Tópicos Avançados em E. S.
Parte 2

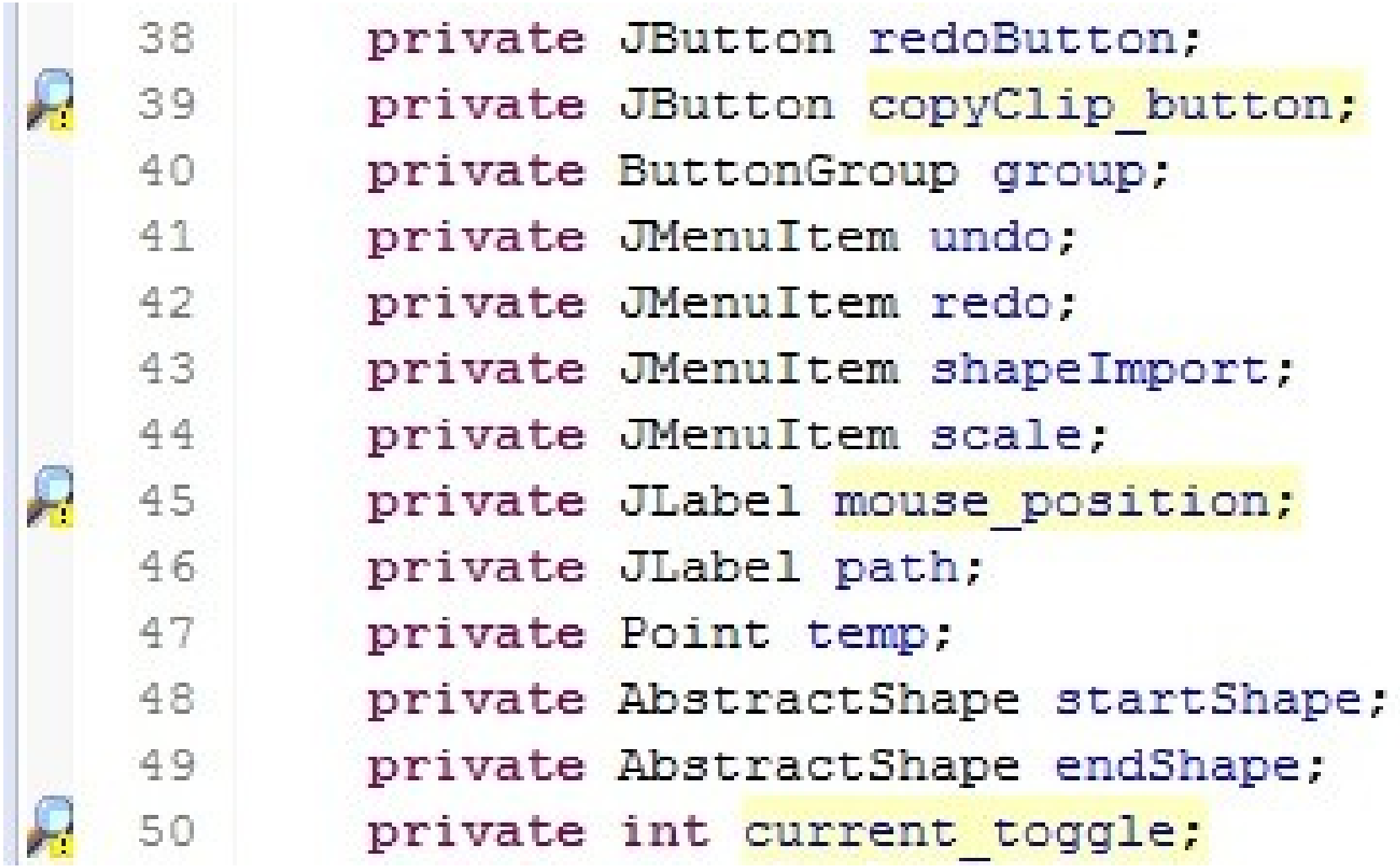
- Testes
- *Bad Smell framework*
- Métricas
- Refactoring techniques

Felipe Gusmão
Jean Ferreira

CheckStyle antes

```
private JButton copyClipButton;
private ButtonGroup group;
private JMenuItem undo;
private JMenuItem redo;
private JMenuItem shapeImport;
private JMenuItem scale;
private JLabel mousePosition;
private JLabel path;
private Point temp;
private AbstractShape startShape;
private AbstractShape endShape;
private int currentToggle;
private int startId;
private JFileChooser fc = new JFileChooser();
private boolean empty = true;
private boolean inside = false;
```

CheckStyle depois

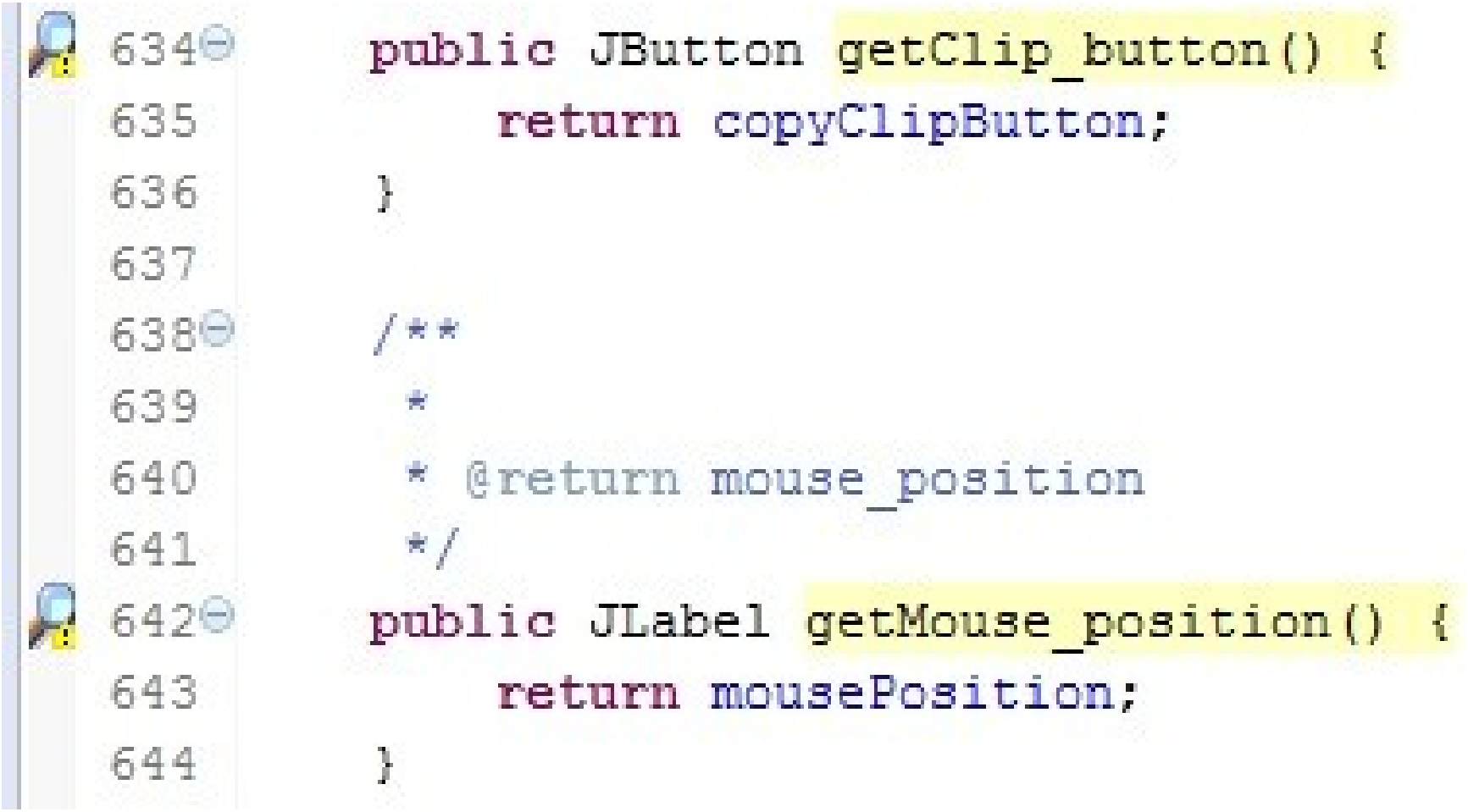


```
38    private JButton redoButton;
39    private JButton copyClip_button;
40    private ButtonGroup group;
41    private JMenuItem undo;
42    private JMenuItem redo;
43    private JMenuItem shapeImport;
44    private JMenuItem scale;
45    private JLabel mouse_position;
46    private JLabel path;
47    private Point temp;
48    private AbstractShape startShape;
49    private AbstractShape endShape;
50    private int current_toggle;
```

CheckStyle antes

```
629
630     public JButton getClipButton() {
631         return copyClipButton;
632     }
633
634     public JLabel getMousePosition(){
635         return this.mousePosition;
636     }
```

CheckStyle depois



CheckStyle antes

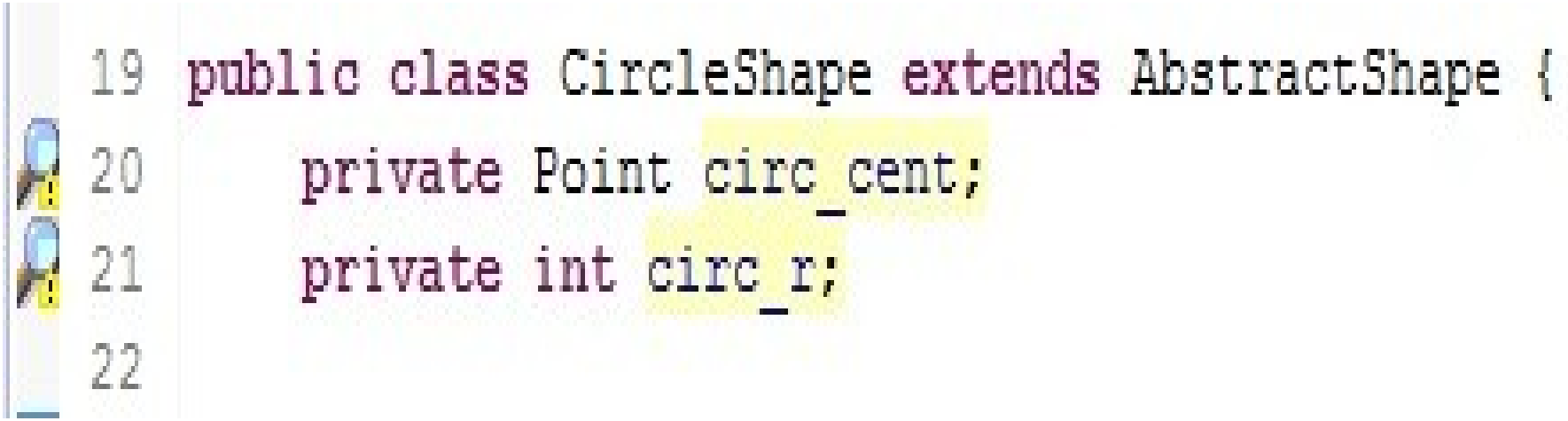
```
 45     private JLabel mousePositionImgMap;

634
635     public JLabel getMousePositionImgMap(){
636         return this.mousePositionImgMap;
637     }
```

CheckStyle antes

```
19 public class CircleShape extends AbstractShape {
20     private Point circCent;
21     private int circR;
```

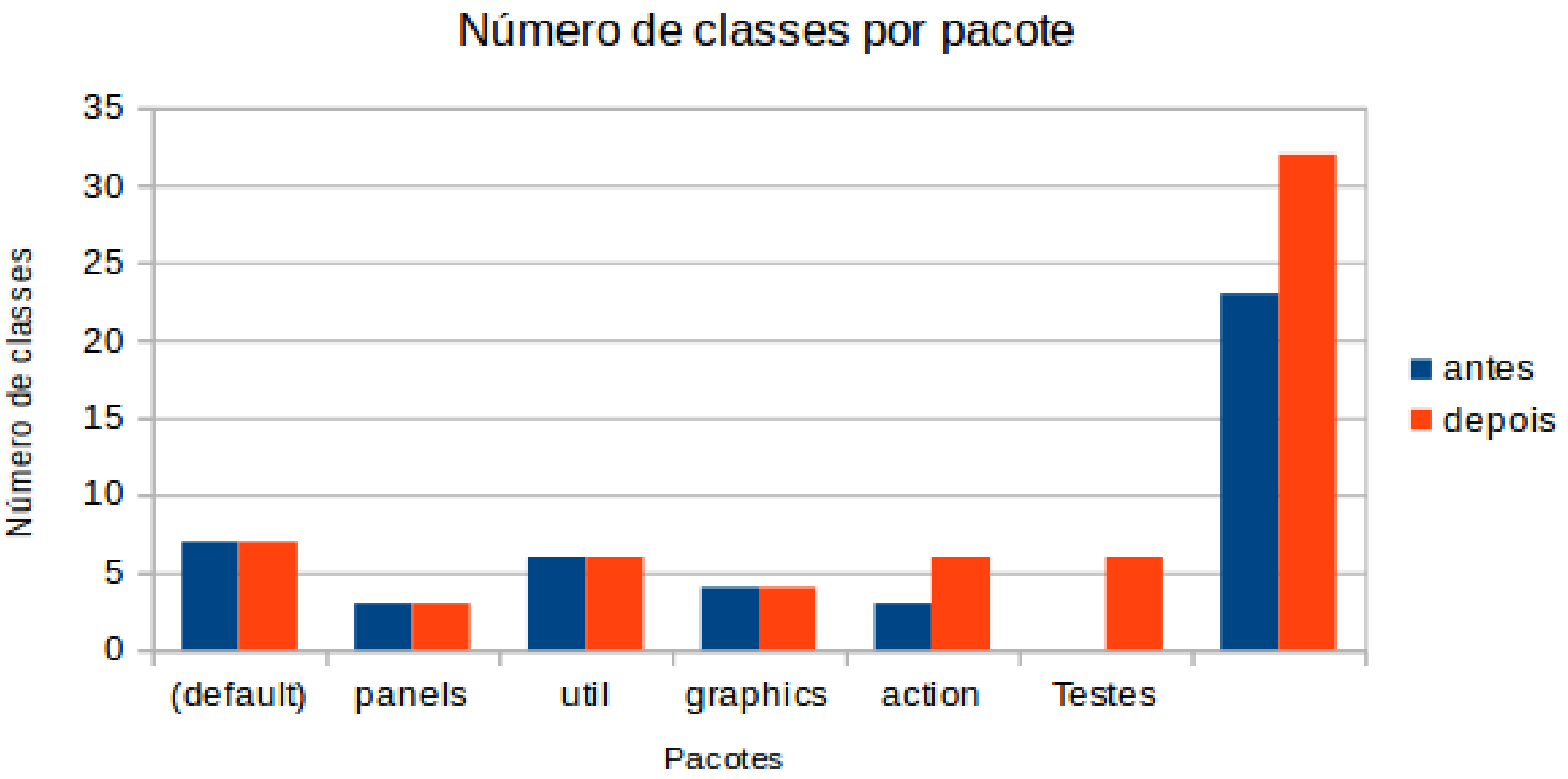
CheckStyle depois



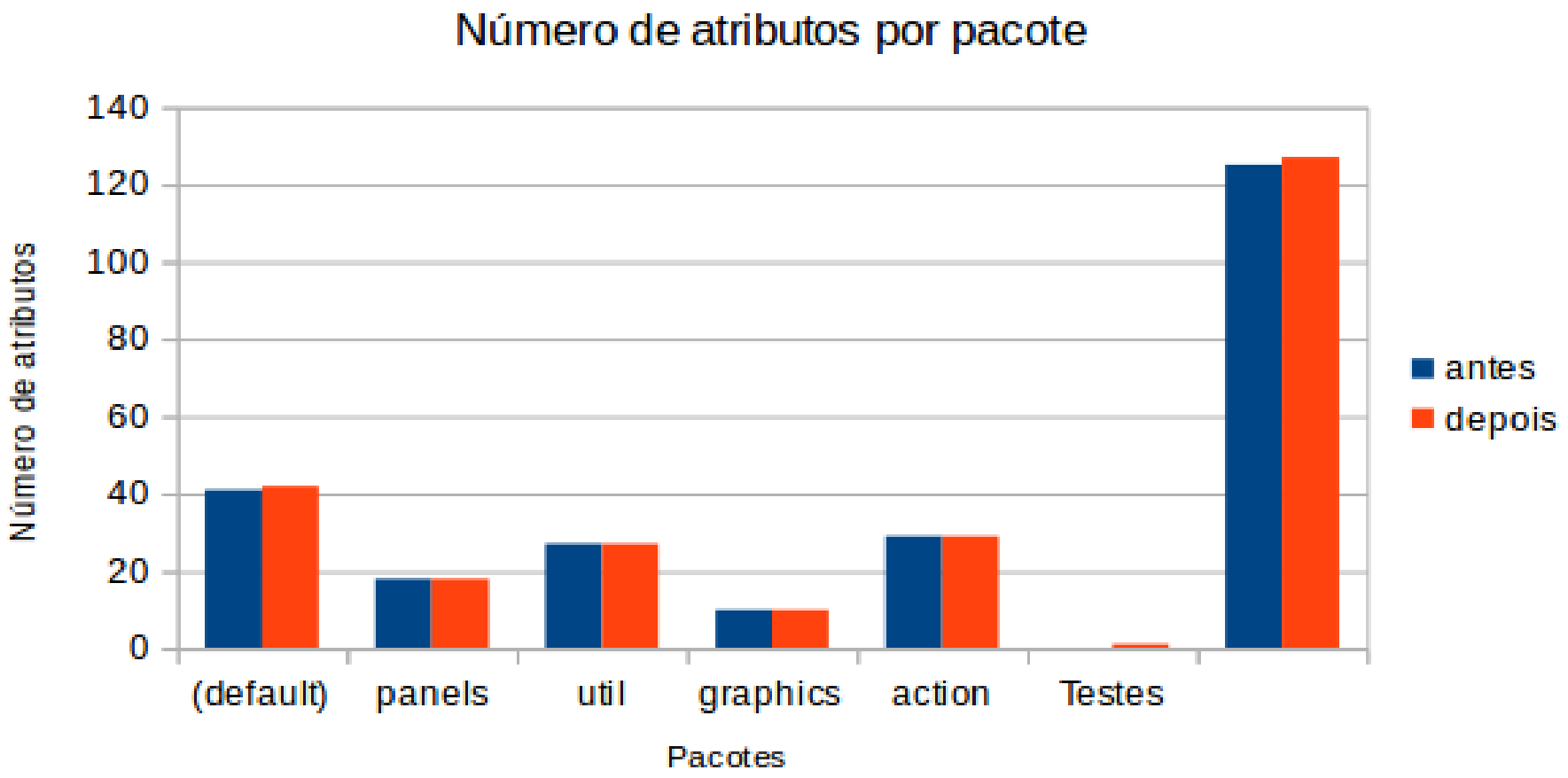
Testes

- Ver código

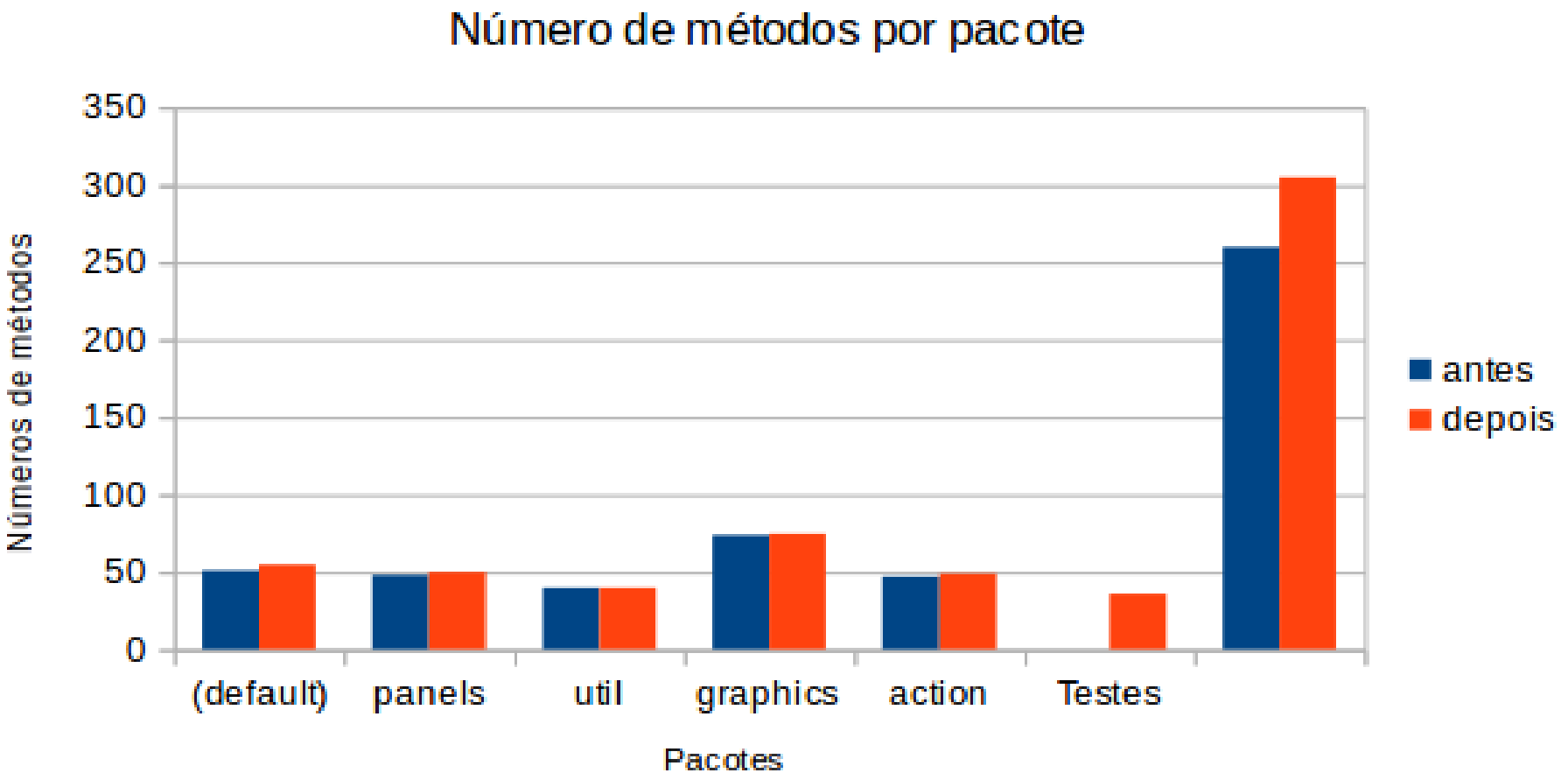
Métricas


Número de classes por pacote
Número de classes
35
30
25
20
15
10
5
0
(default)
panels
util
graphics
action
Testes
Pacotes
antes
depois

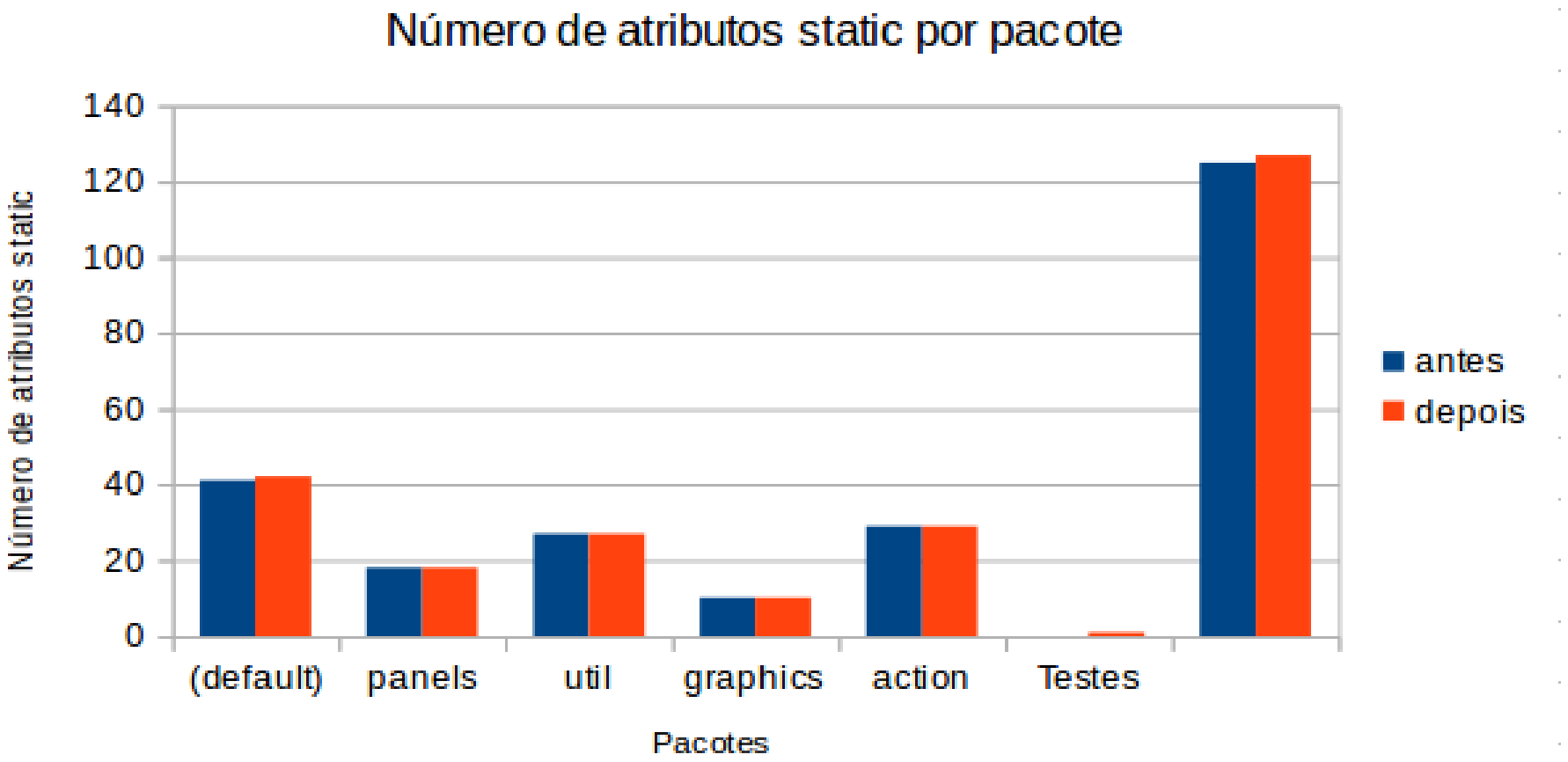
Métricas


Número de atributos por pacote
Número de atributos
140
120
100
80
60
40
20
0
(default)
panels
util
graphics
action
Testes
Pacotes
antes
depois

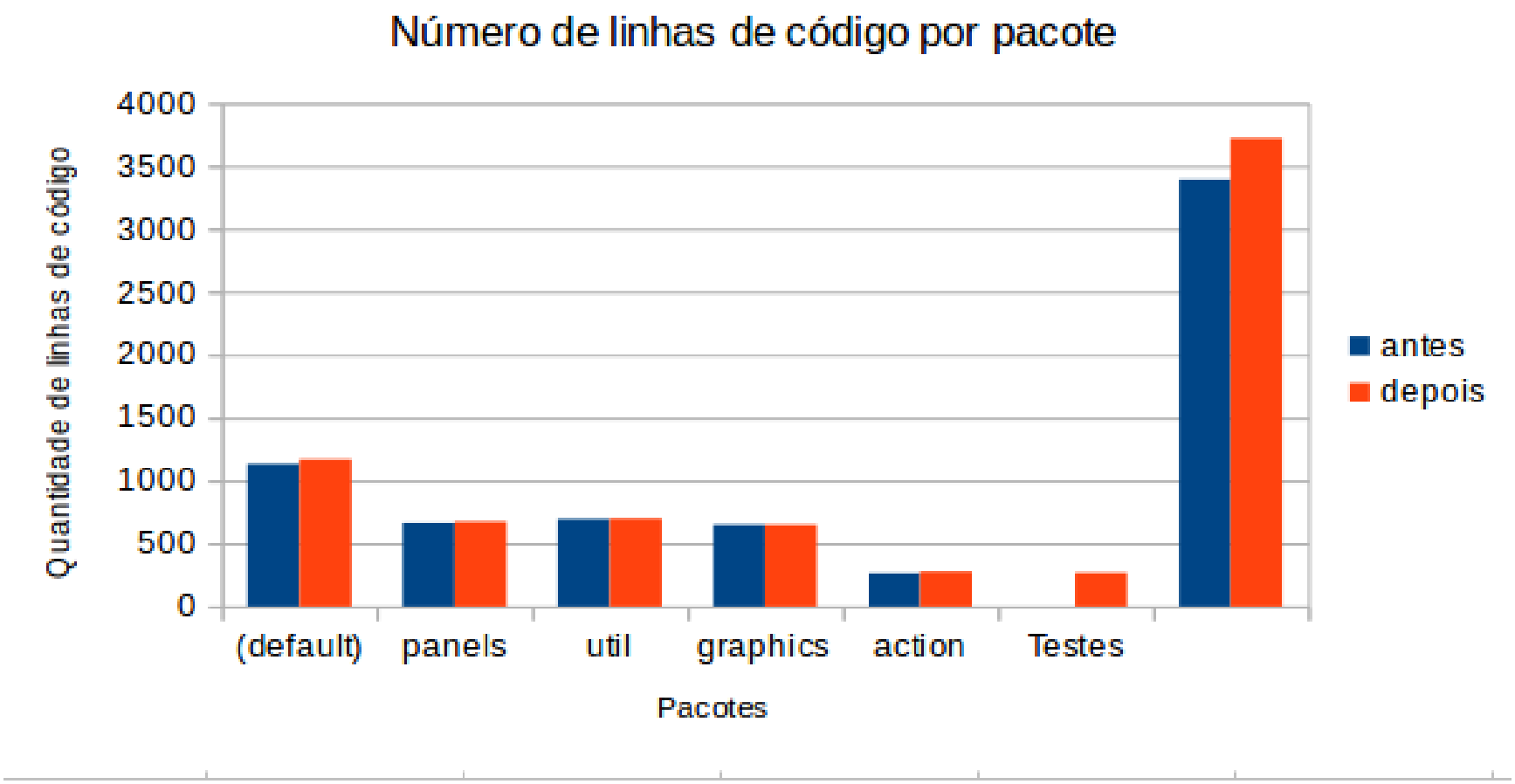
Métricas


Número de métodos por pacote
Números de métodos
350
300
250
200
150
100
50
0
(default)
panels
util
graphics
action
Testes
Pacotes
antes
depois

Métricas


Número de atributos static por pacote
Número de atributos static
140
120
100
80
60
40
20
0
(default)
panels
util
graphics
action
Testes
Pacotes
antes
depois

Métricas


Número de linhas de código por pacote
Quantidade de linhas de código
4000
3500
3000
2500
2000
1500
1000
500
0
(default)
panels
util
graphics
action
Testes
Pacotes
antes
depois

Métricas – Profundidade da árvore de herança

- Antes da refatoração: 6*
- Depois da refatoração: 6*

*Obs.: o *framework* utilizado contabiliza desde a superclasse *Object*.

Métricas – Quantidade de subclasses

- Antes da refatoração: 9
- Depois da refatoração: 3

Refactoring Techniques

- CheckStyle (Bad smells)
- Aplicação de Padrões de Projeto
- Encapsulate downcast
- Raplace Temp with Query
- Encapsulate Fields
- Extract Method
- Extract Variable

Encapsulate Downcast

```
/**
 * @return the editedShape
 */
public AbstractShape getEditedShape() {
    return editedShape;
}
```

```
33
34      /**
35       * @return the editedShape
36       */
37      public AbstractShape getEditedShape() {
38          switch(editedShape.getType()){
39              case AbstractShape.TYPE_RECT:
40                  return returnRectangleEditedShape();
41              case AbstractShape.TYPE_CIRC:
42                  return returnCircleEditedShape();
43          }
44          return editedShape;
45      }
46
47      public RectangleShape returnRectangleEditedShape() {
48          return (RectangleShape) editedShape;
49      }
50
51      public CircleShape returnCircleEditedShape() {
52          return (CircleShape) editedShape;
53      }
54
```

Replace Temp with Query

```
/**
 * @see imagemap.graphics.AbstractShape#contains(int, int)
 */
@Override
public boolean contains(int x, int y) {
    int x_tmp = (int) circCent.getX();
    int y_tmp = (int) circCent.getY();
    double dist = Math.sqrt((Math.pow((x - x_tmp), 2) + Math.pow((y - y_tmp), 2)));
    if (dist < circR) {
        return true;
    } else {
        return false;
    }
}
```

Replace Temp with Query cont.

```
53    /**
54     * @see imagemap.graphics.AbstractShape#contains(int, int)
55     */
56    @Override
57    public boolean contains(int x, int y) {
58        double dist = calculaDistancia(x, (int) circCent.getX(), y, (int) circCent.getY());
59        if (dist < circR) {
60            return true;
61        } else {
62            return false;
63        }
64    }
65
66    private double calculaDistancia(int x, int x_tmp, int y, int y_tmp) {
67        return Math.sqrt((Math.pow((x - x_tmp), 2) + Math.pow((y - y_tmp), 2)));
68    }
69
```

Replace Temp with Query cont.

Reutilização da função criada

```
121    /**
122     * @see imagemap.graphics.AbstractShape#movePoint(java.awt.Point, int, int)
123     */
124    @Override
125    public void movePoint(Point p, int xdir, int ydir) {
126        double xcomponent = p.getX() - circCent.getX() + xdir;
127        double ycomponent = p.getY() - circCent.getY() + ydir;
128        circR = (int) calculaDistancia((int) xcomponent, (int) ycomponent);
129        if (circR < 4) {
130            circR = 4;
131        }
132    }
133
```

Encapsulate Fields

```
34      private AbstractShape tempShape;
35      private AbstractShape draggedShape;
36      public AbstractShape currentShape;
37      private String imagePath;
38      private String savePath;
```

Encapsulate Fields cont.

```
36      private AbstractShape currentShape;
```

```
/**
 * @param currentShape
 *            the currentShape to set
 */
public void setCurrentShape(AbstractShape currentShape) {
    this.currentShape = currentShape;
    repaint();
}
```

```
/**
 * @return the currentShape
 */
public AbstractShape getCurrentShape() {
    return currentShape;
}
```

Extract Method (class HelpFrame)

```
// assembling
String r = "To draw a rectangle click on beginning edge and drag to opposite edge of rectangle.\nReleasing the m
setUpTextArea(rect, "Rectangle", r);
String c = "To draw a circle click on the center of your desired circle and drag the radius.\nReleasing the mous
setUpTextArea(circ, "Circle", c);
String p = "TO draw a polygon click every corner your polygon should have in the intended order.\nPressing escap
setUpTextArea(poly, "Polygon", p);
general.add(rect);
general.add(circ);
general.add(poly);
tabs.add(general, "General");
general.setBorder(new EmptyBorder(5, 5, 5, 5));
general.setBackground(new Color(232, 232, 232));

String m = "To move a shape switch to mouse mode and move the cursor inside of a shape.\nDrag to new position ar
setUpTextArea(mov, "Moving", m);
String re = "To resize a shape siwtch to mouse mode and move the cursor to the marked corner you want to change.
setUpTextArea(res, "Resizing", re);
String b = "Undo: Ctrl + Z or via menubar.\nRedo: Ctrl + Y or via menubar";
setUpTextArea(bf, "Undo/Redo", b);
editing.add(mov);
editing.add(res);
editing.add(bf);
tabs.add(editing, "Editing");
editing.setBorder(new EmptyBorder(5, 5, 5, 5));
editing.setBackground(new Color(232, 232, 232));
```

Extract Method cont. (class HelpFrame)

```
    assembling(general, rect, circ, poly, editing, res, mov, bf, extras, scale, imp, about);

    finalizing();
}

private void finalizing() {
    Dimension d = Toolkit.getDefaultToolkit().getScreenSize();
    this.add(tabs);
    this.setTitle("Help");
    this.setSize((int) d.getWidth() / 3, (int) d.getHeight() / 3);
    this.setLocation((d.width - this.getSize().width) / 2, (d.height - this.getSize().height) / 2);
}

private void assembling(JPanel general, JTextArea rect, JTextArea circ, JTextArea poly, JPanel editing,
        JTextArea res, JTextArea mov, JTextArea bf, JPanel extras, JTextArea scale, JTextArea imp, JPanel about) {
    String r = "To draw a rectangle click on beginning edge and drag to opposite edge of rectangle.\nReleasing the m
```

Extract Variable (class RectangleShape)

```
75          tmp_rect.add(new Rectangle(rect.x - 3, rect.y - 3, 6, 6));
76          tmp_rect.add(new Rectangle(rect.x + rect.width - 3, rect.y - 2, 6, 6));
77          tmp_rect.add(new Rectangle(rect.x - 3, rect.y + rect.height - 2, 6, 6));
78          tmp_rect.add(new Rectangle(rect.x + rect.width - 3, rect.y + rect.height - 3, 6, 6));
```

```
72      @Override
73      public boolean cornerContains(Point p) {
74          Vector<Rectangle> tmp_rect = new Vector<Rectangle>();
75          final Rectangle[] cantos = new Rectangle[] { new Rectangle(rect.x - 3, rect.y - 3, 6, 6),
76                  new Rectangle(rect.x + rect.width - 3, rect.y - 2, 6, 6),
77                  new Rectangle(rect.x - 3, rect.y + rect.height - 2, 6, 6),
78                  new Rectangle(rect.x + rect.width - 3, rect.y + rect.height - 3, 6, 6) };
79
80          for (Rectangle r : cantos) {
81              tmp_rect.add(r);
82              if (r.contains(p))
83                  return true;
84          }
85          return false;
86      }
```