

# Intuitive Source Code Visualization Tools for Improving Student Comprehension: BRICS

Chris Pearson

[pearson@csc.uvic.ca](mailto:pearson@csc.uvic.ca)

[www.csc.uvic.ca/~pearson](http://www.csc.uvic.ca/~pearson)

# My Background

- ▶ Ten years in the industry
  - Java 1.0 applets in 1995
  - C++ web engine research for education and ecommerce during the dot-com boom.
  - Presentations in India, UK, etc.
  - Teaching Linux systems
- ▶ University
  - Not so good the first time...
  - A little bit better this time!
    - Returning to pursue my love of research work.



# Current Work

- ▶ Interests include:
  - User interfaces and visual layout
  - Cognitive psychology
- ▶ This work is speculative, but based on good anecdotal evidence
  - Research exists regarding the basic idea.
- ▶ This work is just beginning, and good feedback is needed.
- ▶ Please inform me of any related research
- ▶ This research will (hopefully!) lead to a doctoral thesis

# Our World Has Changed

- ▶ Rapid changes in the LCD monitor market gives more display area.
  - High resolution, wide screen, multi-monitor workstations
- ▶ Most new students have never used a command line interface.
  - “So... What’s that little black box?”
- ▶ However, they have high expectations as they feel they are advanced computer users.
  - Masters of Web 2.0
  - Greater computer skills than their parents
  - Possibly this is worse than the computer students of old, as they think they know a great deal?

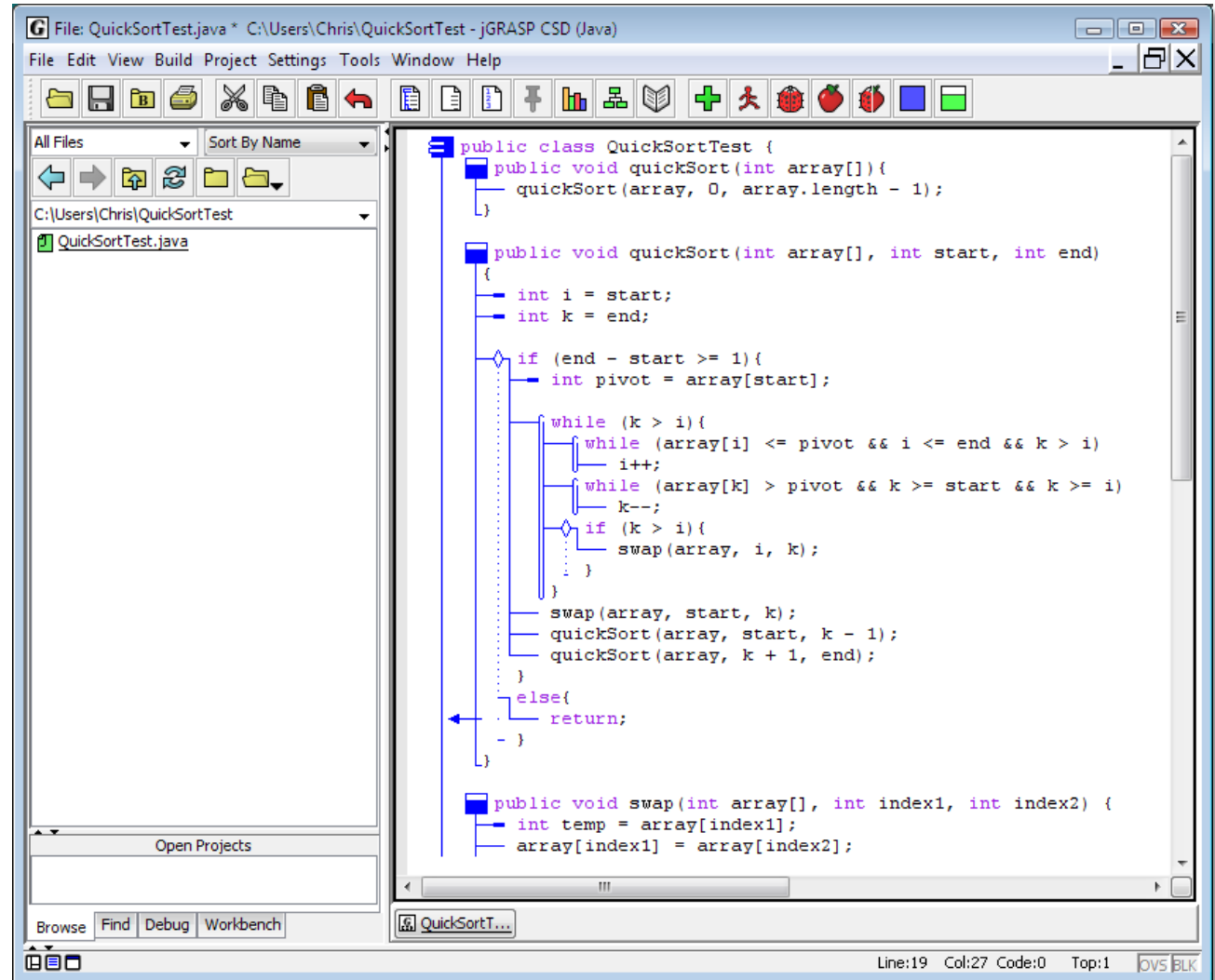
# Difficulties in CS1

- ▶ Making the transition from purely visual tools to textual source code
  - LEGO Mindstorms/NXT, Scratch, etc
- ▶ Losing students who can't make the leap from diagrams to source code
- ▶ Students do not have the tools to assist with the structure within objects
  
- ▶ What if an object oriented approach was taken to the functional structure within methods?



# An Existing Tool: jGRASP

- ▶ Lines connect functional blocks
- ▶ Well researched
- ▶ Is not intuitive at first glance



# A Solution – BRICS

- ▶ Mock-ups are cheap, but good planning is valuable.
- ▶ Any functionality added to an editor must be carefully considered
  - What is the addition to the learning curve?
  - Does the addition balance added visual/mental work load with usefulness?

# BRICS

- ▶ Starting with average source code

```
public class QuickSortTest {
    public void quickSort(int array[]){
        quickSort(array, 0, array.length - 1);
    }

    public void quickSort(int array[], int start, int end)
    {
        int i = start;
        int k = end;

        if (end - start >= 1){
            int pivot = array[start];

            while (k > i){
                while (array[i] <= pivot && i <= end && k > i)
                    i++;
                while (array[k] > pivot && k >= start && k >= i)
                    k--;
                if (k > i){
                    swap(array, i, k);
                }
            }
            swap(array, start, k);
            quickSort(array, start, k - 1);
            quickSort(array, k + 1, end);
        }
        else{
            return;
        }
    }

    public void swap(int array[], int index1, int index2) {
        int temp = array[index1];
        array[index1] = array[index2];
    }
}
```



# BRICS

- ▶ Block the code on the screen like it is done on paper

```
public class QuickSortTest {
    public void quickSort(int array[]){
        quickSort(array, 0, array.length - 1);
    }

    public void quickSort(int array[], int start, int end)
    {
        int i = start;
        int k = end;

        if (end - start >= 1){
            int pivot = array[start];

            while (k > i){
                while (array[i] <= pivot && i <= end && k > i)
                    i++;
                while (array[k] > pivot && k >= start && k >= i)
                    k--;
                if (k > i){
                    swap(array, i, k);
                }
            }
            swap(array, start, k);
            quickSort(array, start, k - 1);
            quickSort(array, k + 1, end);
        }
        else{
            return;
        }
    }

    public void swap(int array[], int index1, int index2) {
        int temp = array[index1];
        array[index1] = array[index2];
    }
}
```



# BRICS

- ▶ Block the code on the screen like it is done on paper

```
public class QuickSortTest {
    public void quickSort(int array[]){
        quickSort(array, 0, array.length - 1);
    }

    public void quickSort(int array[], int start, int end)
    {
        int i = start;
        int k = end;

        if (end - start >= 1){
            int pivot = array[start];

            while (k > i){
                while (array[i] <= pivot && i <= end && k > i)
                    i++;
                while (array[k] > pivot && k >= start && k >= i)
                    k--;
                if (k > i){
                    swap(array, i, k);
                }
            }
            swap(array, start, k);
            quickSort(array, start, k - 1);
            quickSort(array, k + 1, end);
        }
        else{
            return;
        }
    }

    public void swap(int array[], int index1, int index2) {
        int temp = array[index1];
        array[index1] = array[index2];
    }
}
```



# BRICS

- ▶ Block the code on the screen like it is done on paper

```
public class QuickSortTest {
    public void quickSort(int array[]){
        quickSort(array, 0, array.length - 1);
    }

    public void quickSort(int array[], int start, int end)
    {
        int i = start;
        int k = end;

        if (end - start >= 1){
            int pivot = array[start];

            while (k > i){
                while (array[i] <= pivot && i <= end && k > i)
                    i++;
                while (array[k] > pivot && k >= start && k >= i)
                    k--;
                if (k > i){
                    swap(array, i, k);
                }
            }
            swap(array, start, k);
            quickSort(array, start, k - 1);
            quickSort(array, k + 1, end);
        }
        else{
            return;
        }
    }

    public void swap(int array[], int index1, int index2) {
        int temp = array[index1];
        array[index1] = array[index2];
    }
}
```



# BRICS

- ▶ Block the code on the screen like it is done on paper

```
public class QuickSortTest {
    public void quickSort(int array[]){
        quickSort(array, 0, array.length - 1);
    }

    public void quickSort(int array[], int start, int end)
    {
        int i = start;
        int k = end;

        if (end - start >= 1){
            int pivot = array[start];

            while (k > i){
                while (array[i] <= pivot && i <= end && k > i)
                    i++;
                while (array[k] > pivot && k >= start && k >= i)
                    k--;
                if (k > i){
                    swap(array, i, k);
                }
            }
            swap(array, start, k);
            quickSort(array, start, k - 1);
            quickSort(array, k + 1, end);
        }
        else{
            return;
        }
    }

    public void swap(int array[], int index1, int index2) {
        int temp = array[index1];
        array[index1] = array[index2];
    }
}
```



# BRICS

- ▶ Block the code on the screen like it is done on paper
  - This is been mentioned by several first-time viewers of BRICS.

```
public class QuickSortTest {
    public void quickSort(int array[]){
        quickSort(array, 0, array.length - 1);
    }

    public void quickSort(int array[], int start, int end)
    {
        int i = start;
        int k = end;

        if (end - start >= 1){
            int pivot = array[start];

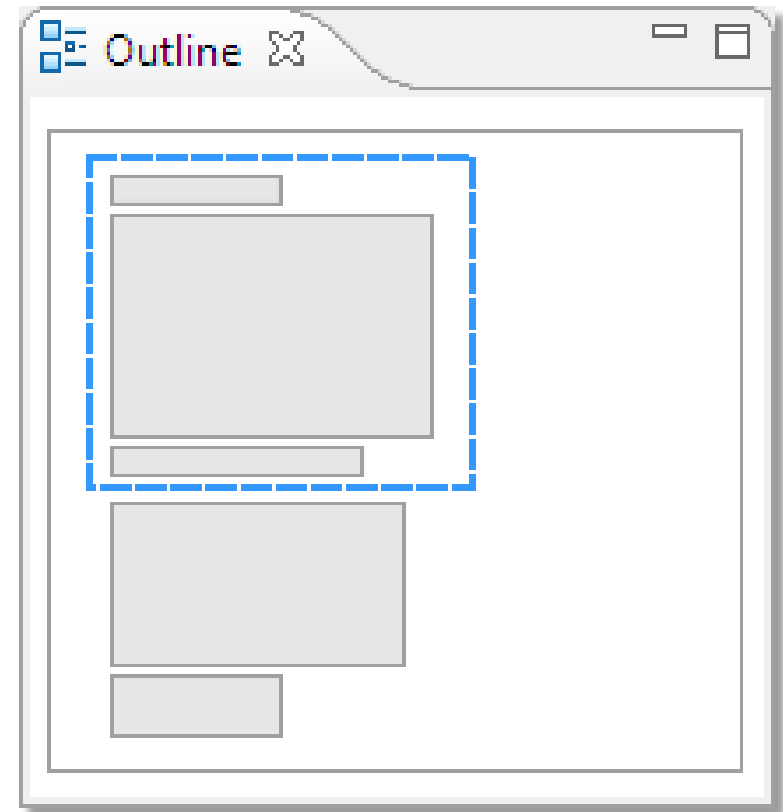
            while (k > i){
                while (array[i] <= pivot && i <= end && k > i)
                    i++;
                while (array[k] > pivot && k >= start && k >= i)
                    k--;
                if (k > i){
                    swap(array, i, k);
                }
            }
            swap(array, start, k);
            quickSort(array, start, k - 1);
            quickSort(array, k + 1, end);
        }
        else{
            return;
        }
    }

    public void swap(int array[], int index1, int index2) {
        int temp = array[index1];
        array[index1] = array[index2];
    }
}
```



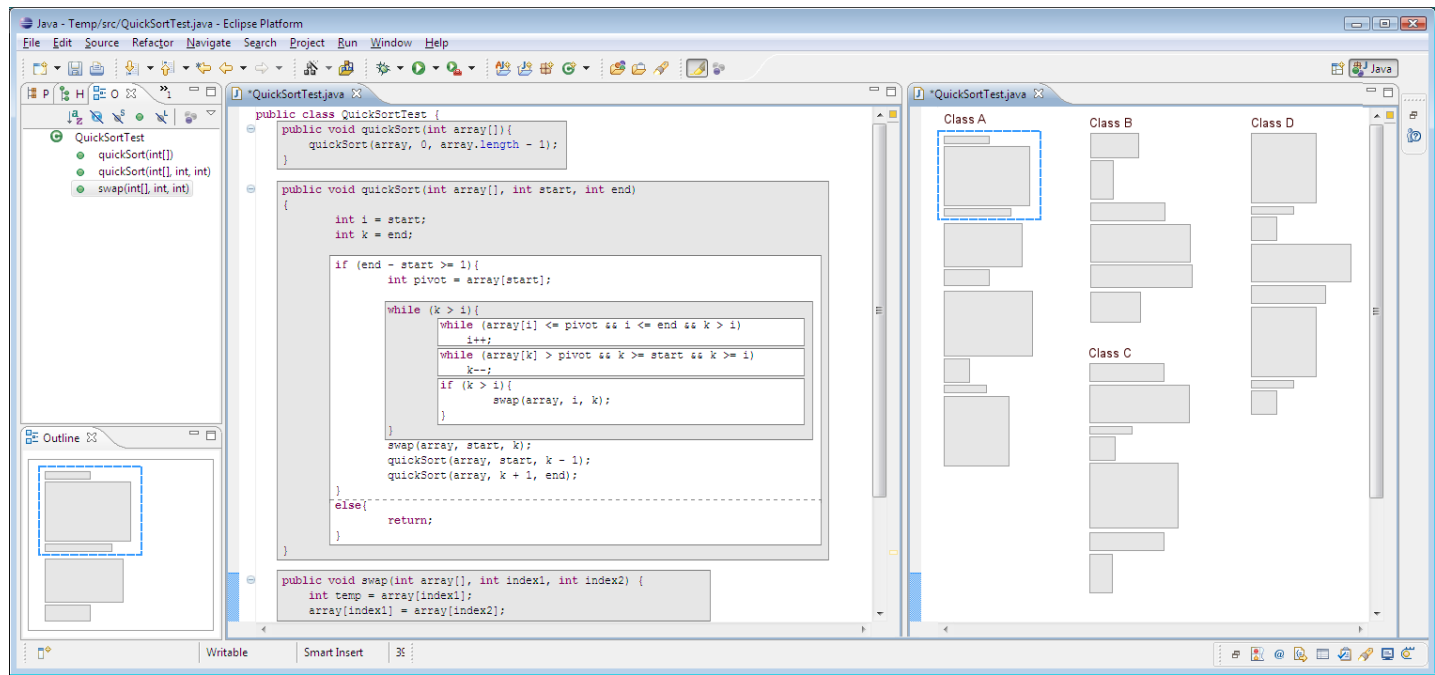
# BRICS and Overviews

- ▶ Giving shape to code gives the ability to overview
  - Similar to what is seen in graphic software



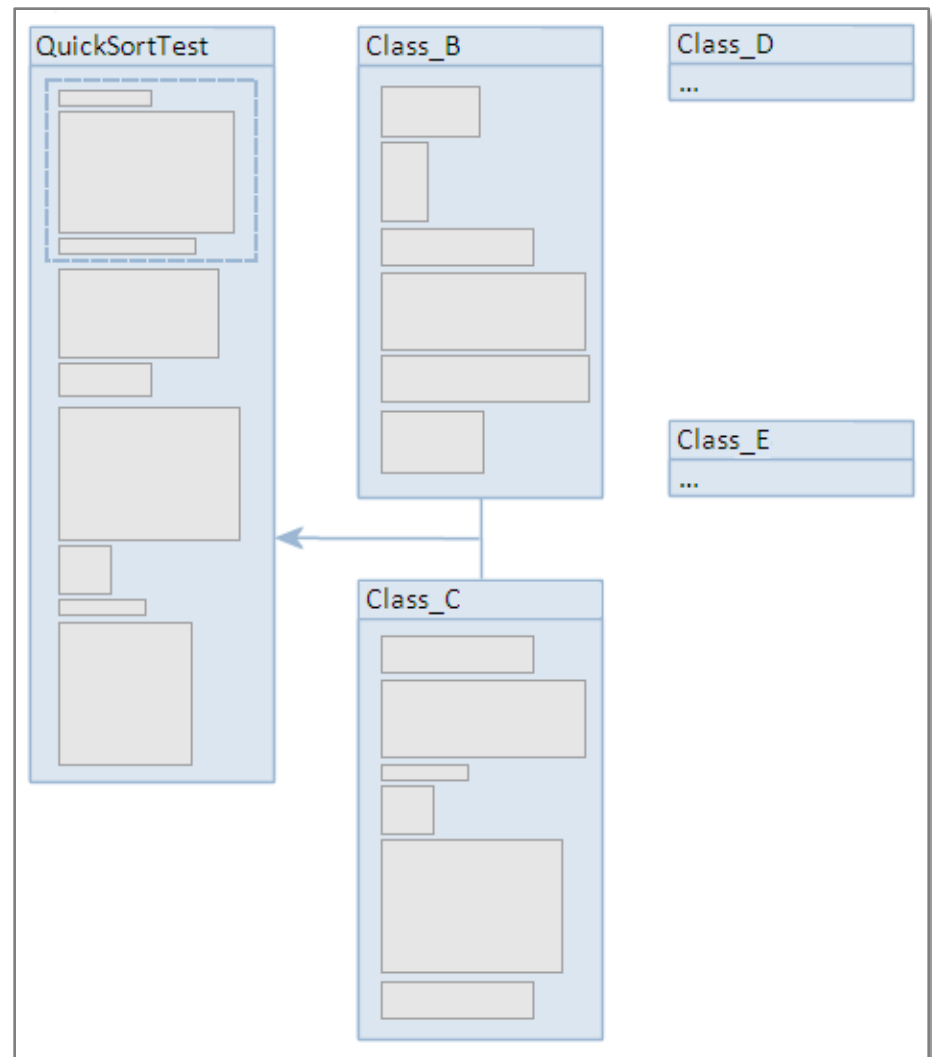
# BRICS and Overview Windows

- ▶ Overviews of several files



# Extending BRICS to Class Diagrams

- ▶ Zoom out and show method shape blocks inside class diagram blocks



# Peripheral Vision

- ▶ The part of vision outside the very center of gaze.
- ▶ Truly focused vision is an area the size of a quarter at arms length
- ▶ Peripheral vision is good for motion and simple shapes
- ▶ Why not make greater use of peripheral vision?



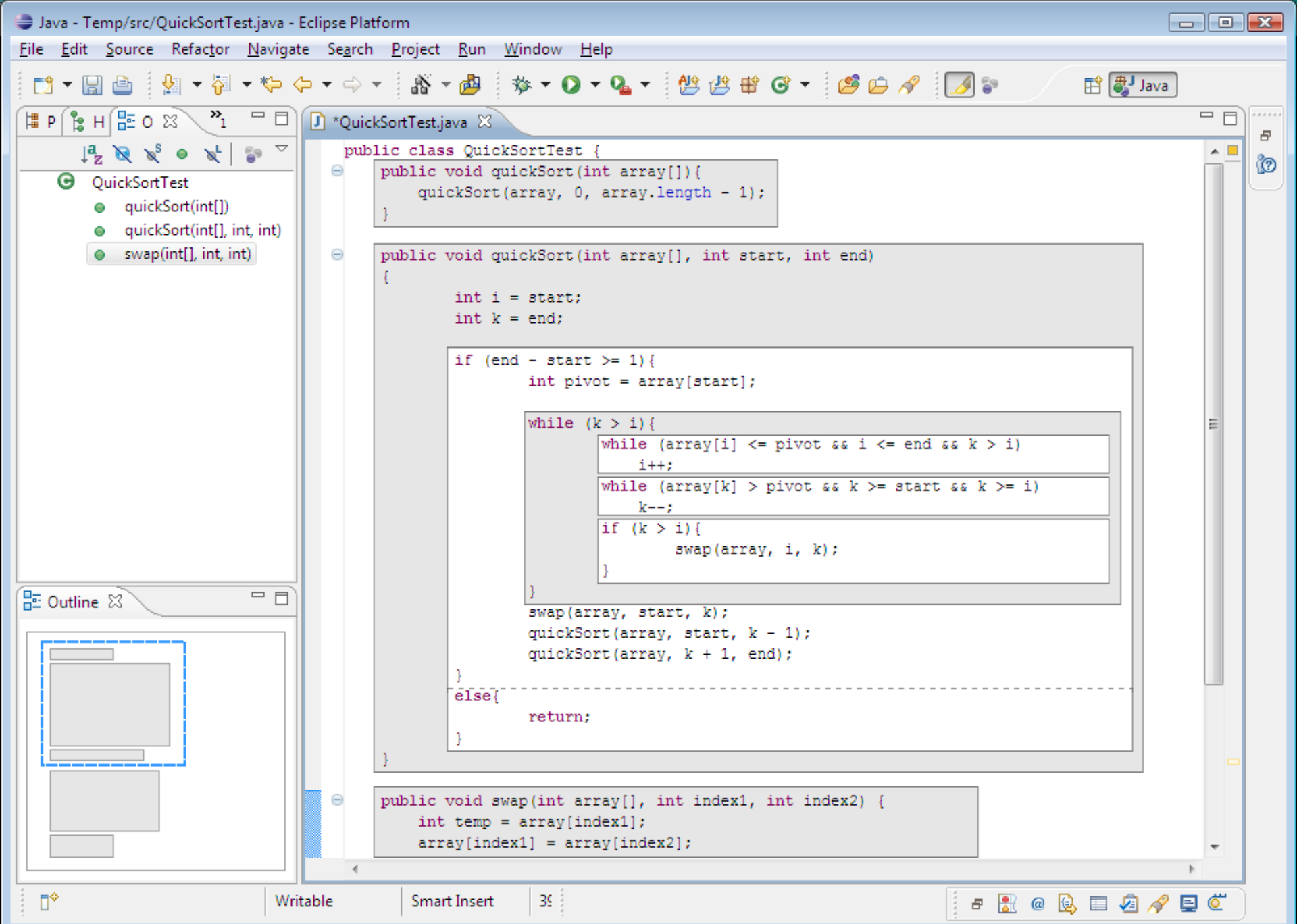
# BRICS and Peripheral Vision

- ▶ Blocks that make use of peripheral vision
  - Placing borders on the blocks
  - Visual nesting
  - What about visual noise?

```
public void quickSort(int array[], int start, int end)
{
    int i = start;
    int k = end;

    if (end - start >= 1){
        int pivot = array[start];

        while (k > i){
            while (array[i] <= pivot && i <= end && k > i)
                i++;
            while (array[k] > pivot && k >= start && k >= i)
                k--;
            if (k > i){
                swap(array, i, k);
            }
        }
        swap(array, start, k);
        quickSort(array, start, k - 1);
        quickSort(array, k + 1, end);
    }
    else{
        return;
    }
}
```



# Features for BRICS

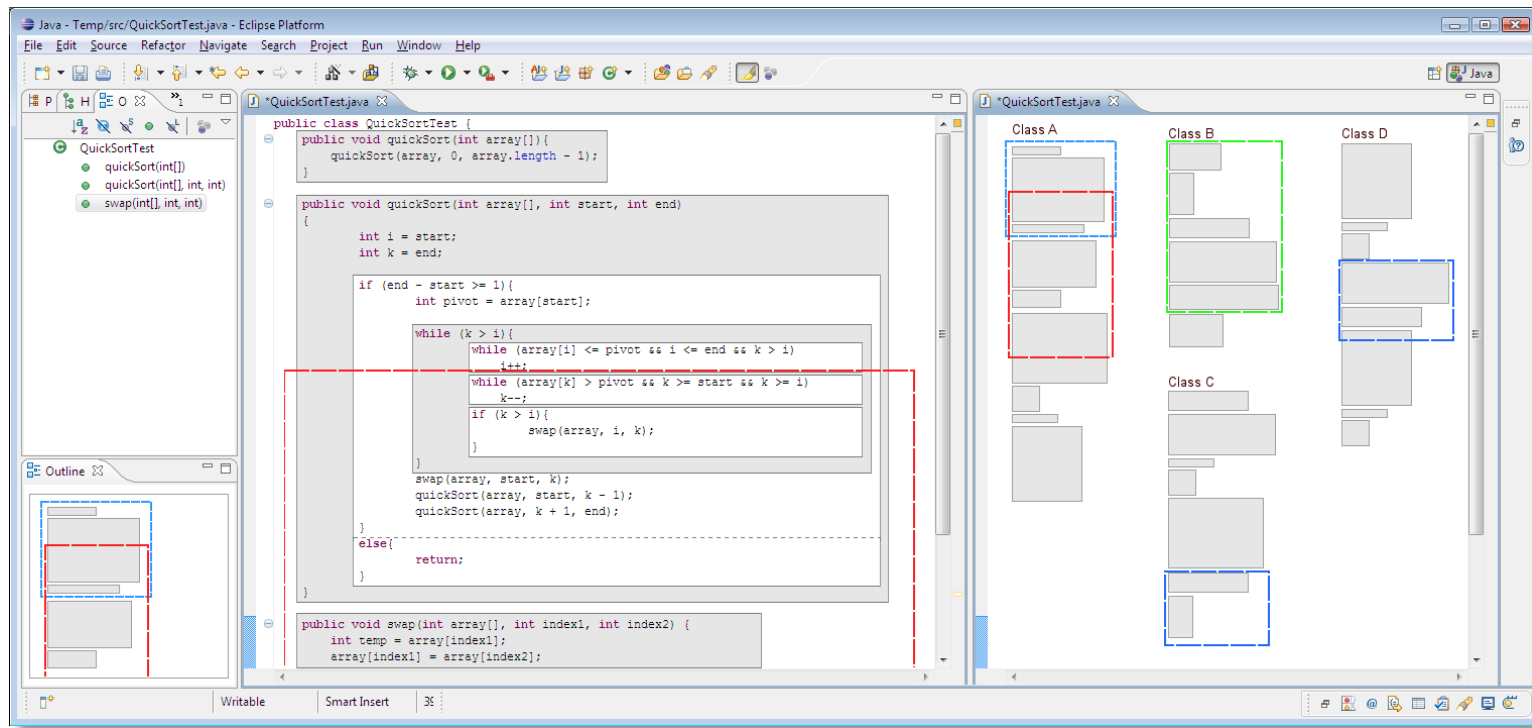
- ▶ Code collapsing
  - Roll up to the top, leaving the top line visible.
- ▶ Moving bricks of code by drag and drop
  - Using Eclipse as an example, this kind of block movement is supported by built-in refactoring support.
- ▶ Flagging of structure errors

# Evaluating BRICS

- ▶ Only anecdotal evidence (so far)
  - Every CS1 student that has seen BRICS wishes it already existed.
  - Every grad student who has seen it has said they could make use of it
  - But this isn't enough feedback...
- ▶ Formal evaluations before code is written
  - Use survey and eye tracking information with mock-ups to demonstrate that BRICS is worth pursuing

# Extending BRICS to Collaboration

- ▶ BRICS leads to the ability to draw overviews, which in turn leads to the ability to show what code segments others are looking at.



# Extending BRICS to Collaboration

- ▶ BRICS leads to the ability to draw overviews, which in turn leads to the ability to show what code segments others are looking at.
- ▶ Voice communication to talk about the code in the same way they would in the same room.
- ▶ How well can we simulate the computer lab experience over a distance?

# Questions to be Answered

## BRICS:

- ▶ Is it useful?
- ▶ Would combining BRICS class outlines with standard class diagrams be effective?
- ▶ How simple is too simple?

## Collaboration:

- ▶ Is real-time multi-user editing of source code feasible?

# Conclusion

- ▶ These ideas will be worked on
  - Tests with the mock-ups will begin in January.
  - As part of their honours technical project, two 4<sup>th</sup> years CSC students are working on an Eclipse plugin of the BRICS editor
  - My intention is to develop a thesis from this material.
  
- ▶ Any suggestions or feedback? I will be at the conference all week!

# Intuitive Source Code Visualization Tools for Improving Student Comprehension: BRICS

Chris Pearson

[pearson@csc.uvic.ca](mailto:pearson@csc.uvic.ca)

[www.csc.uvic.ca/~pearson](http://www.csc.uvic.ca/~pearson)