Workshop
jGRASP: An Integrated Development Environment with Visualizations for Teaching Java in CS1, CS2, and Beyond

James H. Cross II and Dean Hendrix
Computer Science and Software Engineering
Auburn University, Auburn, AL 36849
cross@eng.auburn.edu, hendrix@eng.auburn.edu

Abstract – The purpose of the workshop is to introduce faculty to the advanced pedagogical features provided by jGRASP for teaching and learning Java, especially for first year students. This includes the traditional CS1 (introduction to programming) and CS2 (introduction to data structures and algorithms). jGRASP is a freely available lightweight integrated development environment (http://jgrasp.org) that provides automatically generated visualizations for improving the comprehensibility of software. These visualizations, which are particularly well suited for CS1 and CS2 students learning Java, include Control Structure Diagrams, UML Class Diagrams, and dynamic Object Views (including arrays, ArrayList, LinkedList, HashMap, and TreeMap). The object workbench and integrated debugger facilitate a unique way for students to view objects created by their programs. The workshop will include hands-on activities and example programs to demonstrate how instructors can improve the learning and programming experience of their students by using jGRASP and the pedagogically sound visualizations it provides.

Index Terms – control structures, data structures, integrated development environment, software visualizations.

EXAMPLE

Figure 1 shows a UML class diagram generated by jGRASP for a program called LinkedListExample. This program creates an instance of class LinkedList which consists of a set of LinkedNodes. Figure 2 shows the `insert()` method for LinkedList with the control structure diagram generated by jGRASP. Figure 3 shows a dynamic object view for the instance of LinkedList as the user steps through the program in debug mode. After the links for the local node have been set, the node moves up into the list.

```java
public void insert(Object value, int index) {
    if (index == 0) {
        add(value);
        return;
    }
    ListNode node = new ListNode(value);
    ListNode prev = head;
    for (int i = 1; i < index; i++)
        prev = prev.next;
    node.next = prev.next;
    prev.next = node;
}
```

Figure 2. Control structure diagram

Figure 1. UML class diagram

Figure 3. Inserting an element into linked list