Why Programmers don't use Refactoring Tools
(and what we can do about it)

Andrew P. Black
joint work with Emerson Murphy-Hill

Portland State University

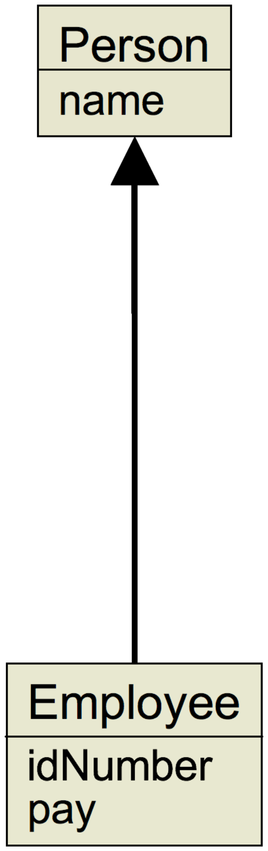
Outline

- What is Refactoring/Refactoring Tools
  - Refactoring tools are "a good thing"
- Are the tools being used?
  - No
- Why Refactoring Tools are Underused
  - It's the tools fault
- What's Wrong with Typical Tools
- How to fix the problem

What is Refactoring?

Changing the structure of code without changing the way that it behaves.

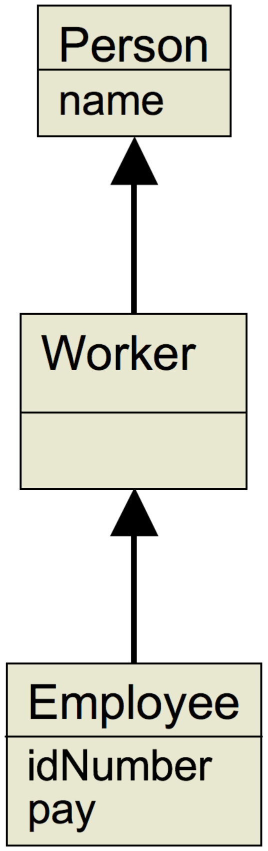
We do it because no one has perfect foresight.



What is Refactoring?

Changing the structure of code without changing the way that it behaves.

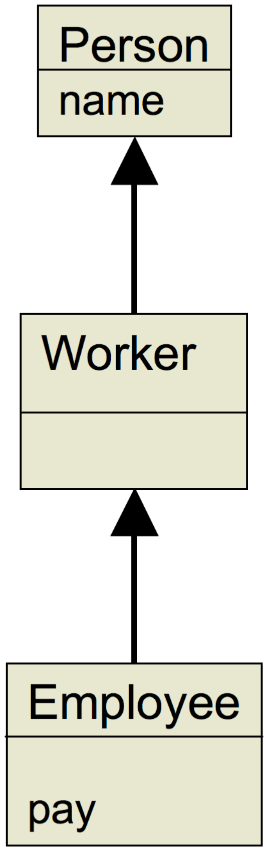
We do it because no one has perfect foresight.



What is Refactoring?

Changing the structure of code without changing the way that it behaves.

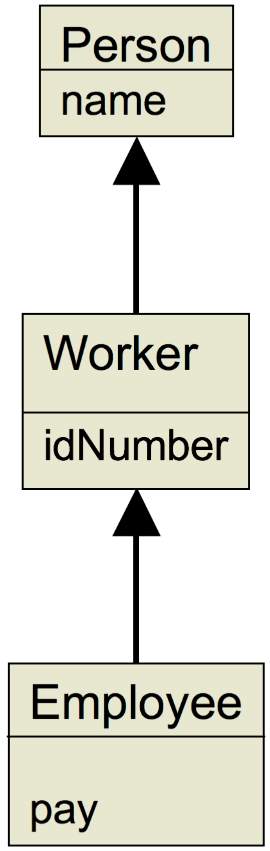
We do it because no one has perfect foresight.



What is Refactoring?

Changing the structure of code without changing the way that it behaves.

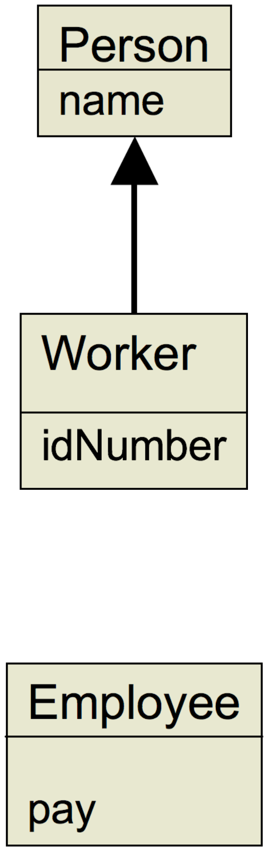
We do it because no one has perfect foresight.



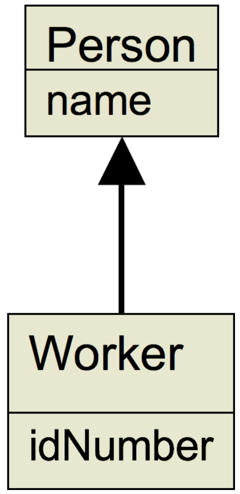
What is Refactoring?

Changing the structure of code without changing the way that it behaves.

We do it because no one has perfect foresight.



What is Refactoring?



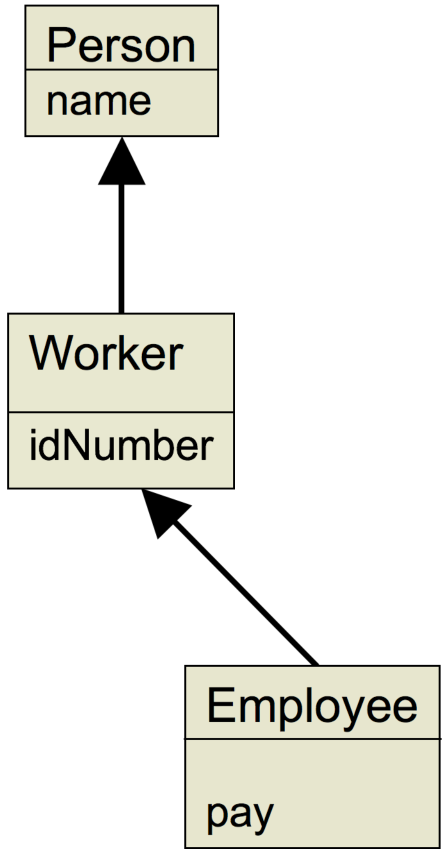
Changing the structure of code without changing the way that it behaves.

We do it because no one has perfect foresight.

What is Refactoring?

Changing the structure of code without changing the way that it behaves.

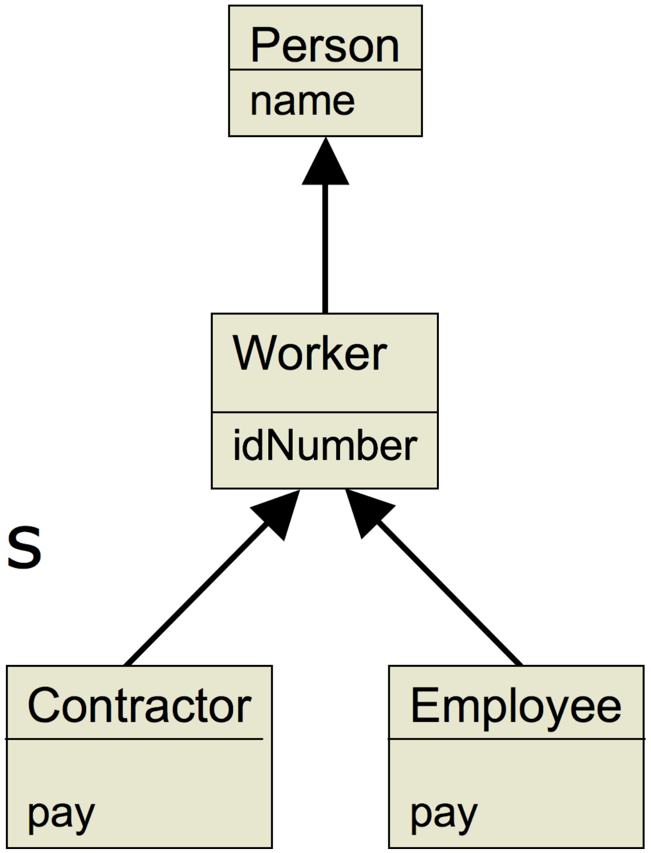
We do it because no one has perfect foresight.



What is Refactoring?

Changing the structure of code without changing the way that it behaves.

We do it because no one has perfect foresight.



Common Refactorings

- Rename
- Insert Superclass
- Push up/down method
- Push up/down field
- Extract Method/Inline Method
- Abstract/Reify field

Common Refactorings

- Rename
- Insert Superclass
- Push up/down method
- Push up/down field
- Extract Method/Inline Method
- Abstract/Reify field

```
speed := distance / time
```

Common Refactorings

- Rename
- Insert Superclass
- Push up/down method
- Push up/down field
- Extract Method/Inline Method
- Abstract/Reify field

```
speed := distance / time
```

→

Portland State University

Common Refactorings

- Rename
- Insert Superclass
- Push up/down method
- Push up/down field
- Extract Method/Inline Method
- Abstract/Reify field

```
speed := distance / time
```

$\rightarrow$

```
speed := this.distance() / time
```

Common Refactorings

- Rename
- Insert Superclass
- Push up/down method
- Push up/down field
- Extract Method/Inline Method
- Abstract/Reify field

```
speed := distance / time
```

```
speed := this.distance() / time
```

Common Refactorings

- Rename
- Insert Superclass
- Push up/down method
- Push up/down field
- Extract Method/Inline Method
- Abstract/Reify field

```
speed := distance / time   ←   speed := this.distance() / time
```

What’s the big deal?

Refactoring is a fancy name for what we used to call “keeping code clean”

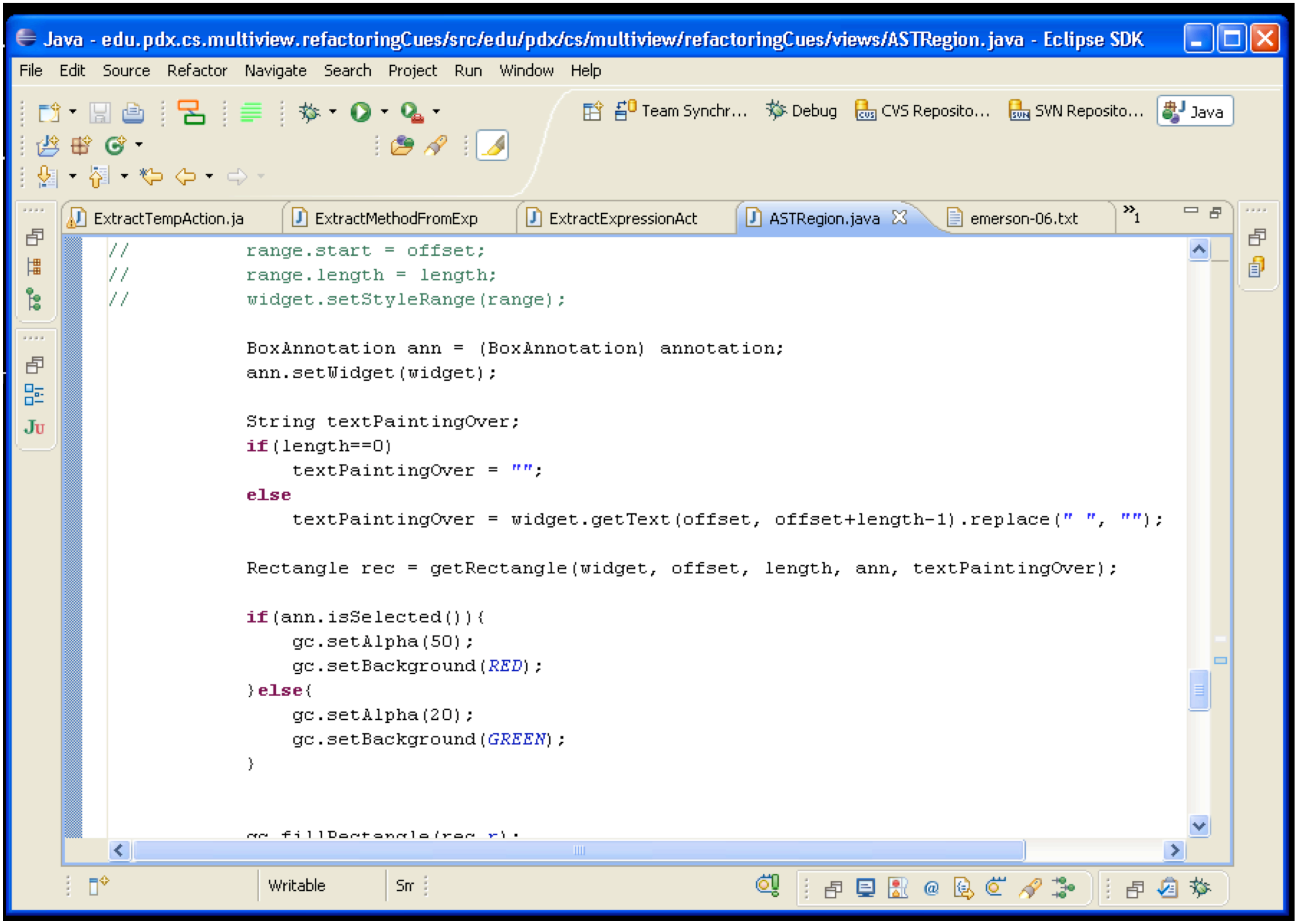
- It’s essential for a healthy code base
- Xing and Stroulia (ICSM ‘06) report that up to 70% of code changes can be due to refactoring
- Empirical data show that refactoring does improve code:
  - Kataoka: decreased coupling
  - Benn et al: complexity, size, cohesion, and coupling all improved
  - Kolb et al: maintainability and usability are increased

  … and many other studies

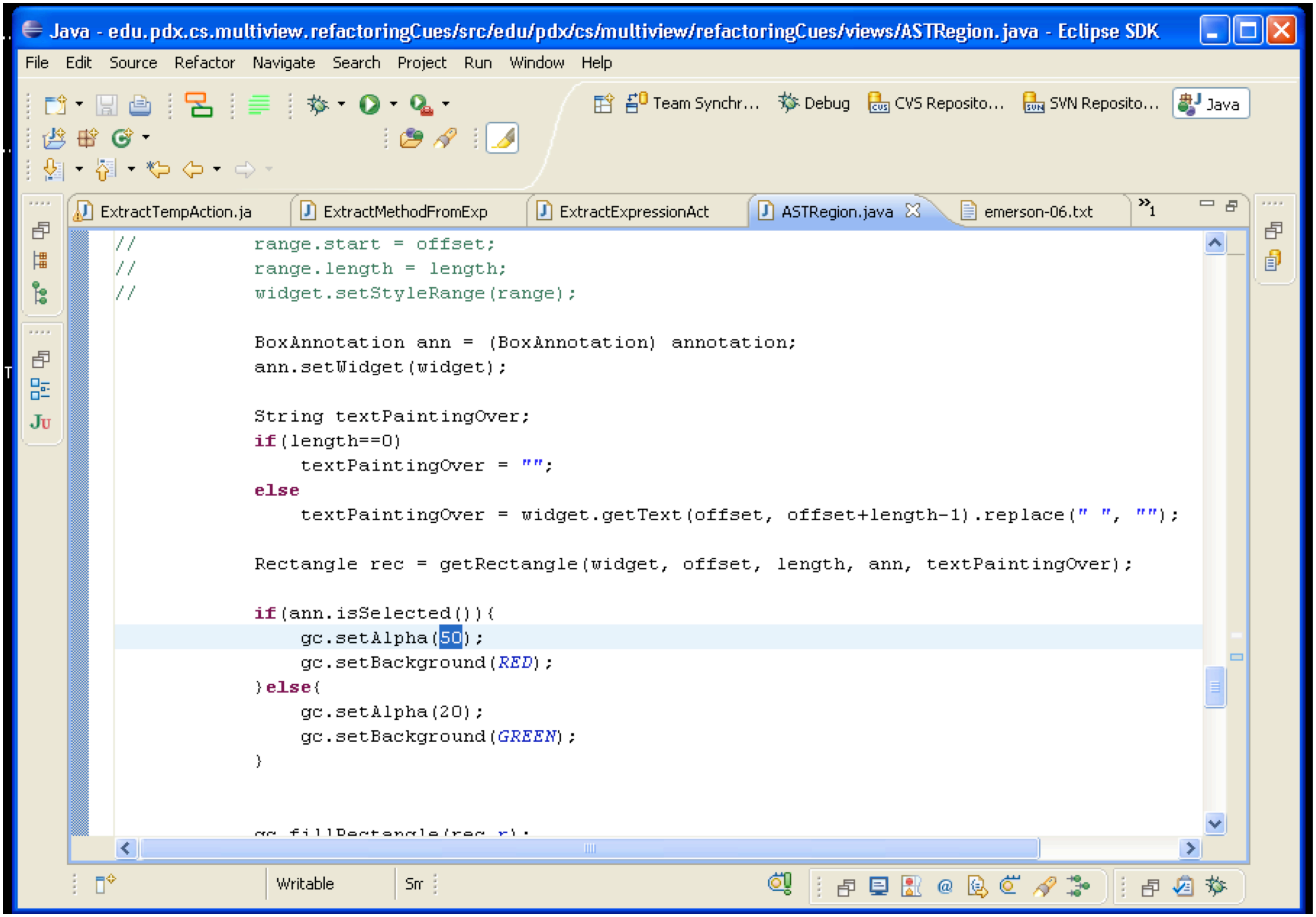
What's a Refactoring Tool?

- A tool that automates how you used to refactor by hand.
  - Knows about the semantics of the language
  - Refactors quickly
  - Refactors without introducing new errors
- Refactoring tools are provided by:
  - Eclipse/Java
  - IDEA (from IntelliJ)
  - Visual Studio
  - Smalltalk
- Not refactoring tools:
  - Find/replace
  - SED / AWK

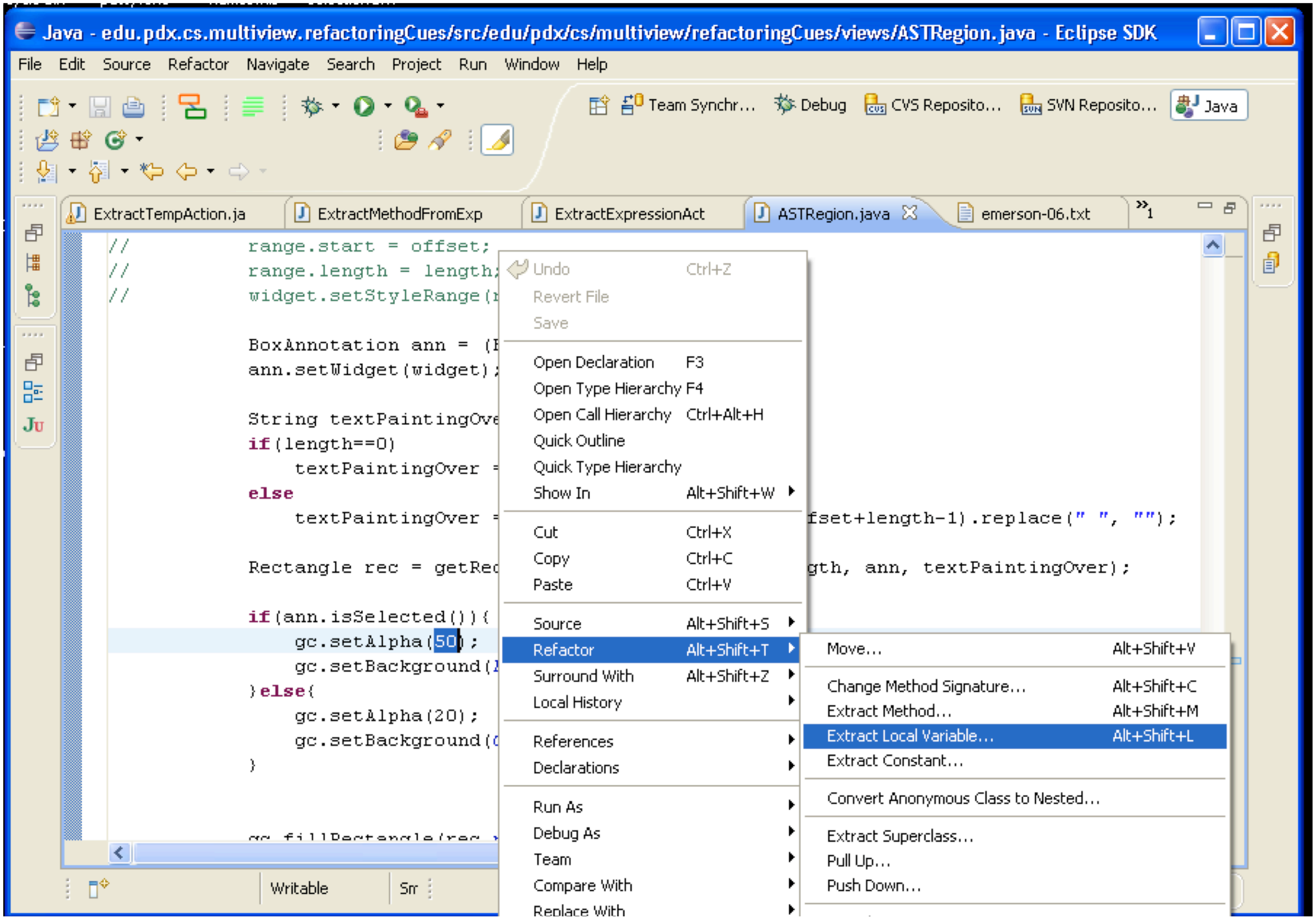
A typical tool interaction


Java - edu.pdx.cs.multiview.refactoringCues/src/edu/pdx/cs/multiview/refactoringCues/views/ASTRegion.java - Eclipse SDK
File Edit Source Refactor Navigate Search Project Run Window Help
Team Synchr... Debug CVS Reposito... SVN Reposito... Java
ExtractTempAction.ja ExtractMethodFromExp ExtractExpressionAct ASTRegion.java emerson-06.txt
Writable Smart Insert

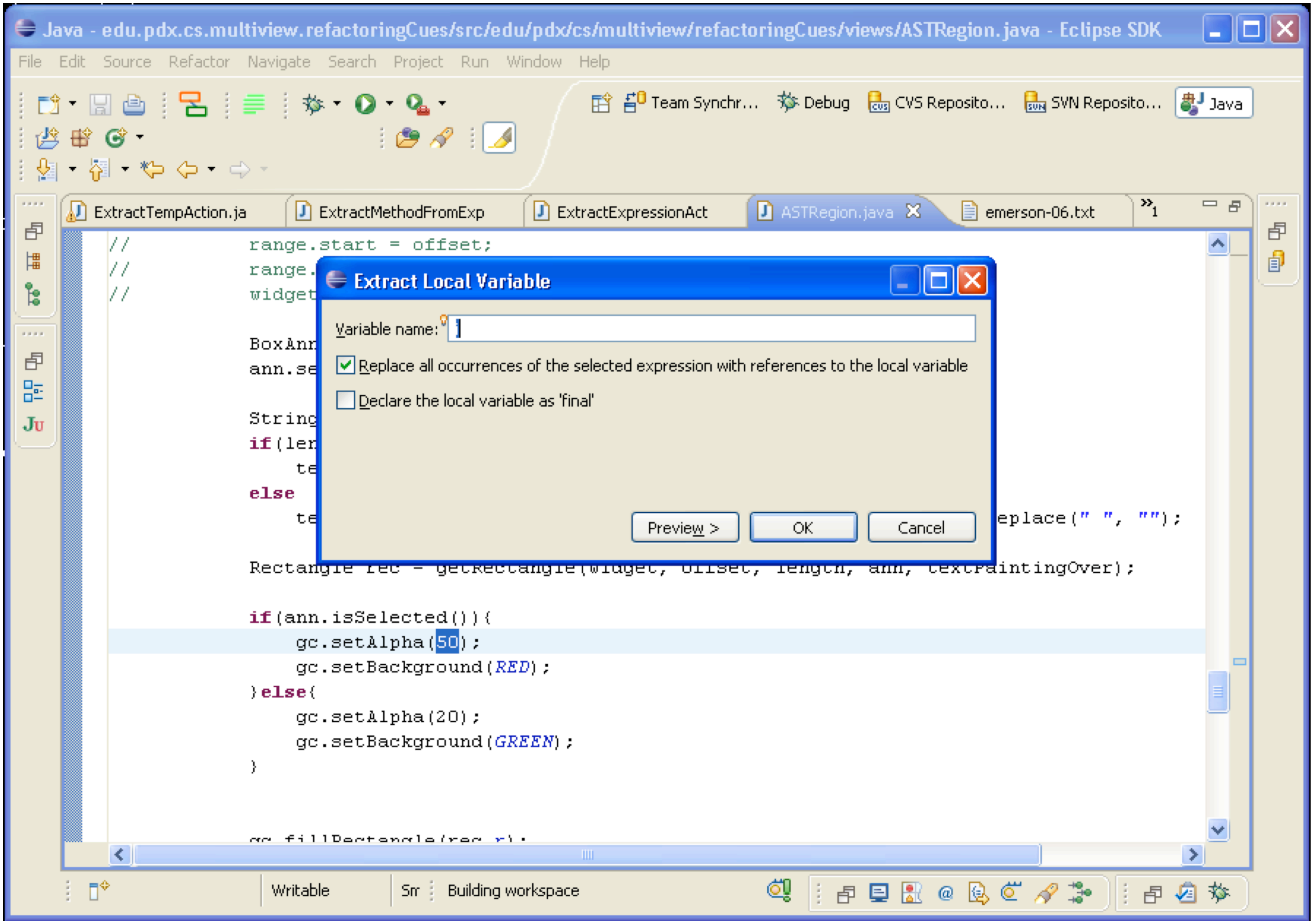
```java
//            range.start = offset;
//            range.length = length;
//            widget.setStyleRange(range);

              BoxAnnotation ann = (BoxAnnotation) annotation;
              ann.setWidget(widget);

              String textPaintingOver;
              if(length==0)
                   textPaintingOver = "";
              else
                   textPaintingOver = widget.getText(offset, offset+length-1).replace(" ", "");

              Rectangle rec = getRectangle(widget, offset, length, ann, textPaintingOver);

              if(ann.isSelected()){
                   gc.setAlpha(50);
                   gc.setBackground(RED);
              }else{
                   gc.setAlpha(20);
                   gc.setBackground(GREEN);
              }
```


Java - edu.pdx.cs.multiview.refactoringCues/src/edu/pdx/cs/multiview/refactoringCues/views/ASTRegion.java - Eclipse SDK
File Edit Source Refactor Navigate Search Project Run Window Help
Team Synchr... Debug CVS Reposito... SVN Reposito... Java
ExtractTempAction.ja
ExtractMethodFromExp
ExtractExpressionAct
ASTRegion.java
emerson-06.txt
Writable
Sn

```
//          range.start = offset;
//          range.length = length;
//          widget.setStyleRange(range);

            BoxAnnotation ann = (BoxAnnotation) annotation;
            ann.setWidget(widget);

            String textPaintingOver;
            if(length==0)
                textPaintingOver = "";
            else
                textPaintingOver = widget.getText(offset, offset+length-1).replace(" ", "");

            Rectangle rec = getRectangle(widget, offset, length, ann, textPaintingOver);

            if(ann.isSelected()){
                gc.setAlpha(50);
                gc.setBackground(RED);
            }else{
                gc.setAlpha(20);
                gc.setBackground(GREEN);
            }
```



```
//          range.start = offset;
//          range.length = length;
//          widget.setStyleRange(r

            BoxAnnotation ann = (B
            ann.setWidget(widget);

            String textPaintingOve
            if(length==0)
                textPaintingOver =
            else
                textPaintingOver =                     fset+length-1).replace(" ", "");

            Rectangle rec = getRec                     gth, ann, textPaintingOver);

            if(ann.isSelected()){
                gc.setAlpha(50);
                gc.setBackground(
            }else{
                gc.setAlpha(20);
                gc.setBackground(
            }
```


Java - edu.pdx.cs.multiview.refactoringCues/src/edu/pdx/cs/multiview/refactoringCues/views/ASTRegion.java - Eclipse SDK

File Edit Source Refactor Navigate Search Project Run Window Help
Team Synchr... Debug CVS Reposito... SVN Reposito... Java
ExtractTempAction.ja ExtractMethodFromExp ExtractExpressionAct ASTRegion.java emerson-06.txt

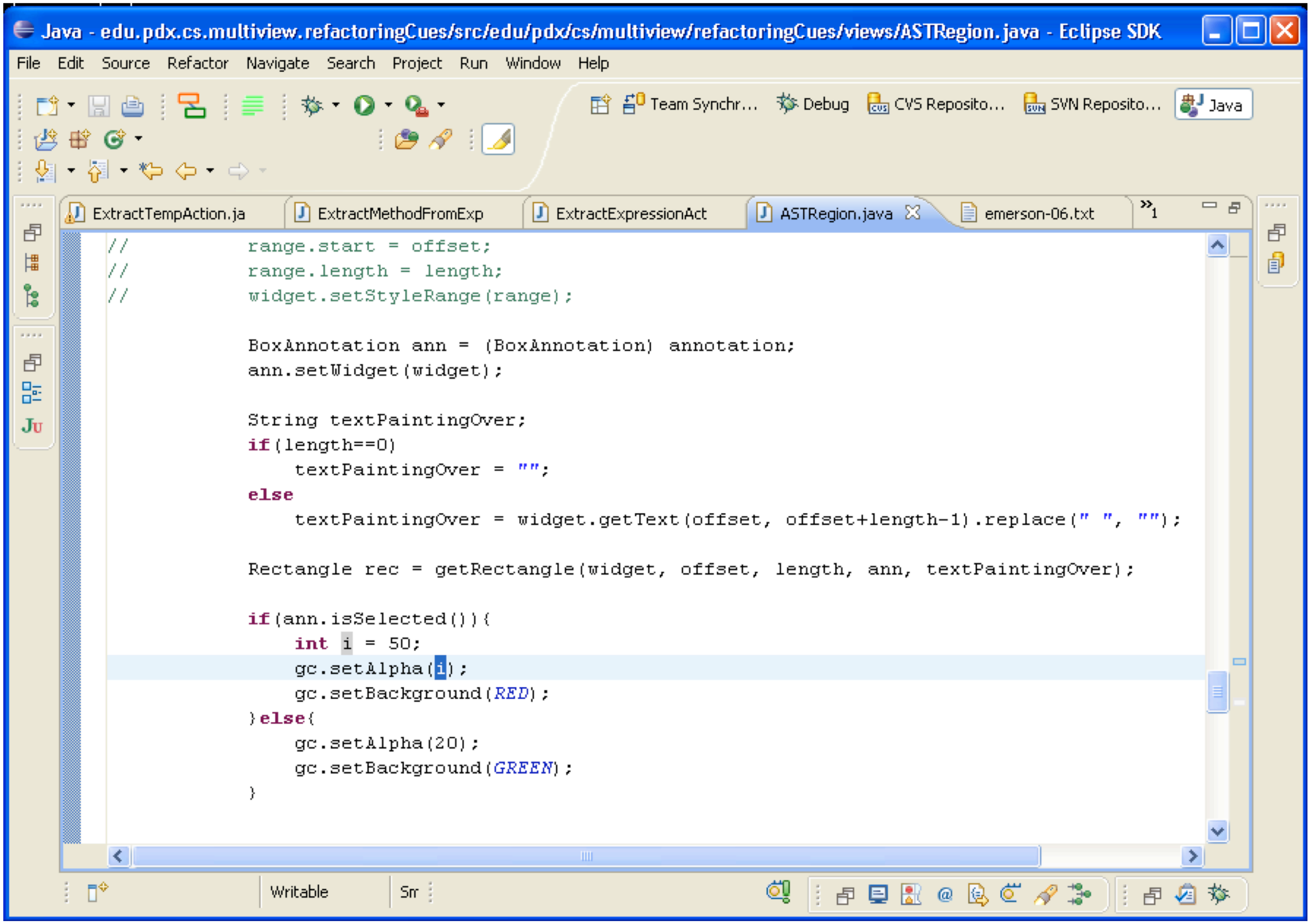
```
//          range.start = offset;
//          range.
//          widget

            BoxAnn
            ann.se

            String
            if(len
                te
            else
                te                                          eplace(" ", "");

            Rectangle rec = getRectangle(widget, offset, length, ann, textPaintingOver);

            if(ann.isSelected()){
                gc.setAlpha(50);
                gc.setBackground(RED);
            }else{
                gc.setAlpha(20);
                gc.setBackground(GREEN);
            }


            gc.fillRectangle(rec,r);
```

Extract Local Variable

Variable name:

[✓] Replace all occurrences of the selected expression with references to the local variable

[ ] Declare the local variable as 'final'



Java - edu.pdx.cs.multiview.refactoringCues/src/edu/pdx/cs/multiview/refactoringCues/views/ASTRegion.java - Eclipse SDK
File Edit Source Refactor Navigate Search Project Run Window Help
Team Synchr... Debug CVS Reposito... SVN Reposito... Java
ExtractTempAction.ja ExtractMethodFromExp ExtractExpressionAct ASTRegion.java emerson-06.txt
Writable
Sm

```
//          range.start = offset;
//          range.length = length;
//          widget.setStyleRange(range);

            BoxAnnotation ann = (BoxAnnotation) annotation;
            ann.setWidget(widget);

            String textPaintingOver;
            if(length==0)
                textPaintingOver = "";
            else
                textPaintingOver = widget.getText(offset, offset+length-1).replace(" ", "");

            Rectangle rec = getRectangle(widget, offset, length, ann, textPaintingOver);

            if(ann.isSelected()){
                int i = 50;
                gc.setAlpha(i);
                gc.setBackground(RED);
            }else{
                gc.setAlpha(20);
                gc.setBackground(GREEN);
            }
```

Java Tools

Eclipse
JBuilder
Netbeans
IntilliJ IDEA
RefactorIT
X-Develop / CodeGuide
X-Refactory*
JFactor *
JRefactory *
Transmogrify*
JavaRefactor *

* Indicates a "dead" tool

Refactoring Tool under-use

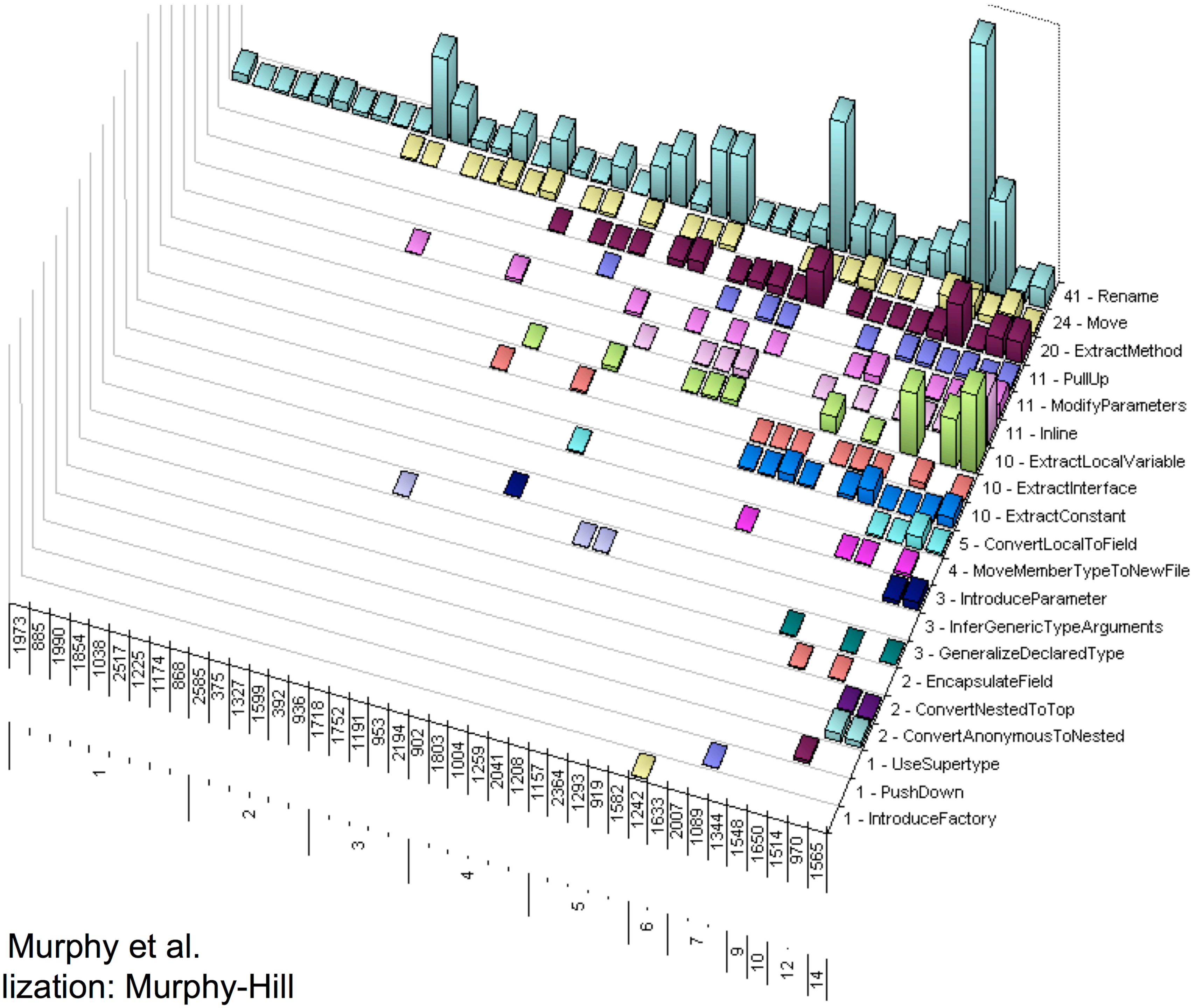
16 Object-Oriented students
- Only 2 used Refactoring Tools

37 users of Eclipse in PSU lab
- 2 used Refactoring Tools

112 participants at Agile Open NW 2007:
- Of 72 programmers, 63 have tools available some of the time
- They claimed an average of 68% use of the tools
- Why not 100%?

Refactoring Tool (Under) Usage

- Murphy et al. looked at 41 Programmers
  - Only two refactoring tools used by “most” programmers: Rename and Move
  - Median number of refactoring hotkeys used by programmers is 2; maximum is 5
- Disconnect between refactoring desires and tool use:
  - According to Mantayla, programmers overwhelmingly *want* to Extract Method
  - But according to Murphy, programmers overwhelmingly *perform* Rename



Data: Murphy et al.
Vizualization: Murphy-Hill

Agile Open Northwest 2007: When performing a refactoring where a tool is available but you choose not to use it, what usually prevents you?

***44 responses**. The tool isn't flexible enough—it doesn't do quite what I want.*

***26 responses.** I never really learned how to use that particular refactoring tool / I don't know what tool to use.*

***24 responses**. I can do it faster by hand.*

***13 responses**. I don't trust the tool to be correct.*

***7 responses**. The tool will probably mutilate my code.*

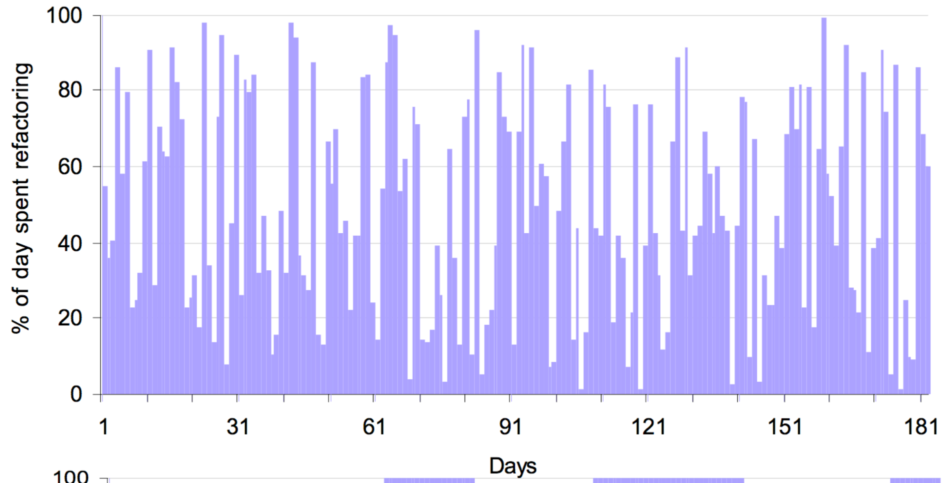
***2 responses**. My code base is so large that the refactoring tool takes too long.*

Other: *• Habit. • Menu too big. • Avoid GUIs—Keybindings only! • Prefer to be aware of the changes myself. • Hard to trust the refactored code, even if it applies. • Usually I do multistep refactoring—tools do one step at a time.*

Two kinds of Refactoring

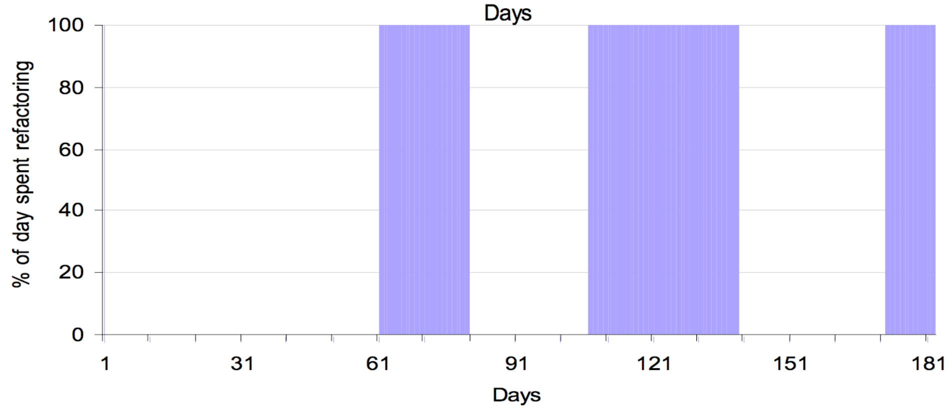
Floss Refactoring

- Programmers refactor constantly to maintain healthy software
- Refactoring is interleaved with programming



Root Canal Refactoring

- Programmers refactor in clumps to fix unhealthy software.
- Programming and refactoring are distinct activities.



Floss vs. Root Canal

Floss

- Impromptu: refactor whenever you think the code needs it
- You know exactly what code you're going to refactor, because you're working on it
- Supported by Fowler (1999), Parnin+ (2006), Hayashi+ (2006)

Root Canal

- Planned: set aside time for refactoring
- You don't know what needs to be refactored, but past experience indicates that future changes will be difficult
- Assumed by Van Emden+ (2002) and Kataoka+ (2002); case study by Pizka (2004)

Why you should Floss rather than waiting for a Root Canal

Your dentist says so:

- “Refactoring is something you do all the time in little bursts. You don’t decide to refactor, you refactor because you want to do something else, and refactoring helps you do that other thing.”

  Martin Fowler, *Refactoring*

- “Avoid the temptation to stop work and refactor for several weeks. Even the most disciplined team inadvertently takes on design debt, so eliminating debt needs to be an ongoing activity. Have your team get used to refactoring as part of their daily work.”

  James Shore, “Design Debt”

Why you should Floss rather than waiting for a Root Canal

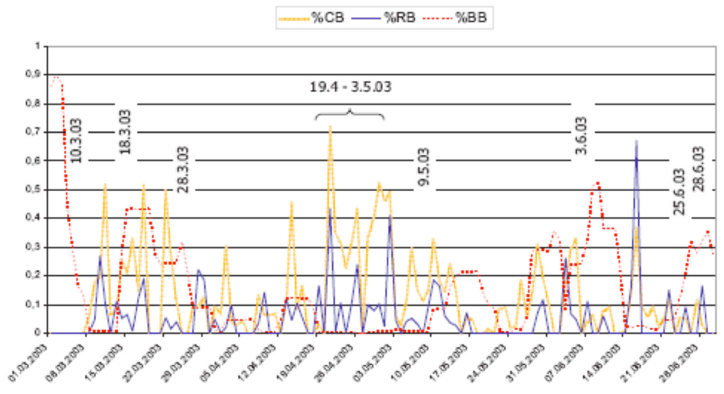


Figure 8: JEDIT: March to June 2003

Your friends are doing it!

- Weißgerber and Diehl (MSR 2006) looked at JUnit, ArgoUML and JEdit:

  "It turned out that in all three projects, there are no days which only contain refactorings. This is quite surprising, as we would expect that at least in small projects like JUnit there are phases in a project when only refactorings have been done to enhance the program structure."

- Murphy et al. (Software 2006) observed 41 Eclipse developers:

  2672 repository commits

  At most 9 out of 283 iterations were pure refactoring

Portland State UNIVERSITY

Why you should Floss rather than waiting for a Root Canal

You are becoming Agile!

- Continuous design and continuous refactoring are key practices for Agile programmers

"We keep the code simple *at all times*. This minimizes the investment in excess framework and support code. We retain the necessary flexibility through refactoring."

Jeffries, Anderson & Hendrickson
Extreme Programming Installed, 2000

Root Canal: Ineffective

- Pizka (2004) describes a root canal refactoring over 5 months; concludes that the time was mostly wasted.
- Bourquin and Keller (2007) describe a root canal refactoring over 7 months
  - few objectively positive results
  - dramatic increase in duplicated code.

Why the Distinction Matters

Claim: a tool built for root canal refactoring will not be very usable for floss refactoring, and to a lesser degree, vice versa.

Example

Floss Tool	Root Canal Tool
Hayashi's incremental smell detector (2006)	Jcosmo smell detector (Van Emden 2002)
• Runs continuously in background. • Reports on the code on which programmers are working	• Runs as a batch job on request. • Provides information about the whole code base

So What's Really Wrong with Refactoring Tools?

Tools don't always fit with the way programmers want to refactor…

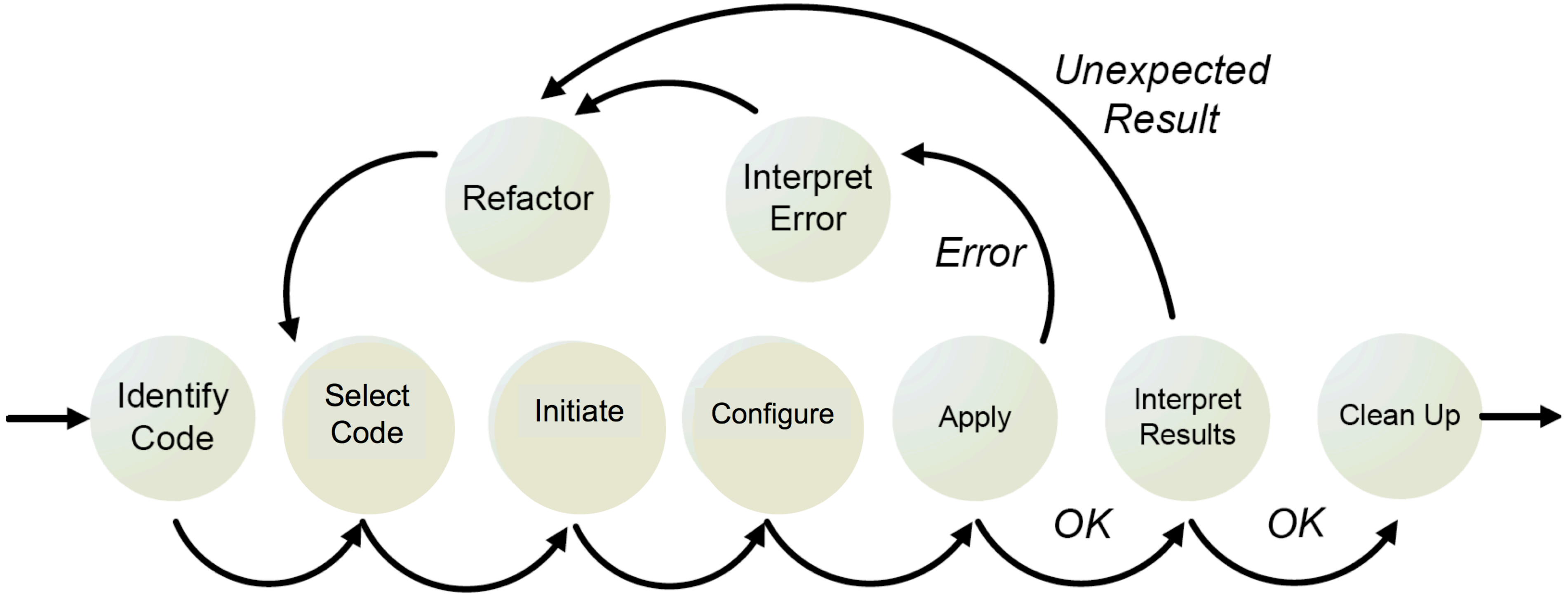
- data from Open Agile Survey:

  94 responses indicated a usability problem, *vs*

  22 indicated a technical problem

…so programmers snub the tools and refactor by hand.

How *do* Programmers Refactor?



Portland State UNIVERSITY

So, what have people been doing about it?

Building better tools!

- In industry
- In research groups

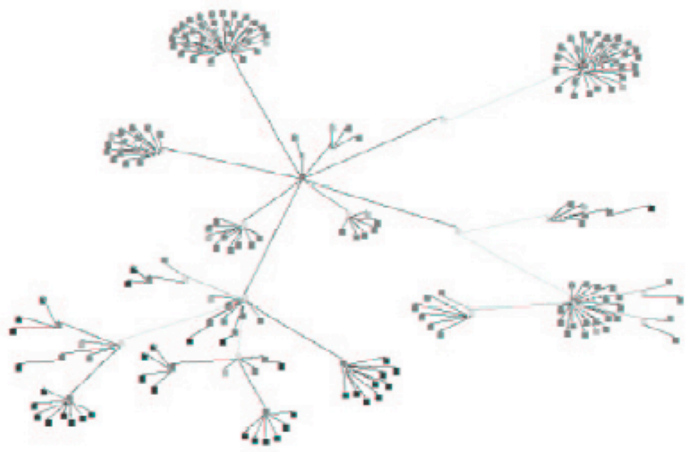
Industrial Improvements (Code Identification)

Automated smell detection

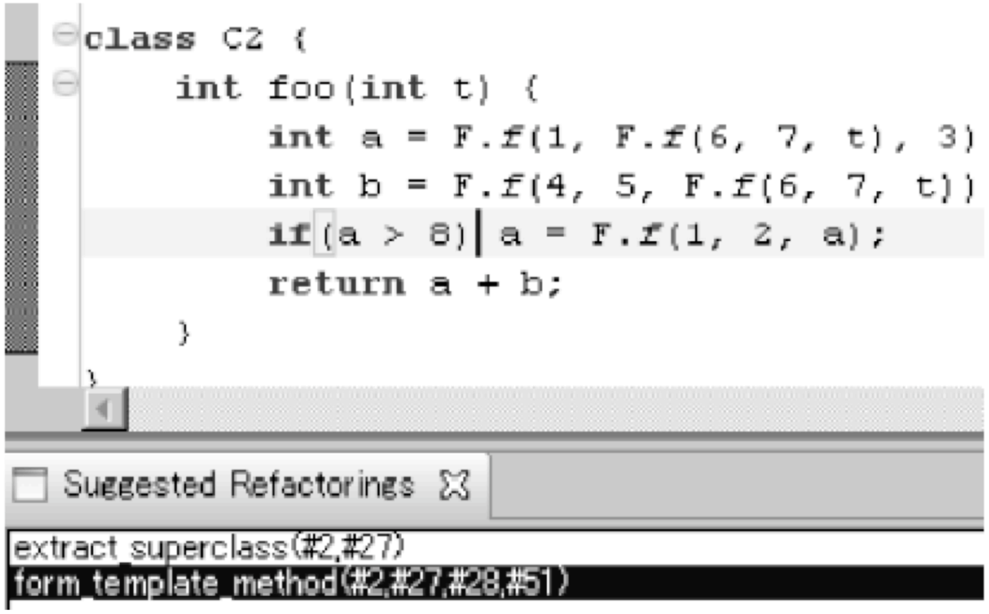
- Eclipse’s Test and Performance Tools
- IntelliJ’s code inspections

Industrial Improvements (Code Identification)

JCosmo



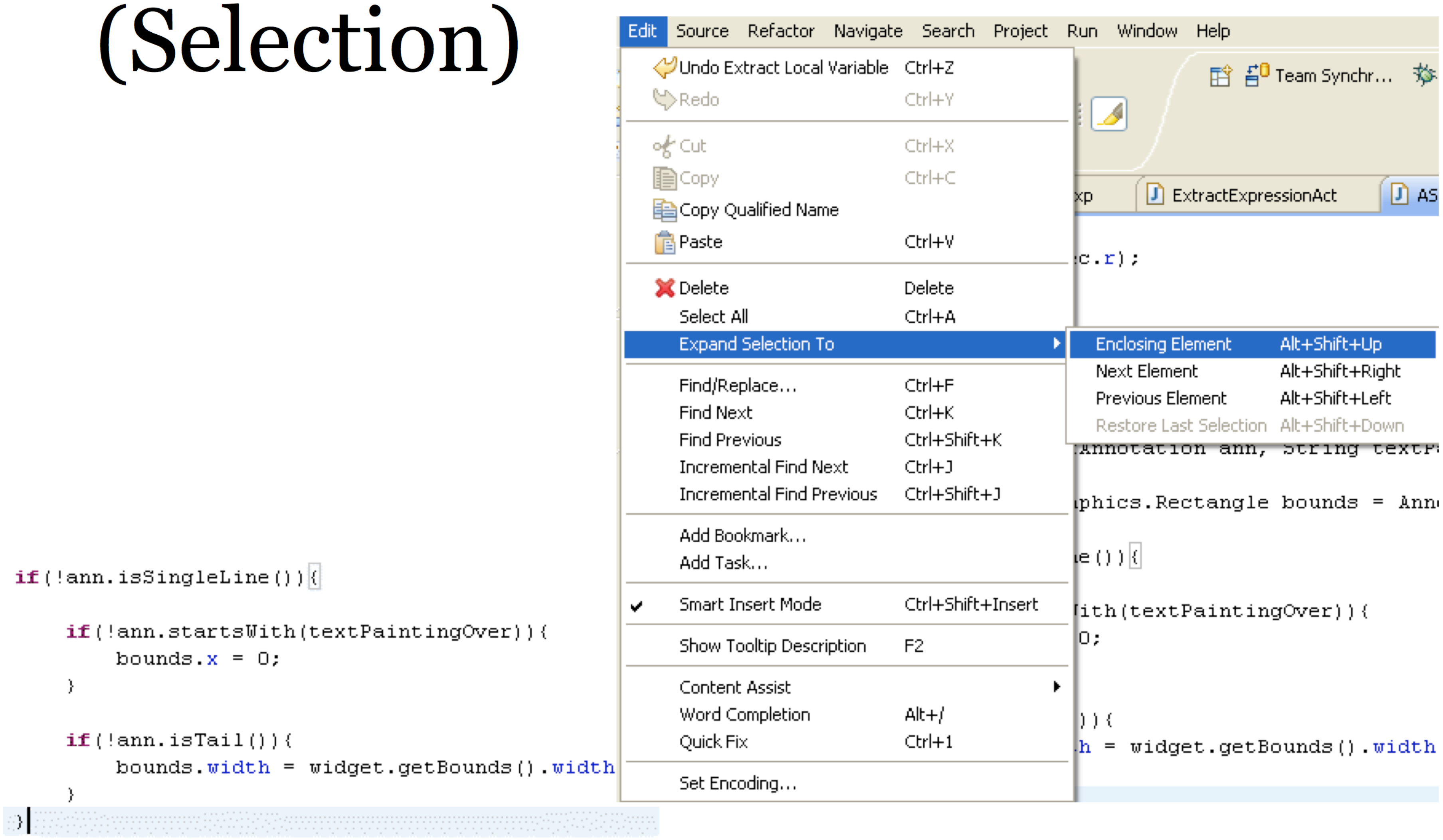
Hayashi and Colleagues



```
class C2 {
    int foo(int t) {
        int a = F.f(1, F.f(6, 7, t), 3)
        int b = F.f(4, 5, F.f(6, 7, t))
        if(a > 8)| a = F.f(1, 2, a);
        return a + b;
    }
}
```

extract_superclass(#2,#27)
form_template_method(#2,#27,#28,#51)

Industrial Improvements (Selection)


Edit Source Refactor Navigate Search Project Run Window Help
Undo Extract Local Variable Ctrl+Z
Redo Ctrl+Y
Cut Ctrl+X
Copy Ctrl+C
Copy Qualified Name
Paste Ctrl+V
Delete Delete
Select All Ctrl+A
Expand Selection To
Enclosing Element Alt+Shift+Up
Next Element Alt+Shift+Right
Previous Element Alt+Shift+Left
Restore Last Selection Alt+Shift+Down
Find/Replace... Ctrl+F
Find Next Ctrl+K
Find Previous Ctrl+Shift+K
Incremental Find Next Ctrl+J
Incremental Find Previous Ctrl+Shift+J
Add Bookmark...
Add Task...
✓ Smart Insert Mode Ctrl+Shift+Insert
Show Tooltip Description F2
Content Assist
Word Completion Alt+/
Quick Fix Ctrl+1
Set Encoding...
Team Synchr...
xp
ExtractExpressionAct
AS

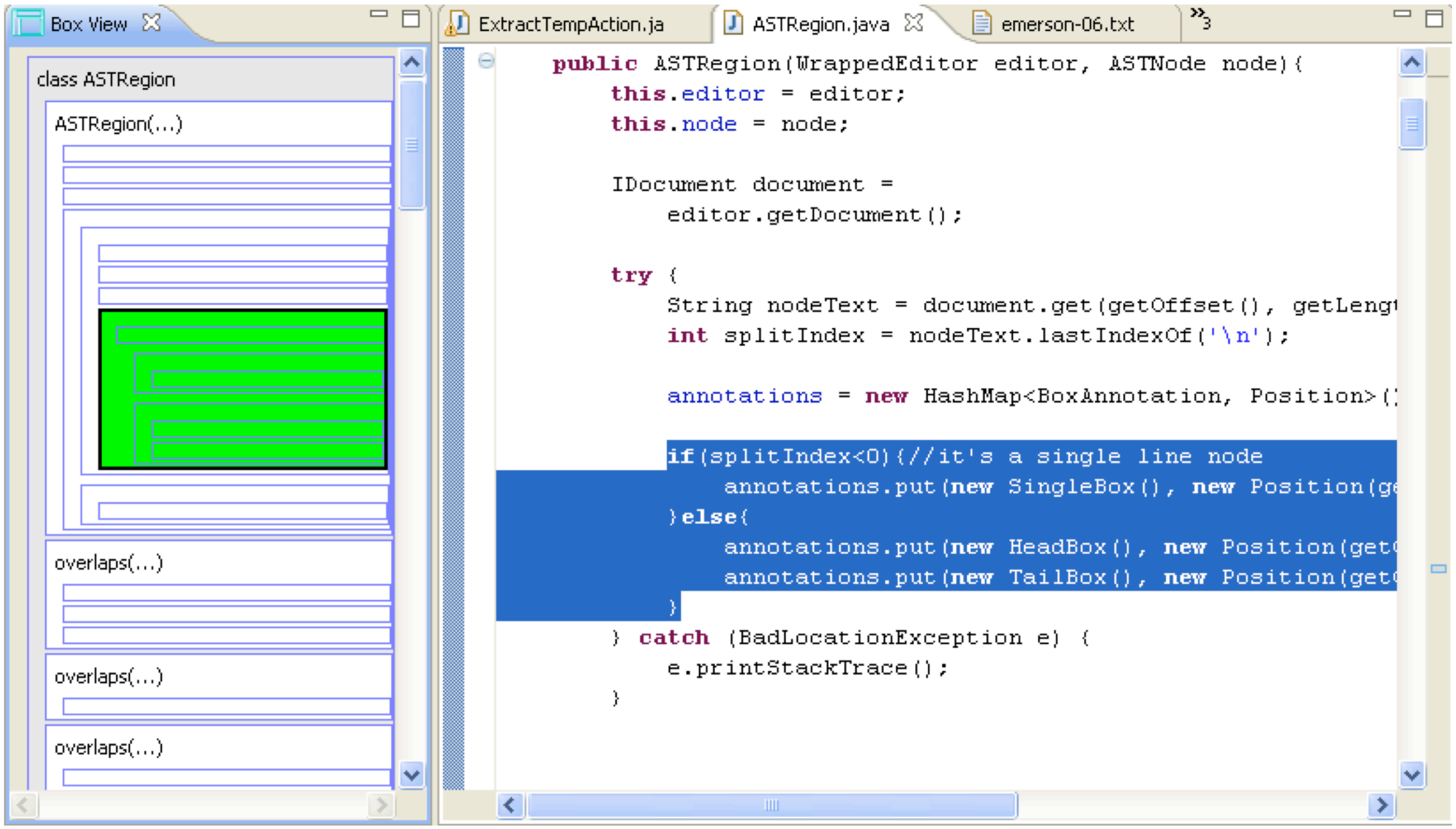
```
c.r);
Annotation ann, String textP
phics.Rectangle bounds = Ann
e()){
ith(textPaintingOver)){
0;
)){
h = widget.getBounds().width
```

```
if(!ann.isSingleLine()){

    if(!ann.startsWith(textPaintingOver)){
        bounds.x = 0;
    }

    if(!ann.isTail()){
        bounds.width = widget.getBounds().width
    }
}
```

Portland State UNIVERSITY

Research Improvements (Selection)

BoxView

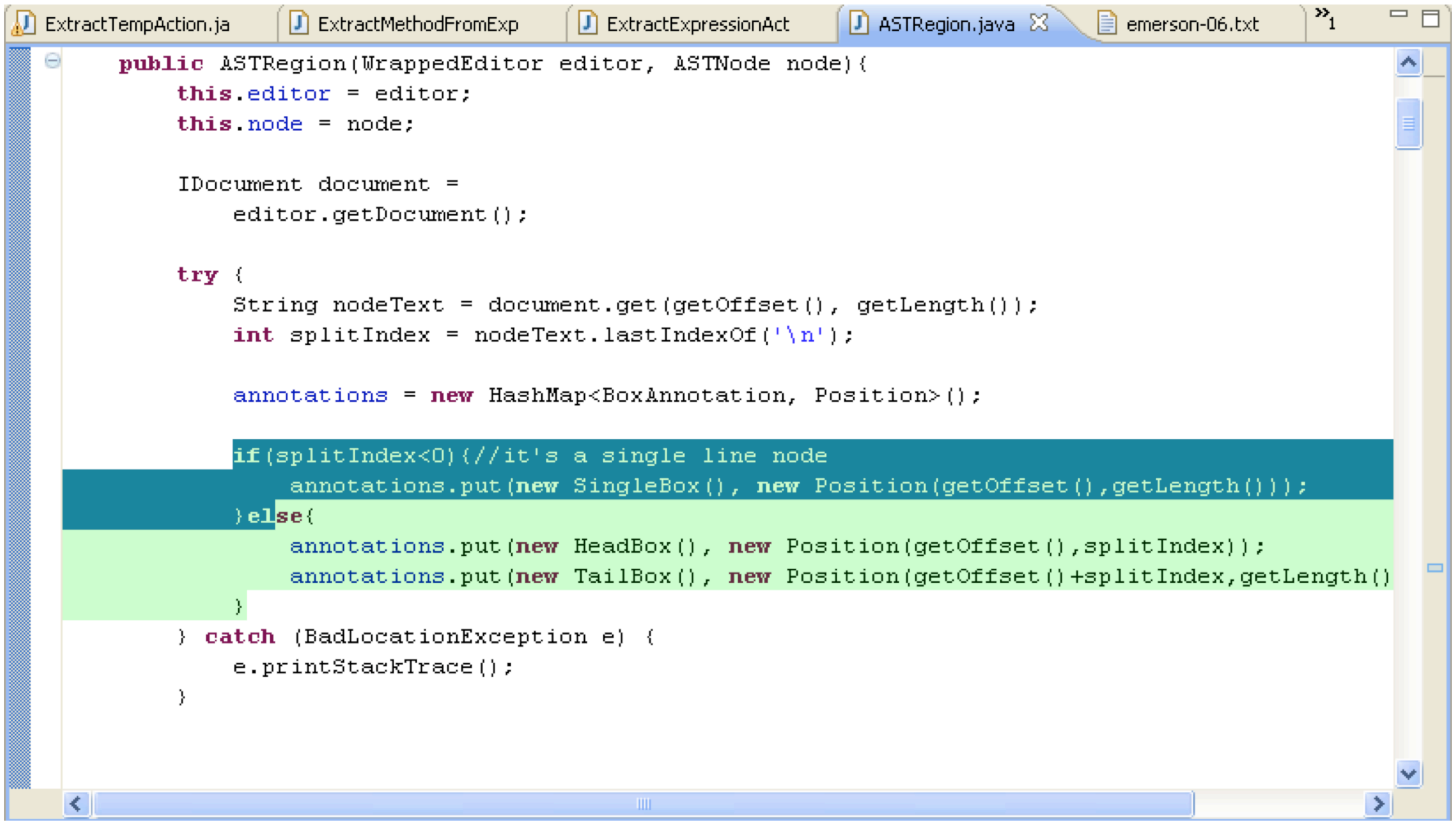


```
    public ASTRegion(WrappedEditor editor, ASTNode node){
        this.editor = editor;
        this.node = node;

        IDocument document =
            editor.getDocument();

        try {
            String nodeText = document.get(getOffset(), getLengt
            int splitIndex = nodeText.lastIndexOf('\n');

            annotations = new HashMap<BoxAnnotation, Position>(

            if(splitIndex<0){//it's a single line node
                annotations.put(new SingleBox(), new Position(ge
            }else{
                annotations.put(new HeadBox(), new Position(get
                annotations.put(new TailBox(), new Position(get
            }
        } catch (BadLocationException e) {
            e.printStackTrace();
        }
```

Portland State UNIVERSITY

Research Improvements (Selection)

SelectionAssist

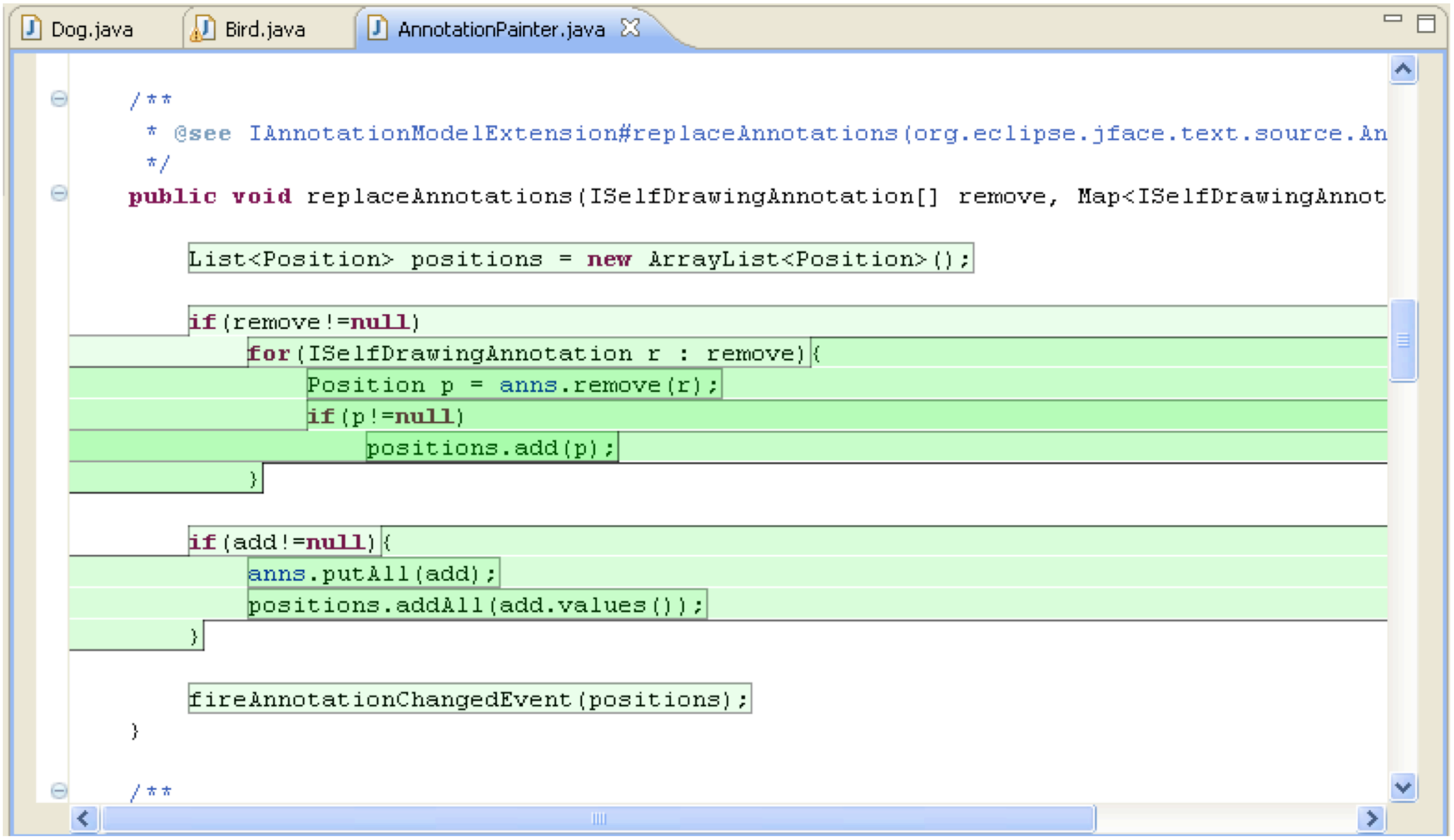


```
public ASTRegion(WrappedEditor editor, ASTNode node){
    this.editor = editor;
    this.node = node;

    IDocument document =
        editor.getDocument();

    try {
        String nodeText = document.get(getOffset(), getLength());
        int splitIndex = nodeText.lastIndexOf('\n');

        annotations = new HashMap<BoxAnnotation, Position>();

        if(splitIndex<0){//it's a single line node
            annotations.put(new SingleBox(), new Position(getOffset(),getLength()));
        }else{
            annotations.put(new HeadBox(), new Position(getOffset(),splitIndex));
            annotations.put(new TailBox(), new Position(getOffset()+splitIndex,getLength()
        }
    } catch (BadLocationException e) {
        e.printStackTrace();
    }
```

Portland State UNIVERSITY

Research Improvements (Selection)

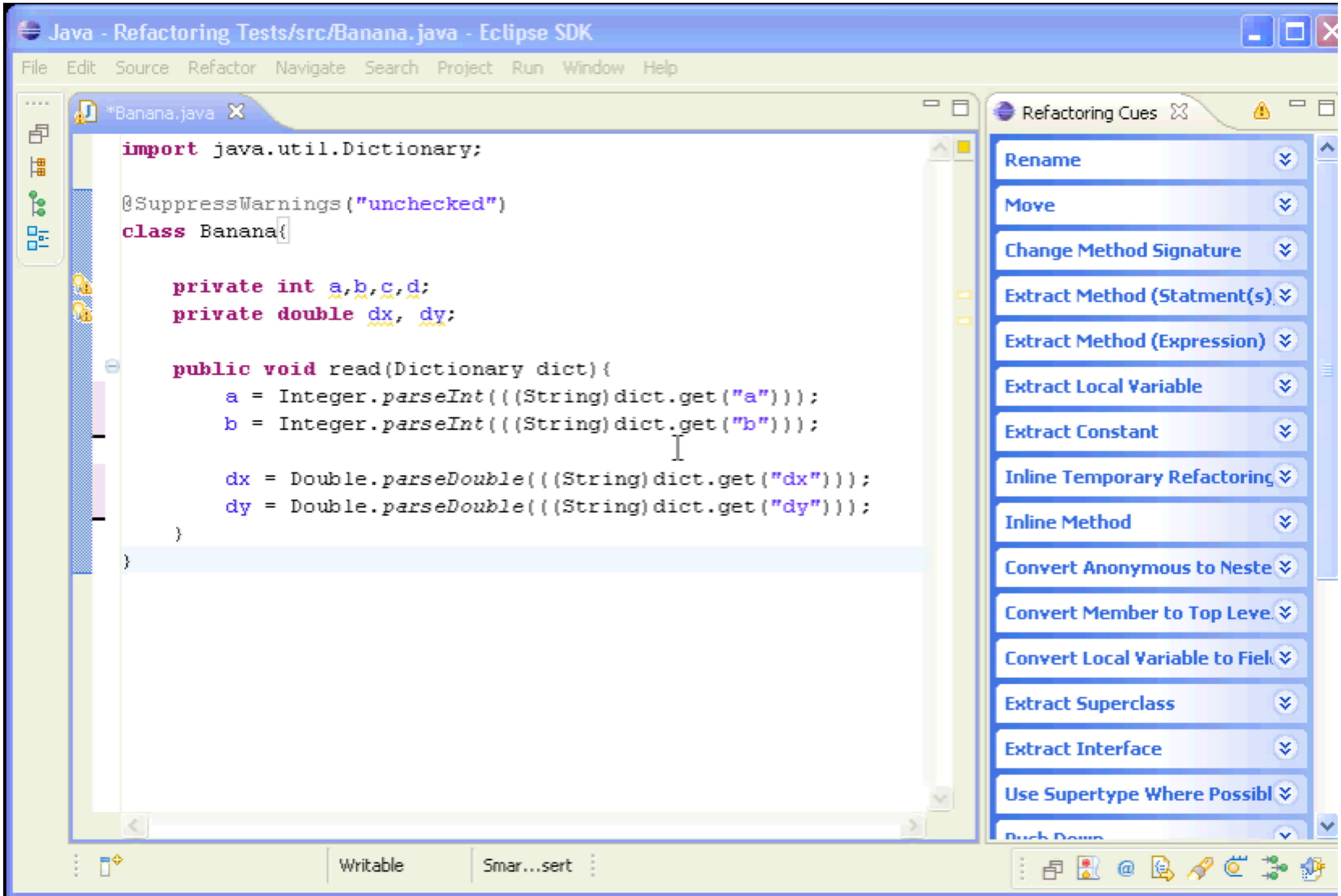
Refactoring Cues



Dog.java | Bird.java | AnnotationPainter.java

```
    /**
     * @see IAnnotationModelExtension#replaceAnnotations(org.eclipse.jface.text.source.An
     */
    public void replaceAnnotations(ISelfDrawingAnnotation[] remove, Map<ISelfDrawingAnnot

        List<Position> positions = new ArrayList<Position>();

        if(remove!=null)
            for(ISelfDrawingAnnotation r : remove){
                Position p = anns.remove(r);
                if(p!=null)
                    positions.add(p);
            }

        if(add!=null){
            anns.putAll(add);
            positions.addAll(add.values());
        }

        fireAnnotationChangedEvent(positions);
    }

    /**
```

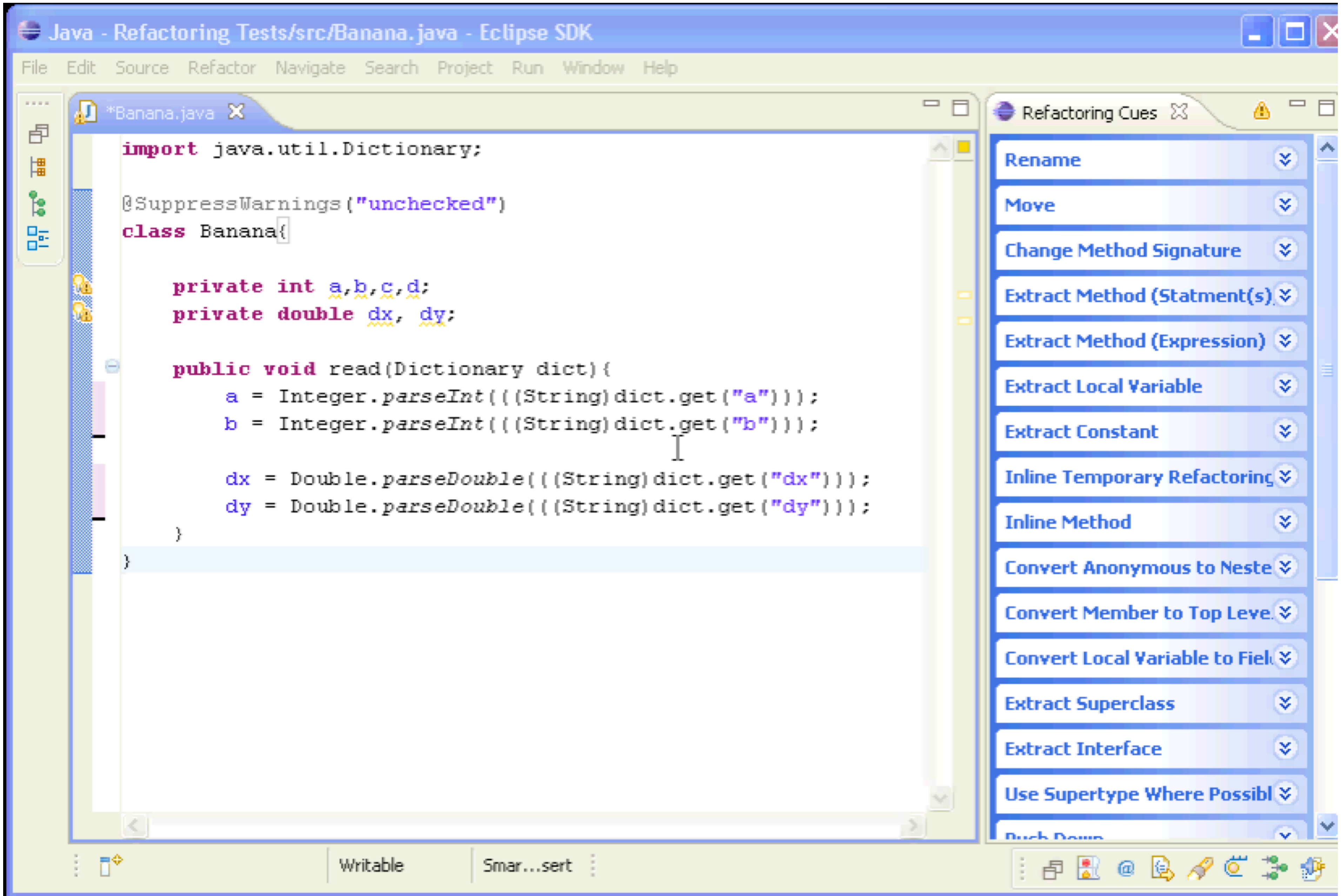
Portland State UNIVERSITY



Java - Refactoring Tests/src/Banana.java - Eclipse SDK
File Edit Source Refactor Navigate Search Project Run Window Help
*Banana.java
Refactoring Cues
Rename
Move
Change Method Signature
Extract Method (Statment(s)
Extract Method (Expression)
Extract Local Variable
Extract Constant
Inline Temporary Refactoring
Inline Method
Convert Anonymous to Neste
Convert Member to Top Leve
Convert Local Variable to Fiel
Extract Superclass
Extract Interface
Use Supertype Where Possibl
Writable
Smar…sert

```java
import java.util.Dictionary;

@SuppressWarnings("unchecked")
class Banana{

    private int a,b,c,d;
    private double dx, dy;

    public void read(Dictionary dict){
        a = Integer.parseInt(((String)dict.get("a")));
        b = Integer.parseInt(((String)dict.get("b")));

        dx = Double.parseDouble(((String)dict.get("dx")));
        dy = Double.parseDouble(((String)dict.get("dy")));
    }
}
```

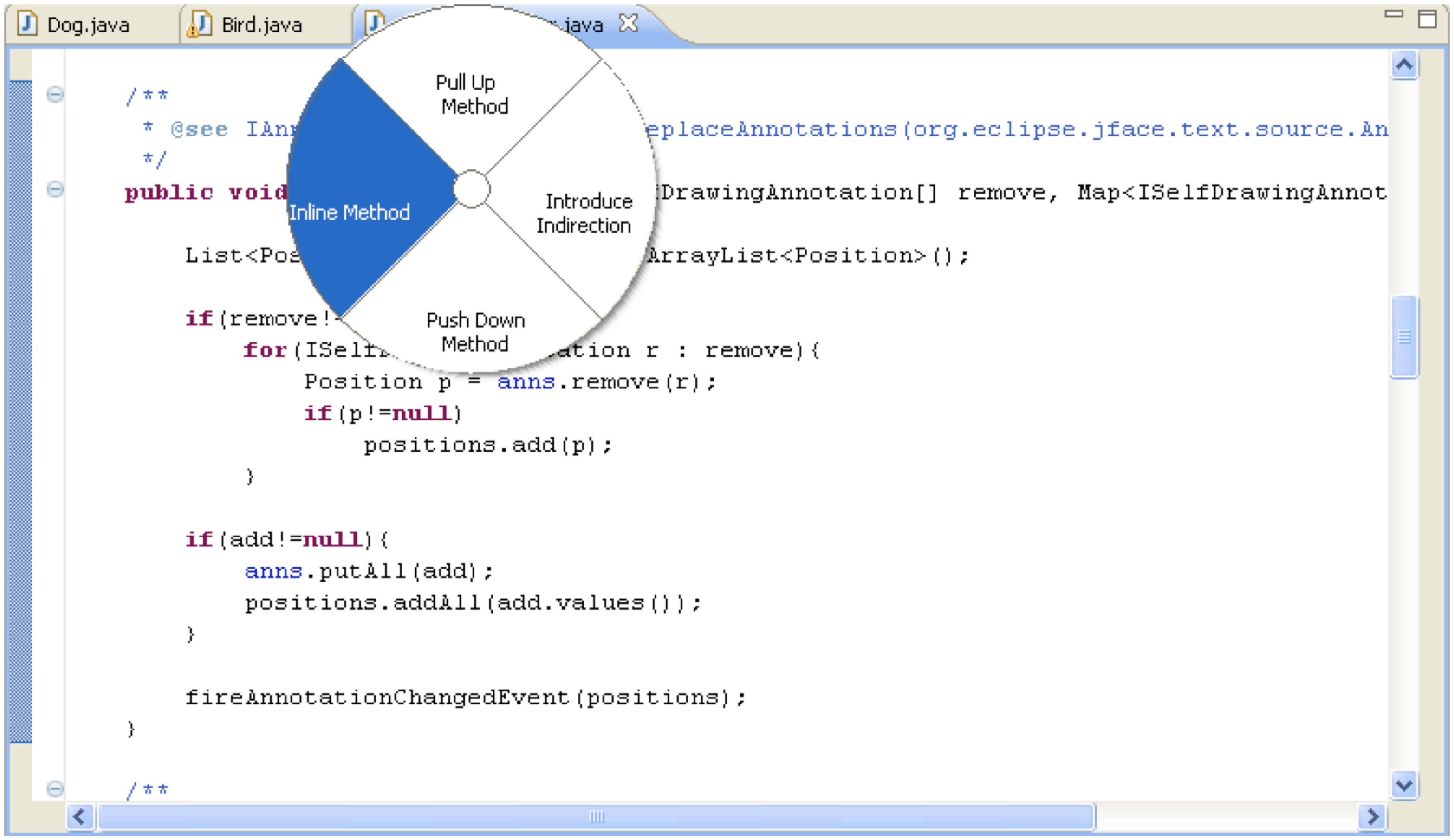
Portland State UNIVERSITY


Java - Refactoring Tests/src/Banana.java - Eclipse SDK
File Edit Source Refactor Navigate Search Project Run Window Help
*Banana.java
Refactoring Cues
Writable
Smar...sert

```
import java.util.Dictionary;

@SuppressWarnings("unchecked")
class Banana{

    private int a,b,c,d;
    private double dx, dy;

    public void read(Dictionary dict){
        a = Integer.parseInt(((String)dict.get("a")));
        b = Integer.parseInt(((String)dict.get("b")));

        dx = Double.parseDouble(((String)dict.get("dx")));
        dy = Double.parseDouble(((String)dict.get("dy")));
    }
}
```

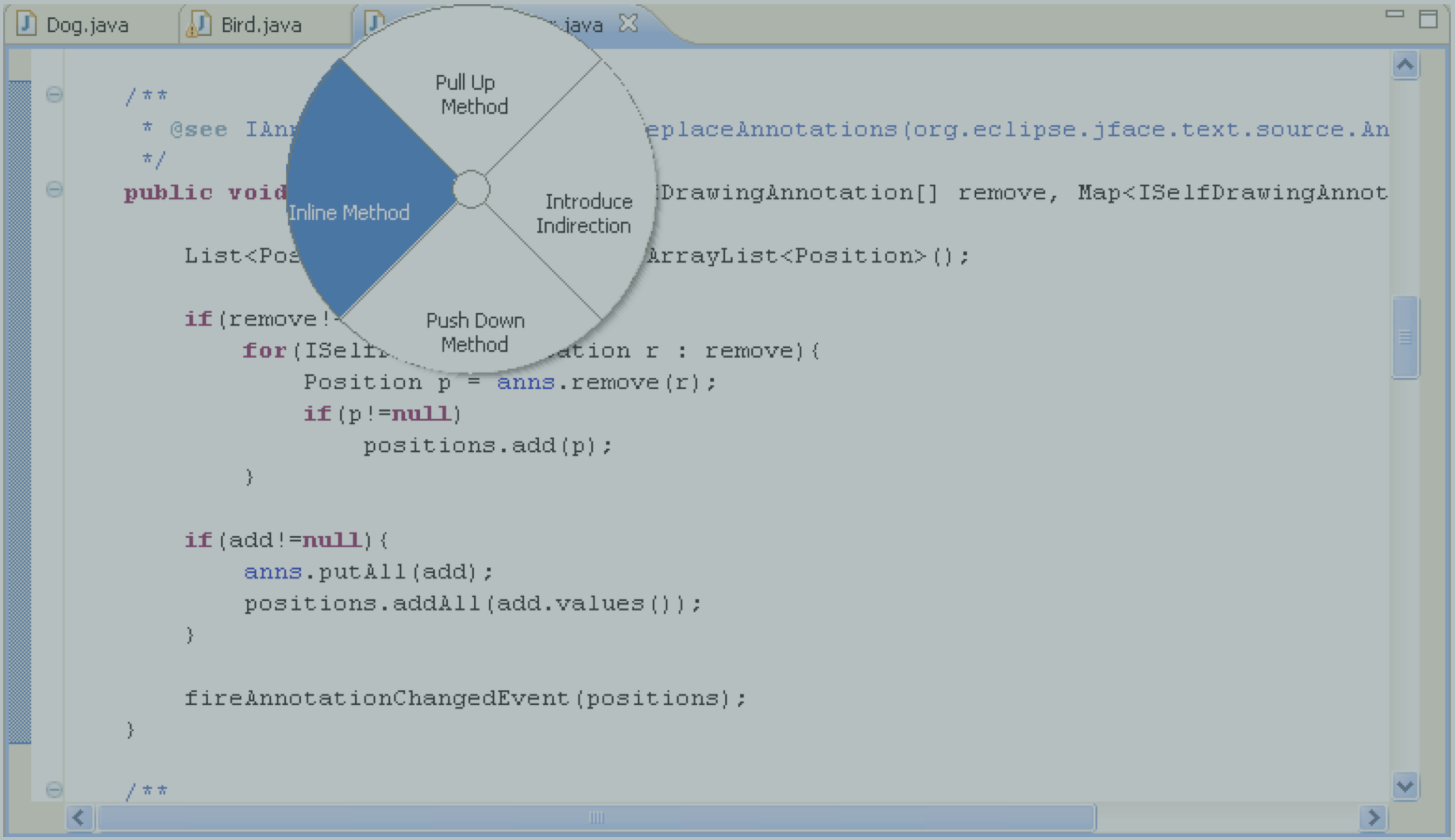
- Rename
- Move
- Change Method Signature
- Extract Method (Statment(s)
- Extract Method (Expression)
- Extract Local Variable
- Extract Constant
- Inline Temporary Refactoring
- Inline Method
- Convert Anonymous to Neste
- Convert Member to Top Leve
- Convert Local Variable to Fiel
- Extract Superclass
- Extract Interface
- Use Supertype Where Possibl

Research Improvements (Activation)



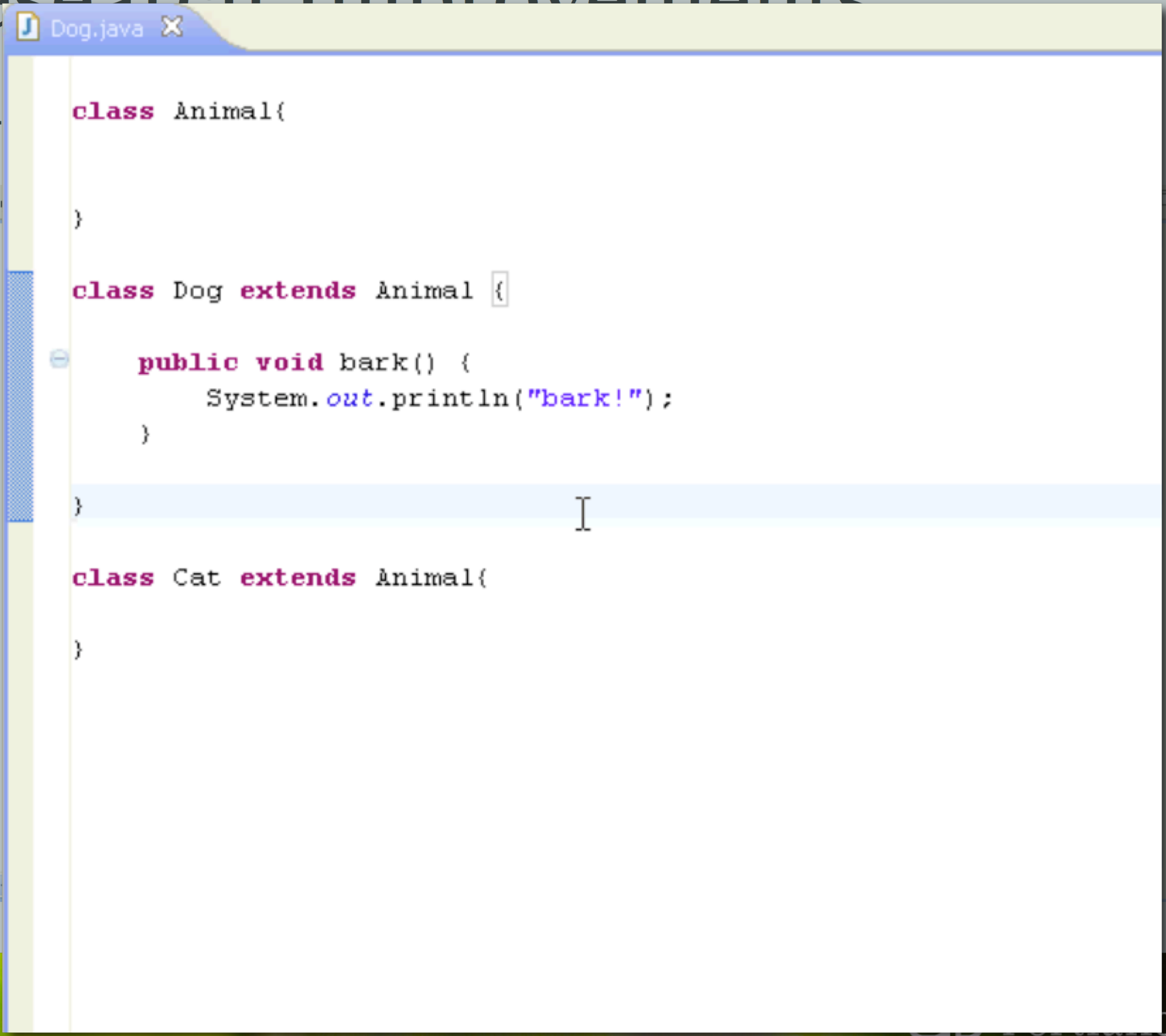
```
/**
 * @see IAnn                         eplaceAnnotations(org.eclipse.jface.text.source.An
 */
public void                         DrawingAnnotation[] remove, Map<ISelfDrawingAnnot

    List<Pos                        ArrayList<Position>();

    if(remove!-
        for(ISelf              ation r : remove){
            Position p = anns.remove(r);
            if(p!=null)
                positions.add(p);
        }

    if(add!=null){
        anns.putAll(add);
        positions.addAll(add.values());
    }

    fireAnnotationChangedEvent(positions);
}

/**
```

Research Improvements (Activation)



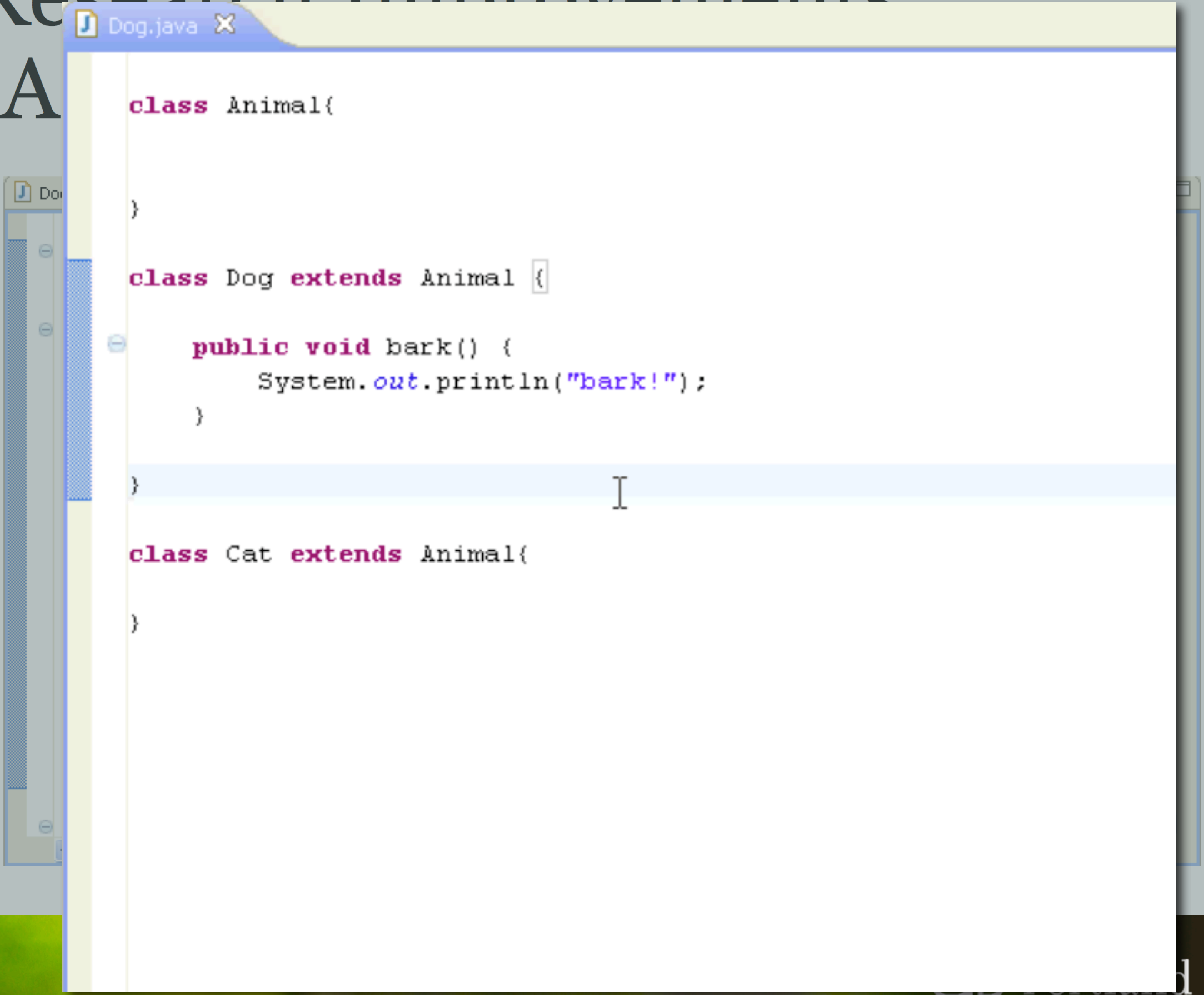
```
/**
 * @see IAnn                    eplaceAnnotations(org.eclipse.jface.text.source.An
 */
public void                    DrawingAnnotation[] remove, Map<ISelfDrawingAnnot

    List<Pos                   ArrayList<Position>();

    if(remove!=
        for(ISelf              ation r : remove){
            Position p = anns.remove(r);
            if(p!=null)
                positions.add(p);
        }

    if(add!=null){
        anns.putAll(add);
        positions.addAll(add.values());
    }

    fireAnnotationChangedEvent(positions);
}

/**
```

Research Improvements (A


Dog.java

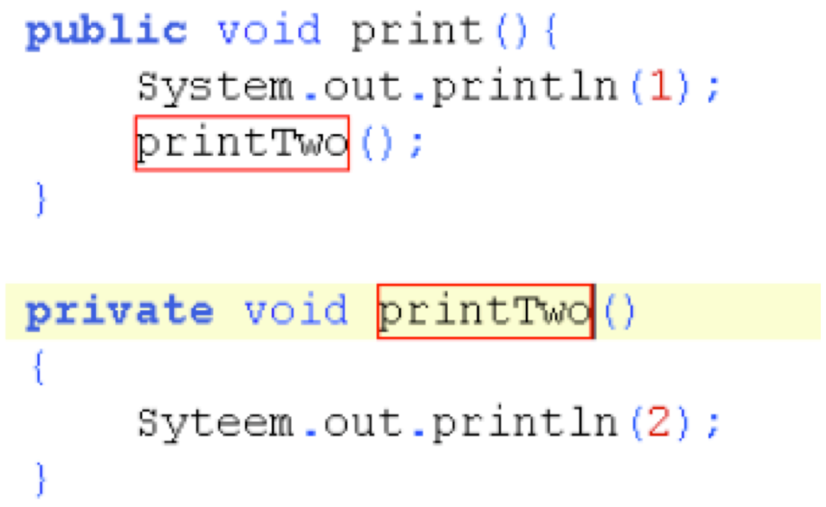
```
class Animal{


}

class Dog extends Animal {

    public void bark() {
        System.out.println("bark!");
    }

}

class Cat extends Animal{

}
```

Research Improvements (A



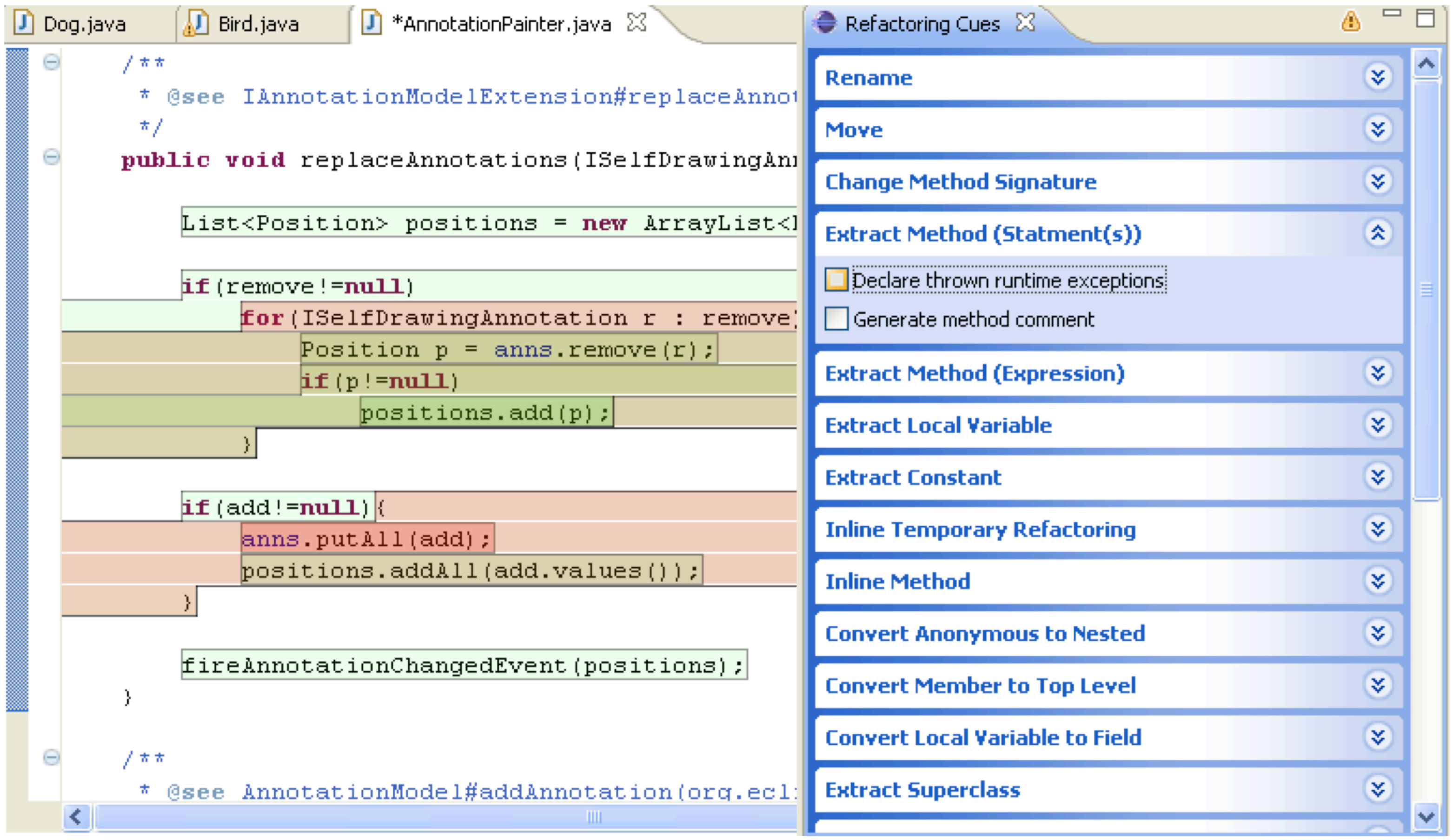
```
class Animal{


}

class Dog extends Animal {

    public void bark() {
        System.out.println("bark!");
    }

}

class Cat extends Animal{

}
```

Industrial Improvements (Configuration)



```
public void print(){
    System.out.println(1);
    printTwo();
}

private void printTwo()
{
    Syteem.out.println(2);
}
```

Research Improvements (Configuration)



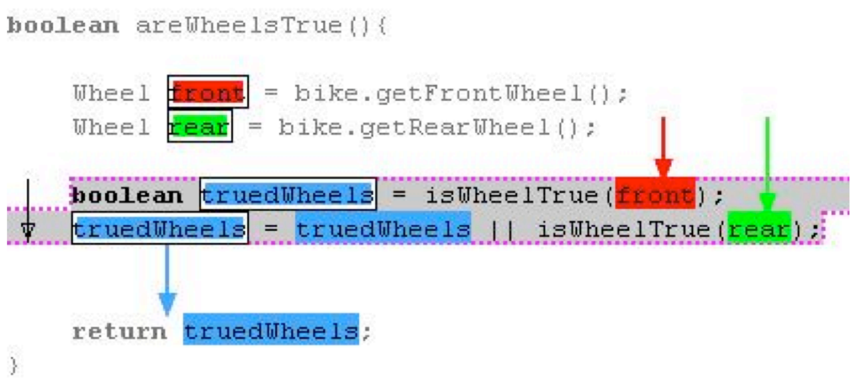
Dog.java | Bird.java | *AnnotationPainter.java

```
/**
 * @see IAnnotationModelExtension#replaceAnno
 */
public void replaceAnnotations(ISelfDrawingAn

    List<Position> positions = new ArrayList<

    if(remove!=null)
        for(ISelfDrawingAnnotation r : remove)
            Position p = anns.remove(r);
            if(p!=null)
                positions.add(p);
        }

    if(add!=null){
        anns.putAll(add);
        positions.addAll(add.values());
    }

    fireAnnotationChangedEvent(positions);
}

/**
 * @see AnnotationModel#addAnnotation(org.ecl
```

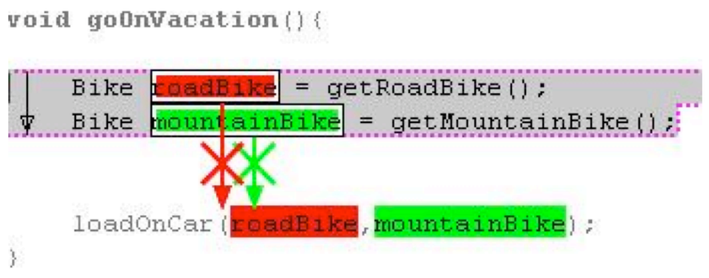
[ ] Declare thrown runtime exceptions
[ ] Generate method comment

Extract Method (Expression)
Extract Local Variable
Extract Constant
Inline Temporary Refactoring
Inline Method
Convert Anonymous to Nested
Convert Member to Top Level
Convert Local Variable to Field
Extract Superclass

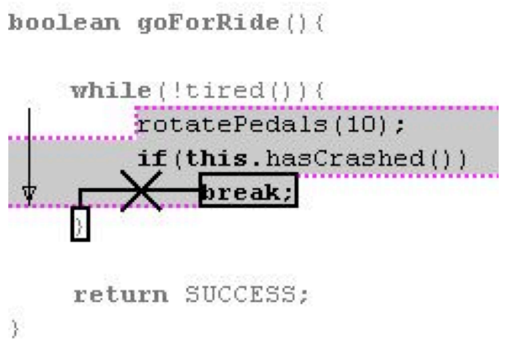
Research Improvements (Understanding Errors)



```
boolean areWheelsTrue(){

    Wheel front = bike.getFrontWheel();
    Wheel rear = bike.getRearWheel();

    boolean truedWheels = isWheelTrue(front);
    truedWheels = truedWheels || isWheelTrue(rear);

    return truedWheels;
}
```



```
void goOnVacation(){

    Bike roadBike = getRoadBike();
    Bike mountainBike = getMountainBike();

    loadOnCar(roadBike,mountainBike);
}
```



```
boolean goForRide(){

    while(!tired()){
        rotatePedals(10);
        if(this.hasCrashed())
            break;
    }

    return SUCCESS;
}
```

Summary

- Refactoring tools are a *good thing*
  - but only if they are used
- Programmers don’t use refactoring tools much because they don’t fit with how they work
- Researchers and industry are attempting to make better refactoring tools!
  - To be successful, we must pay attention to how programmers work

References

M. Fowler, K. Beck, J. Brant, W. Opdyke and D. Roberts, Refactoring: Improving the Design of Existing Code: Addison-Wesley Professional, 1999.

J. Shore. “Design Debt.” Software Profitability Newsletter. February 2004.

G. Murphy, M. Kersten and L. Findlater, "How Are Java Software Developers Using the Eclipse IDE?," IEEE Software, 2006.

E. Van Emden and L. Moonen, "Java Quality Assurance by Detecting Code Smells," presented at WCRE '02: Proceedings of the Ninth Working Conference on Reverse Engineering, 2002.

S. Hayashi, M. Saeki, and M. Kurihara, "Supporting Refactoring Activities Using Histories of Program Modification," IEICE Transactions on Information and Systems, 2006.

M. Pizka. “Straightening Spaghetti Code with Refactoring.” In Proc. of the Int. Conf. on Software Engineering Research and Practice - SERP, pages 846- 852, Las Vegas, NV, June 2004. CSREA Press.

F. Bourquin and R. Keller, “High-impact refactoring based on architecture violations,” Proceedings of CSMR 2007.

M, Mäntylä and C. Lassenius, "Drivers for software refactoring decisions," presented at ISESE '06: Proceedings of the 2006 ACM/IEEE international symposium on International symposium on empirical software engineering, 2006.

P. Weißgerber and S. Diehl, "Are refactorings less error-prone than other changes?," presented at MSR '06:Proceedings of the 2006 international workshop on Mining software repositories, 2006.

X-Develop. Omnicore Software. 2007.