978265

[2] H. Tsukamoto, Y. Takemura and Y. Oomori. Textual vs. Visual Programming Languages in Programming Educatoin for Primary Schoolchildren. In Frontiers in Education Conference, FIE 2016.

[3] G. Marceau, K. Fisler and S. Krishnamurthi. 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! 2011, pages 3-18, New York, NY, USA, 2011. ACM. ISBN 978-1-4503-0941-7. doi 10.1145/2048237.2048241

[4] J. Jackson, M. Cobb and C. Carver. Identifying Top Java Errors for Novice Programmers. In *Proceedings of the Frotiers in Education Conference*, FIE 2005, doi 10.1109/FIE.2005.1611967

APPENDIX A

Name of Exercise	Java Compiler Error Message	Novice Helper Translation:
Hello World	Systm cannot be resolved	System must be spelled properly. Did you mean System.out.print() or System.out.println()?
Hello World	Sytm cannot be resolved	System must be spelled properly. Did you mean System.out.print() or System.out.println()?
Hello World	Syste cannot be resolved	System must be spelled properly. Did you mean System.out.print() or System.out.println()?
Hello World	Sys cannot be resolved	System must be spelled properly. Did you mean System.out.print() or System.out.println()?
Hello World	Syste cannot be resolved	System must be spelled properly. Did you mean System.out.print() or System.out.println()?
Hello World	Syntax error on token “)”, { expected after this token	A function must be have a { and } to define the body of the function. { and } wrap around a block of code.
Introduction to Variables	Type mismatch: cannot convert from int to boolean	Java is a typed language. Variables of type boolean can be assigned to true, false or null.
Introduction to Variables	Type mismatch: cannot convert from double to int	Java is a typed language. Variables of type int must be assigned to whole number values or null.
Introduction to Variables	Type mismatch: cannot convert from int to String	Java is a typed language. For a number to be considered a String, the number must have quotation marks around it " ".
Introduction to Variables	Type mismatch: cannot convert from boolean to int	Java is a typed language. Variables of type int must be assigned to whole numbers values.
Introduction to Variables	string cannot be resolved to a type	String must be capitalized.

APPENDIX A

Introduction to Loops	Syntax error on token"<", delete this token	Did you mean <=?
Introduction to Loop	Syntax error on token "}", delete this token	Blocks of code, such as classes and methods must have { and }. Are you creating a loop? for and while loops use { and } to define a block of code that is repeated.
Introduction to Loops	counter cannot be resolved to a variable	Variables must be initialized before they are used.
Introduction to Loops	count cannot be resolved to a variable	Variables must be initialized before they are used.
Introduction to Loops	c cannot be resolved to a variable	Variables must be initialized before they are used.
Loops Continued	i cannot be resolved to a variable	Variables must be initialized before they are used.
Loops Continued	Syntax error on token ">", delete this token	Did you mean >=?
Loops Continued	Syntax error on token "-", invalid AssignmentOperator	Assignment is from left to right, a variable on the left must be assigned to a value on the right.

APPENDIX
B

5 Introductory to Java Programming Exercises for Children aged 9 through 12

Exercise 1 First Program

Create a program that prints "Hello World!" to the console. Create a comment in your code using //. Comments are for human eyes only and are ignored by the computer. Do this task in the main method.

Exercise 2 Introduction to Variables

Tell a story about your family, favourite sport, what you did last weekend or something else entirely. Use your creativity! Declare and use at least 3 different type of variables to print a story to the console. This should be done in the main method. The following lines of code are an example of a story.

```
int christinaAge = 18;                                          \\int variables
int luciaAge = 23;
int lilyAge = 25;
String lilyCity = "Denver";                                     \\String variables
String luciaUni = "Technical University of Berlin";
boolean arePets = false;                                        \\boolean variable

System.out.println("In my family, there are four girls.\n"
                        + "Lily is the eldest "
                        + "she is " + lilyAge + ".\n"
                        + "She lives in " + lilyCity + ".\n"
                        + "Lucia is " + luciaAge + " and she attends the " + luciaUni +".\n"
                        + "Christina is "+ christinaAge + ".\n"
                        + "She lives with our parents.\n"
                        + "If someone were to ask me if I had pets, "
                        + "I would say " + arePets +".");
```

Exercise 3 Introduction to Loops

Loops are used in programming to preform a task or group of tasks multiple times. Loops can help programmers reduce the number of lines that he or she must write. Create a program that prints values from 0 to 10 using a while loop. Do this task in the main method. The basic structure of a while loop is as follows

```
while (counter <= endValue){ // while (counter < endValue){ is also valid code
    //do something
    //increase the counter
}
```

Exercise 4 Loops Continued

For loops can be used to create the same task as while loops. Create a program that prints 10 to 0 (decreasing order). Do this task in the main method. The basic structure of a for loop is as follows.

```
for ( initialize value; check value; update value){
    //do something
}
the following program produces the same output as problem 3
for( int i = 0; i <= 10; i = i+1 ){
    System.out.println(i);
}
```

APPENDIX
B

Exercise 5 Subtleties of Variables & Introduction to Methods

Print the value of x and y. Swap the values of the variables x and y, so that whatever was in x is now in y and whatever was in y is now in x. Print the value of x and y again. Do this in the main method.

```
Sample output:
Before swap:
x = 0
y = 1
After swap:
x = 1
y = 0
```

Challenge Problem

Hello secret agent, you've been assigned to a top secret programming challenge. The provided class called ChallengeProblem is filled with errors. Fix the errors in order to decode a secret message. Run the program to view this message. Good luck!

Sources

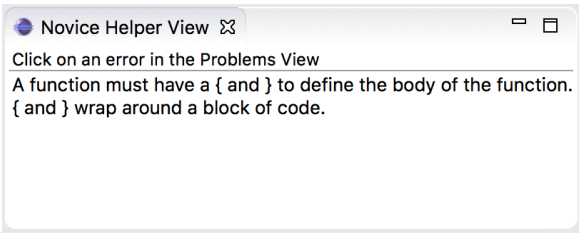
Guzidial, M., Ericson, B. (2005). *Introduction to Computing and Programming in Java: A Multimedia Approach.* Upper Saddle River, NJ: Prentice Hall.

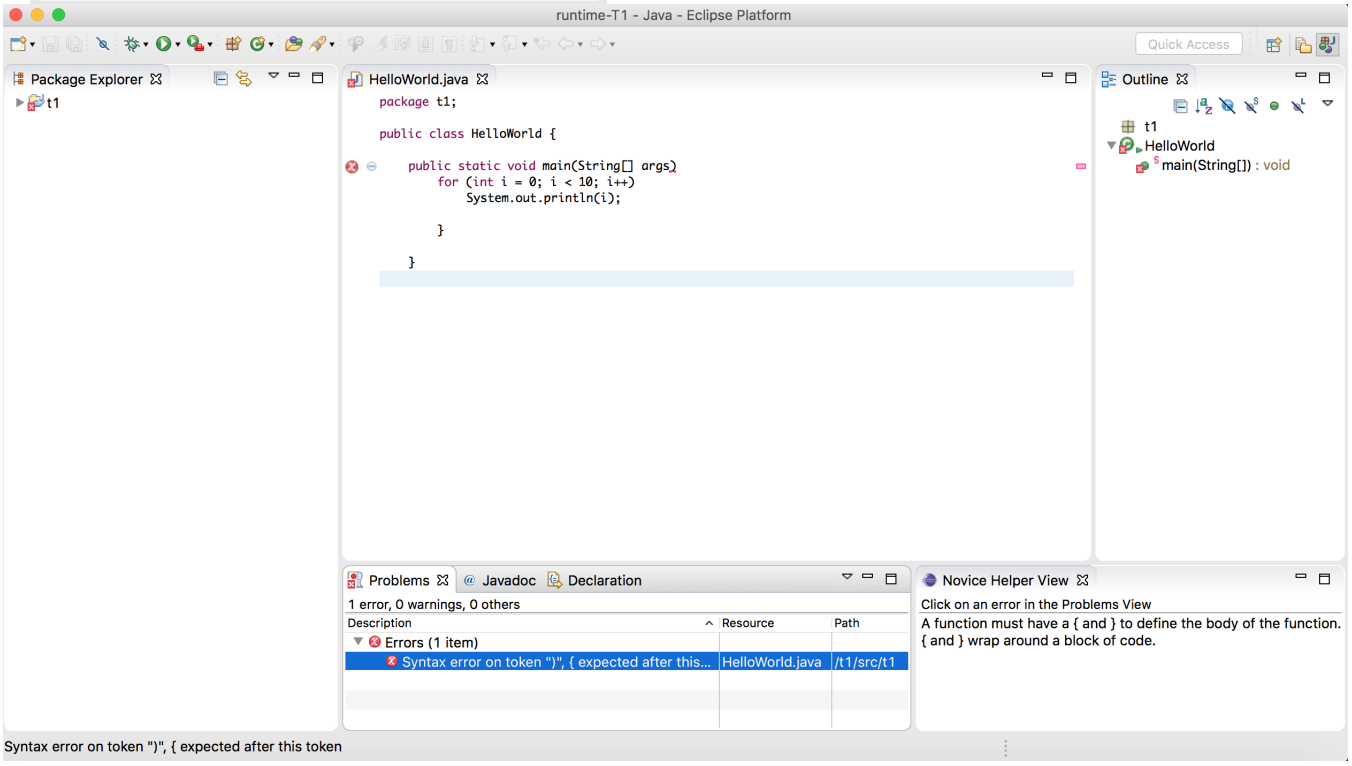
Lister, R., Leaney, J. (2003). 5th Australasian Computer Education Conference. *First Year Programming: Let All the Flowers Bloom.*

Pomerantz, D. (2015, October 5). *Assignment 1* COMP 202.

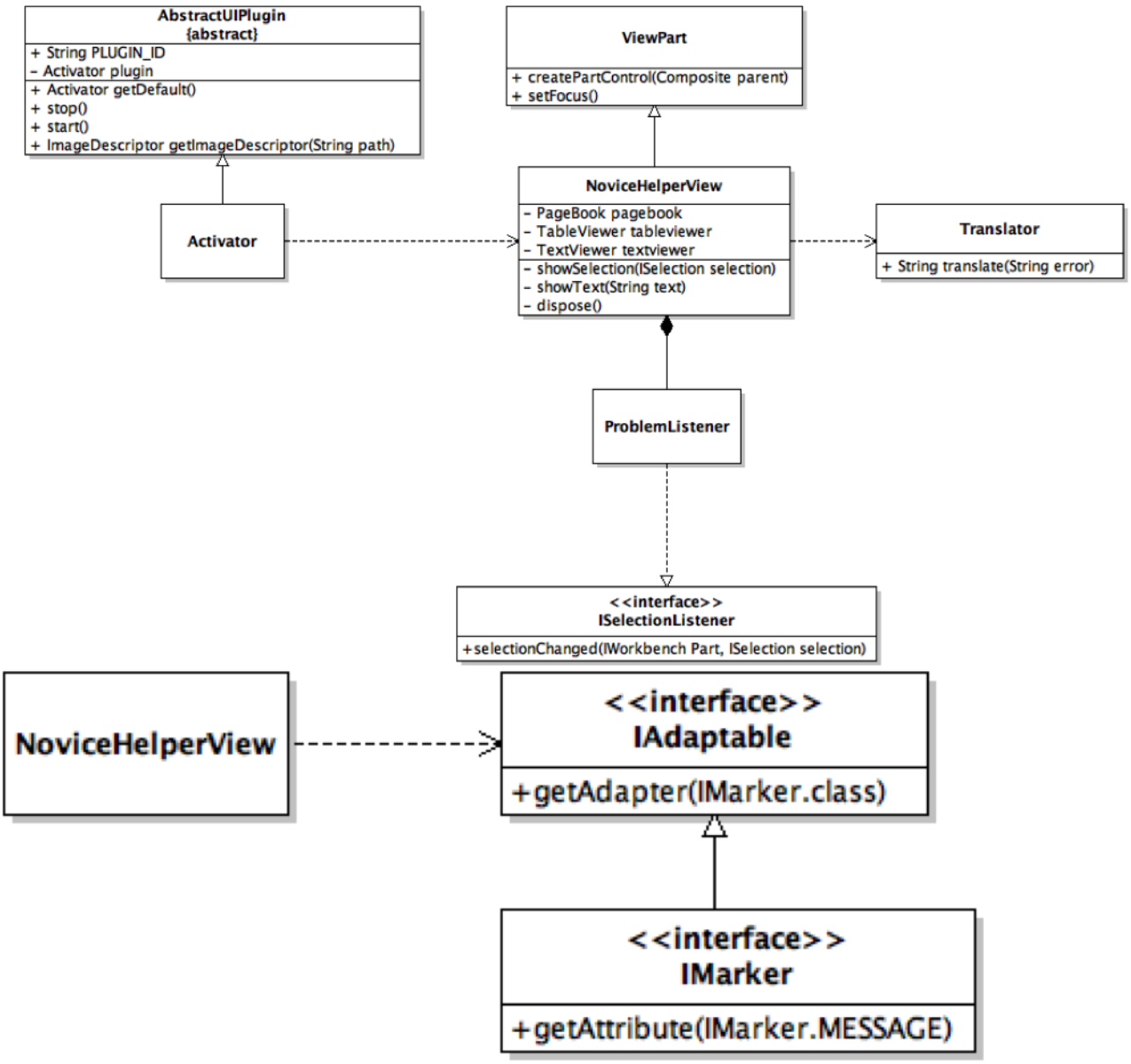
Shiffman, D. (2015). *Learning Processing.* Burlington, MA: Morgan Kaufmann.

APPENDIX C


Novice Helper View
Click on an error in the Problems View
A function must have a { and } to define the body of the function.
{ and } wrap around a block of code.


runtime-T1 - Java - Eclipse Platform
Quick Access
Package Explorer
t1
HelloWorld.java
package t1;
public class HelloWorld {
public static void main(String[] args)
for (int i = 0; i < 10; i++)
System.out.println(i);
}
}
Outline
t1
HelloWorld
main(String[]) : void
Problems
Javadoc
Declaration
1 error, 0 warnings, 0 others
Description
Resource
Path
Errors (1 item)
Syntax error on token ")", { expected after this...
HelloWorld.java
/t1/src/t1
Novice Helper View
Click on an error in the Problems View
A function must have a { and } to define the body of the function.
{ and } wrap around a block of code.
Syntax error on token ")", { expected after this token

APPENDIX D


AbstractUIPlugin
{abstract}
+ String PLUGIN_ID
- Activator plugin
+ Activator getDefault()
+ stop()
+ start()
+ ImageDescriptor getImageDescriptor(String path)
ViewPart
+ createPartControl(Composite parent)
+ setFocus()
Activator
NoviceHelperView
- PageBook pagebook
- TableViewer tableviewer
- TextViewer textviewer
- showSelection(ISelection selection)
- showText(String text)
- dispose()
Translator
+ String translate(String error)
ProblemListener
<<interface>>
ISelectionListener
+selectionChanged(IWorkbench Part, ISelection selection)
NoviceHelperView
<<interface>>
IAdaptable
+getAdapter(IMarker.class)
<<interface>>
IMarker
+getAttribute(IMarker.MESSAGE)