Green

A Flexible UML Class Diagramming Tool for Eclipse

Gene Wang, Brian McSkimming, Zachary Marzec, Josh Gardner, Adrienne Decker, Carl Alphonce

Department of Computer Science & Engineering, University at Buffalo SUNY, New York, USA
{zgwang, bmm6, zamarzec, jeg34, adrienne, alphonce}@cse.buffalo.edu

Abstract

Green is a live round tripping UML class diagram editor plug-in for Eclipse, originally designed with the intention of focusing CS1/CS2 students on modeling and design. Green's ease of use and flexible features has allowed it to grow into a robust tool providing end users with an easy to use application satisfying their individual class diagramming needs. Green's live round tripping capability allows users to generate (Java) code from UML class diagrams and generate diagrams from (Java) code and have them both update each other as any changes are made.
This demo will demonstrate the main features of Green, including forward and reverse engineering, live round-tripping, incremental exploration, and the various aspects of relationship semantics. Since Green is an ongoing project, demonstrations of additional features not in this list may also be included.

Categories and Subject Descriptors

K.3.2 [**Computers and Education**]: Computer and Information Science Education — *Computer Science Education*;

D.2.3 [**Software Engineering**]: Coding Tools and Techniques — *Object-oriented programming*

General Terms

Design

Keywords

Design, Object-orientation, UML, Java, code generation, reverse-engineering

1. Live Round-tripping

One of the unique features of Green is its live round-tripping abilities. Green keeps its diagrams synchronized with the code with which they are associated at all times. When a change occurs in either the code or the diagram, an update is triggered in the other.

1.1 Creating diagrams from code

Diagrams are easily created via right-click context menus from existing code. Green is capable of generating diagrams from both source code as well as precompiled libraries.

1.2 Code generation

Green diagrams can be created independently from any source code. By drawing items in the diagram, Green will be able to generate corresponding code. A typical Green diagram is shown in Figure 1.

2. Relationships

2.1 Relationships as plug-ins

Relationships in Green are defined separately from the central plug-in. Relationships themselves are plug-ins to the Green plug-in. This setup allows the addition and removal of relationships to Green, depending on the user's preferences. A user can also develop his/her own relationship semantics for use with Green.

2.2 Multiple semantics

Relationships in Green are flexible in that a certain relationship is allowed to have multiple semantics defined for it. The separation will be incorporated seamlessly as subcategories of the main relationship's type.

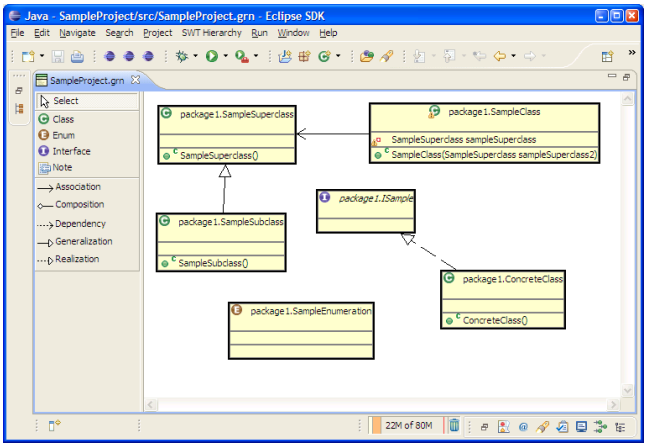

Figure 1. A typical Green diagram.

OOPSLA'07 October 21–25, 2007, Montréal, Québec, Canada.
ACM 978-1-59593-865-7/07/0010.

3. Incremental exploration

Overview of a particular section of a larger project may be necessary, but finding all the components of the section isn't always easy. Green allows for exploration along relationships. By starting at a base type, a user can expand along relationships to find types containing relationships with the base type.

4. Java 5 Support

4.1 Support for Enumerations

Green provides some basic support for the enum type introduced in Java 5. Enumerations can be added to diagrams and are treated, as they are in Java, as a separate type from classes and interfaces. Relationships involving enums, however, may not be displayed and /or recognized correctly.

4.2 Support for Generics

Similar to the case with enums, Green does not yet provide full support for generics. Green recognizes generics and properly handles generic types in diagrams. Relationship manipulation with such types can result in some unwarranted behavior. The majority of these issues come from the way relationship semantics are currently defined in Green.

5. View Customization

5.1 Show / hide relationships

Individual types of relationships can be shown or hidden via a visibility menu. Types which are sources or targets of a hidden relationship will have its border color coded.

5.2 Choosing what to include

A type or multiple types can be added to a Green diagram through the Package Explorer view in Eclipse. Depending on where the context menu is invoked, different sets of types can be added to a diagram at one time. The action can be invoked from the project level, the package level, or the compilation unit level.

5.3 Hiding types

Types can be hidden (or removed) from a diagram using Backspace while it is highlighted. This is in contrast to using Delete, which will delete the particular type from the environment altogether.

5.4 Bendpoints

Relationships arcs in Green are by default a straight line linking the source and target types. The user can add waypoints in the route of the relationship line (known as bendpoints) to customize the diagram and control where the relationship arcs lie.

5.5 Manhattan routing

The standard routing scheme in Green is to use straight lines to draw relationship arcs with optional bendpoints added by the user. A different routing scheme the user can opt to use is Manhattan routing. With this scheme, each relationship arc will be drawn by a combination of vertical and horizontal lines only and have either zero or two bendpoints.

5.6 Notes

In a design diagram, it is sometimes beneficial to make remarks on the diagram. In Green, this is provided for through the note system. Notes are not related to any Java resource and exist only within the diagram.

5.7 Preferences system

Green maintains its own set of preferences within the Eclipse preferences system. All aspects of a Green diagram's appearance can be customized, from the coloring of various components, to how names of types are to be displayed.

6. Exporting as Image

Green diagrams are typically saved in XML format. However, they can be exported as either a JPEG or a GIF type image. This will allow a user to share a generated diagram with others without requiring the viewers to have Green. Images will also make a diagram state permanent in a snapshot manner.

7. Known Issues

Various notable shortcomings of Green have been identified and are being improved or corrected. The demonstration will discuss known issues and their potential future resolutions. However, the issues here may or may not still exist at the time of the demonstration.

- Green does not handle static inner classes with complete accuracy.
- Types added to a Green diagram are currently placed in a grid. The current plan is to incorporate an auto-layout function which will place elements in a more logical arrangement.
- While a Green diagram can be created which spans multiple screen widths and /or heights, there is currently no way of viewing the entirety of such a diagram at one time.
- Refactoring an element represented in a Green diagram while the diagram is closed will cause the diagram to lose the element.

References

[1] Carl Alphonce and Blake Martin. (2005). *Green: A customizable UML class diagram plug-in for Eclipse.* A poster presentation at the 20th Annual Conference on Object-Oriented Programming, Systems, Languages and Applications (OOPSLA), San Diego. http://www.oopsla.org/2005/ShowEvent.do?id=532