

Case Western Reserve University

Project Proposal

JEdu

[REDACTED]
[REDACTED]
[REDACTED]

[REDACTED]
[REDACTED]
[REDACTED]

September 13, 2019

Date	Version Number	Note
September 12	V 1.0	Initial Release Documentation
September 13	V 1.1	Team Reviewed and Signoff

Acronym and Terminology

- **DrJava** - DrJava is a lightweight development environment for writing Java programs. It is designed primarily for students, providing an intuitive interface which allows the students to interactively evaluate Java code.
- **EECS132** - Introduction to Programming in Java. An intro-level programming course available at Case Western Reserve University.
- **Interaction Pane** - Interaction Pane is an interactive UI of Drjava. It works similar to the command line which displays program outputs. It enables programmers to quickly try out code without having to write cumbersome main method structure.
- **Javadoc** - Javadoc is a documentation generator for the Java programming language. It generates API documentation in HTML format from Java source code commenting sections.
- **JDB** - The Java Debugger, JDB, is a simple command-line debugger for Java classes.
- **JDK** - The Java Development Kit.
- **JEdit** - JEdit is a text editor written in Java and runs on any operating system with Java support. It is open to community contributions for plugins.
- **JShell** - The Java Shell tool (JShell) is an interactive tool for learning the Java programming language and prototyping Java code.
- **JUnit** - JUnit is a unit testing framework for the Java programming language.

1. Scope and Vision

1.1 Background

As most IDEs are developed for professional programmers, major Java IDEs have specific requirements for project structures. Failure to meet these requirements will lead to unsuccessful compilation and code execution. Although such project structure allows the programmers to conveniently manage their projects, it is not instinctive for the novice programmers. The majority of students are taking EECS132 along with other intro-level courses, and therefore the project structure will cause unnecessary hindrances.

DrJava is a Java programming environment favored by one of the EECS132 professors, Prof. Harold Connamacher. Over many semesters, thousands of students picked up DrJava as their first Java programming environment. They benefited from its powerful interactive functionalities such as the interaction pane, the auto JUnit testing component, the integrated JDB debugger, as well as the JavaDoc generator. While JDK continues to release new updates, DrJava no longer supports the latest version of JDK and will soon become outdated due to the lack of maintenance.

As students who decided to major in computer science because of Prof. Connamacher's lectures, we are determined to help develop a substitute for DrJava, so that prospective students can benefit from the easy DrJava-like style of programming. After researching different Java text editors, we finally chose JEdit as the platform for our project. JEdit is an editor that allows convenient open source contributions through the integration of JEdit plugins.

1.2 Vision Statement

We aim to improve the development experience of JEdit by implementing plugins that support functionalities similar to those of the JUnit, JDB, and interaction pane in DrJava. To be more specific, we are planning to:

1. Build a JUnit plugin that provides the means of both writing customized JUnit tests and displaying test outputs readable by the novice programmers.
2. Implement a JDB plugin with an intuitive UI and make it accessible to the novice programmers.
3. Integrate an interaction pane plugin by providing a GUI for the JShell feature.

1.3 Success Criteria and Measures

No	Component	Success Criteria	Measure	Priority
1	JDB Plugin	The plugin is integrated with a seamless UI.	Implement a GUI for the JDB plugin that is consistent with the JEdit UI.	2
2	JDB Plugin	The plugin supports all of the necessary functionalities of a debugger.	The plugin features setting breakpoints, "step over", "step into", "step out", "resume", as well as querying variable contents.	1
3	JUnit Plugin	The plugin is developed as a functional testing plugin with a consistent GUI and does not require the presence of a project.	The plugin should allow the users to create test files and input their test cases formatted in the JUnit standard.	1
			The plugin features two displays. One will show each test with a fail/pass	1

			indicator, the other one will give the important error messages for the tests that fail.	
			The plugin should be able to test all of the other Java files that are saved under the same directory as the test file.	1
4	Interaction Pane	The interaction pane component should feature the same behavior as that of DrJava's interaction pane.	The interaction pane should take basic Java expressions and statements as inputs, then interpret and evaluate; the result should be displayed.	2

2. Detailed Feature Description

2.1 Feature Descriptions

2.1.1 Project structure not required

Our proposed development environment does not require building up a project structure for the code to compile and run: putting all the Java files in the same folder, instead of a sophisticated project structure, is sufficient to compile and run all the Java files in the folder.

2.1.2 Auto-detect class files under the same directory

The JShell programmers need to open all of the Java files of which the current compiled Java file depends on. For example, the *Employer* class contains a function *calculateAvgSalary* in *Utils* class, which is under the same folder. JShell will require users to manually open the *Utils* Java file in order to compile and run the *Employer* Java file. Our proposed development environment does not require opening all the files if under the same folder. In this way, opening all the files before compiling and running the program will no longer be a burden for novice programmers.

2.1.3 JUnit

Our proposed JUnit plugin will allow the user to create test files and write test code which conforms with JUnit's standard syntax without setting up a project structure. The plugin will also be able to test other files without opening them as long as they are under the same directory.

To provide feedback from the tests, our JUnit plugin will have two displays. One display shows a list of fail/pass indicator for each of the tests. The other one prints error messages for tests that fail. We will simplify the error reports by trimming out additional unimportant messages, translating technical terms into simple language, and indicating where an uncaught exception

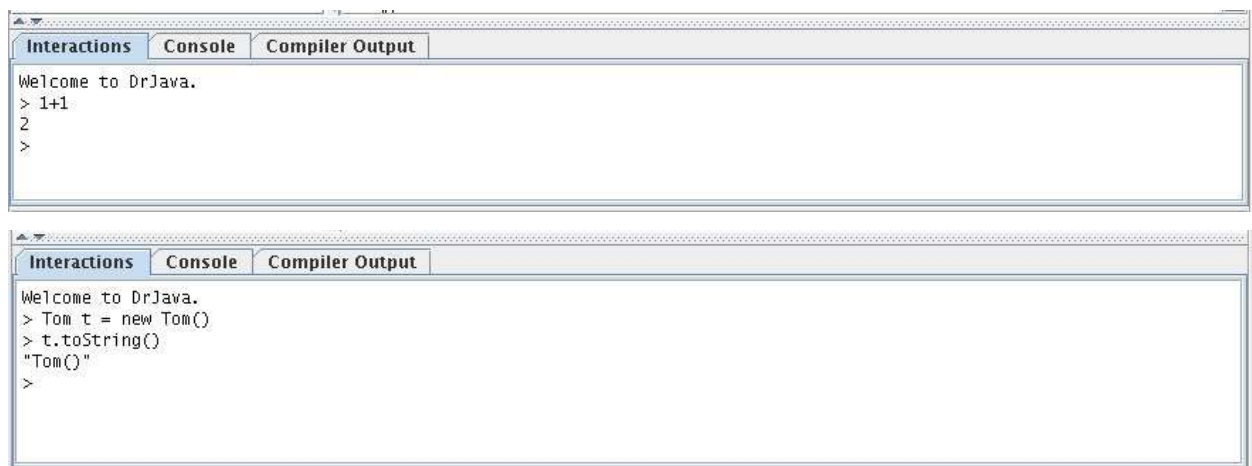
was thrown. To make debugging even easier, when the user clicks on an error message, the corresponding code that caused the error will be highlighted and displayed on the main screen.

2.1.4 JDB

Our proposed JDB plugin will provide standard debugging operations, including setting breakpoints, step into, step out, step over, resume, restart, etc. The plugin should also allow the users to investigate the values of parameters before and after the line is executed and trace the method calls.

2.1.5 Interaction Pane

Our proposed development environment provides an interaction pane, similar to the counterpart provided by DrJava. As demonstrated below, the interaction pane takes basic Java expressions, such as $(1+1)$, class instantiations, method calls, and package imports as inputs. The expressions will then be interpreted and evaluated, and the result will be displayed. The demo figures below show how the interaction pane in DrJava works.



(Reference : <http://www.drjava.org/>)

2.2 Feature Dependencies

FD-1: JEdit plugin dev toolkit.

FD-2: JUnit library.

FD-3: Reference to JEdit's existing plugin source codes for UI designs.

2.3 Operating Environment



OE-1: The plugins that we implement need to be run-able in all the operating systems that JEdit supports, namely Windows, OS X, and Linux. Because the plugins will be written in Java, we do not foresee any obstacle to achieving this goal.

2.4 Design and Implementation Constraints

CO-1: All code should be written in Java and compilable by JSE11 or later.

CO-2: All source code should be compatible with the JEdit environment.

3. Prioritized Deliverable List and Estimates

Pri	Name	Owner	Estimate	Description	Test/Validation
1	Wireframe Deliverable	YS	2 weeks	Wireframe deliverable of the UIUX design for all of the plugins. The wireframe deliverable should demonstrate the end-to-end workflow of the three plugins.	ALL
2	Functional Design Deliverable	ALL	4 weeks - done by the first report	Functional design deliverable of the plugins. The deliverable should contain all of the details necessary for the implementation of the plugins.	ALL
3	JDB Functional Demo	SG, YG, YS	7 weeks	Functional demo of JDB plugin. The plugin should demonstrate at least four basic functionalities of a standard debugger.	
3	JUnit Functional Demo	JY, ML, RY	7 weeks	Functional demo of JUnit plugin. The plugin should demonstrate the basic functionality of indicating testing status and mapping errors to the relevant line of code.	
4	Interaction Pane Functional Demo	ALL	5 weeks	Functional demo of interaction pane plugin. The plugin should demonstrate the full functionality of auto-loading files under the same directory, prompt the users to compile the files, and execute Java expressions or statements.	ALL
4	Final Deliverable	ALL	12 weeks	Final deliverable of all of the plugins. The deliverable should demonstrate seamless end-to-end user experience of each plugin, and pass the validation tests.	