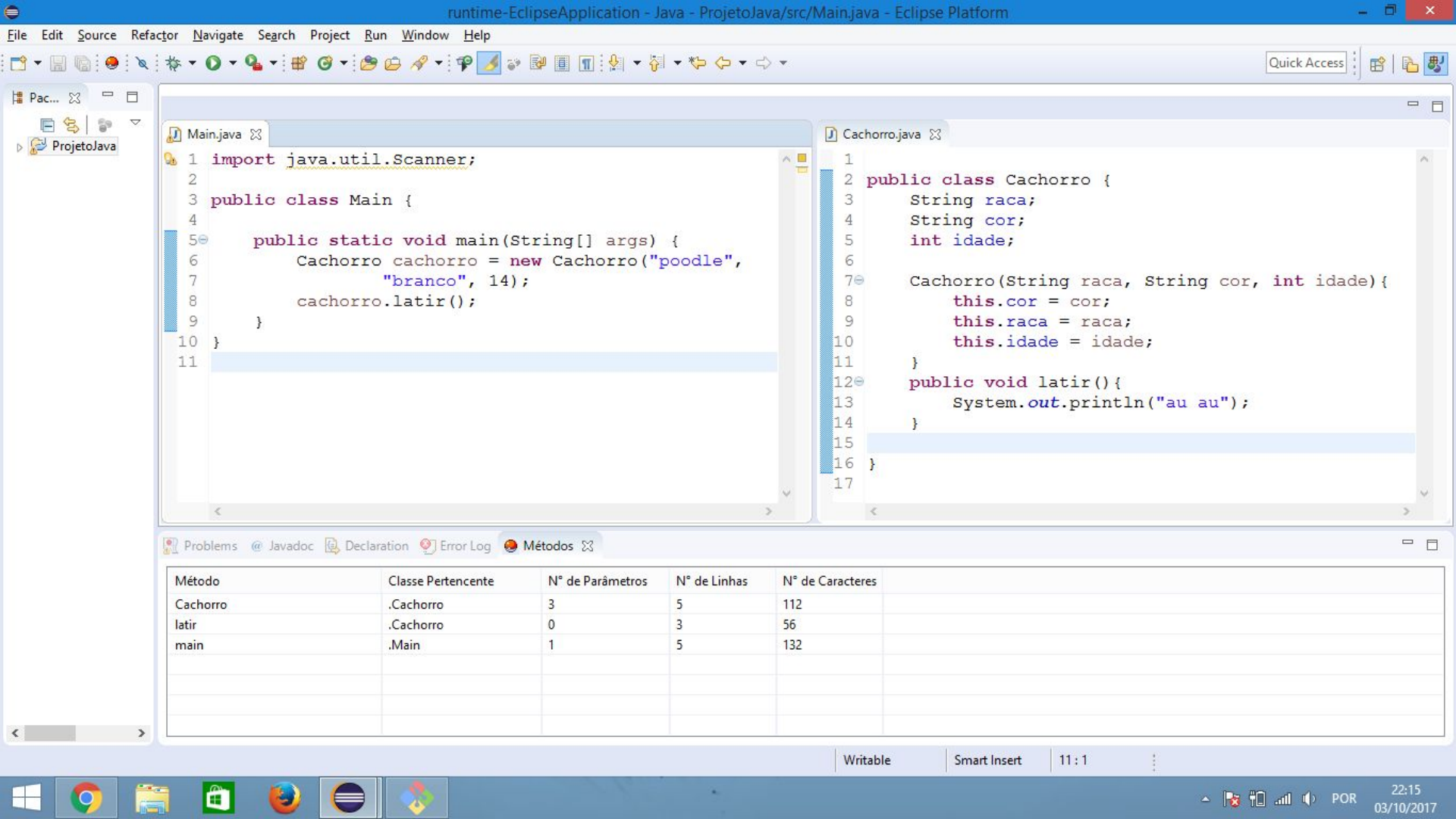
Análise estática e dinâmica

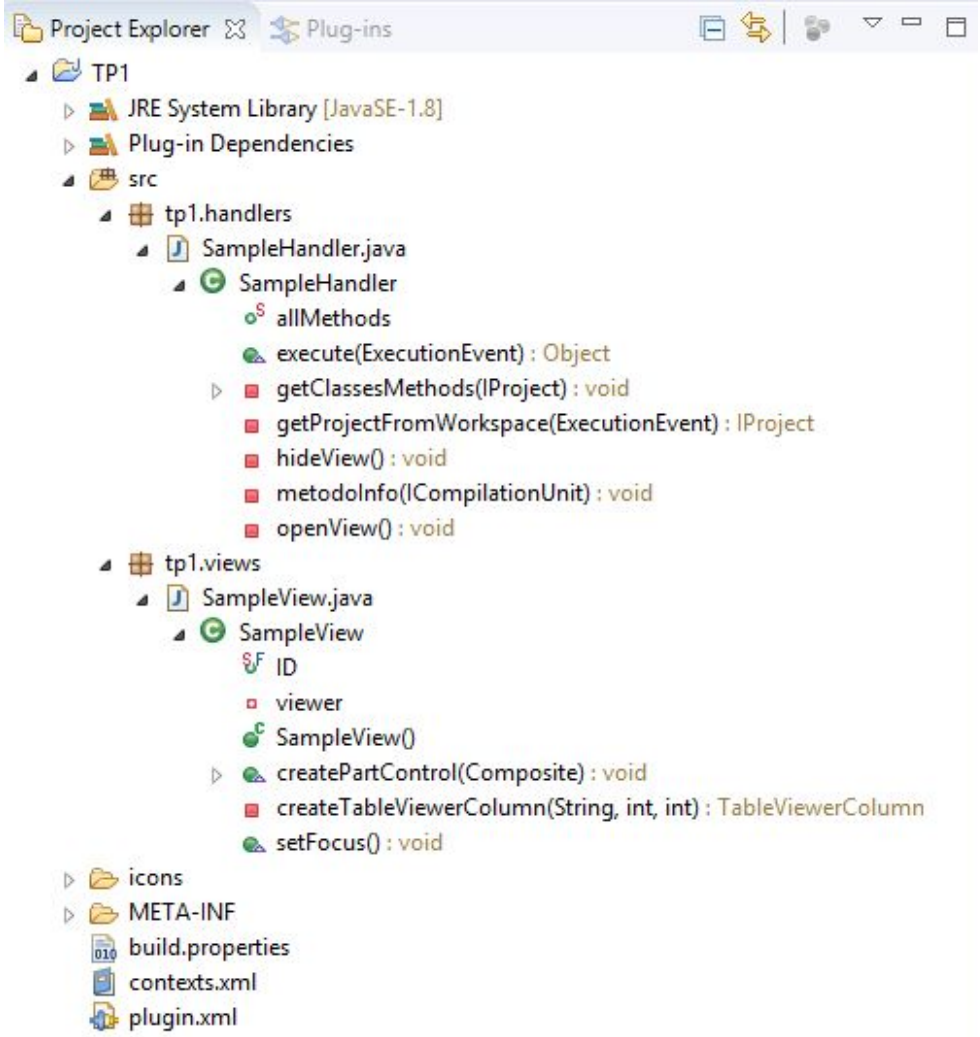
Trabalho prático 01

Christian Marlon Souza Couto
Luana Almeida Martins

Decisões de projeto

- Análise escolhida
- O que motivou a escolha da análise
- Como a análise foi feita


runtime-EclipseApplication - Java - ProjetoJava/src/Main.java - Eclipse Platform
File Edit Source Refactor Navigate Search Project Run Window Help
Quick Access
Pac...
ProjetoJava
Main.java
1 import java.util.Scanner;
2
3 public class Main {
4
5 public static void main(String[] args) {
6 Cachorro cachorro = new Cachorro("poodle",
7 "branco", 14);
8 cachorro.latir();
9 }
10 }
11
Cachorro.java
1
2 public class Cachorro {
3 String raca;
4 String cor;
5 int idade;
6
7 Cachorro(String raca, String cor, int idade){
8 this.cor = cor;
9 this.raca = raca;
10 this.idade = idade;
11 }
12 public void latir(){
13 System.out.println("au au");
14 }
15
16 }
17
Problems Javadoc Declaration Error Log Métodos
Método | Classe Pertencente | N° de Parâmetros | N° de Linhas | N° de Caracteres
Cachorro | .Cachorro | 3 | 5 | 112
latir | .Cachorro | 0 | 3 | 56
main | .Main | 1 | 5 | 132
Writable
Smart Insert
11 : 1
POR
22:15
03/10/2017


Project Explorer
Plug-ins
TP1
JRE System Library [JavaSE-1.8]
Plug-in Dependencies
src
tp1.handlers
SampleHandler.java
SampleHandler
allMethods
execute(ExecutionEvent) : Object
getClassesMethods(IProject) : void
getProjectFromWorkspace(ExecutionEvent) : IProject
hideView() : void
metodoInfo(ICompilationUnit) : void
openView() : void
tp1.views
SampleView.java
SampleView
ID
viewer
SampleView()
createPartControl(Composite) : void
createTableViewerColumn(String, int, int) : TableViewerColumn
setFocus() : void
icons
META-INF
build.properties
contexts.xml
plugin.xml

```
30 public class SampleHandler extends AbstractHandler {
31
32     public static ArrayList<IMethod> allMethods;
33
34     @Override
35     public Object execute(ExecutionEvent event) throws ExecutionException {
36
37         allMethods = new ArrayList<IMethod>();
38
39         hideView();
40
41         IProject iProject = getProjectFromWorkspace(event);
42
43         try {
44             getClassesMethods(iProject);
45         } catch (CoreException e) {
46             e.printStackTrace();
47         }
48
49         openView();
50
51
52         allMethods = null;
53         return null;
54
55     }
56
57     private void getClassesMethods(final IProject project) throws CoreException {
58         project.accept(new IResourceVisitor() {
59
```

```
56
57    private void getClassesMethods(final IProject project) throws CoreException {
58        project.accept(new IResourceVisitor() {
59
60            @Override
61            public boolean visit(IResource resource) throws JavaModelException {
62                if (resource instanceof IFile && resource.getName().endsWith(".java")) {
63                    ICompilationUnit unit = ((ICompilationUnit) JavaCore.create((IFile) resource));
64                    try {
65                        metodoInfo(unit);
66                    } catch (BadLocationException e) {
67
68                        e.printStackTrace();
69                    }
70                }
71                return true;
72            }
73        });
74    }
75
76    private void metodoInfo(ICompilationUnit unit) throws JavaModelException, BadLocationException {
77        IType[] allTypes = unit.getAllTypes();
78        for (IType type : allTypes) {
79            IMethod[] methods = type.getMethods();
80            for (IMethod method : methods) {
81                allMethods.add(method);
82            }
83        }
84    }
85
```

```
27
28    public void createPartControl(Composite parent) {
29
30        GridLayout layout = new GridLayout(2, false);
31        parent.setLayout(layout);
32        viewer = new TableViewer(parent, SWT.MULTI | SWT.H_SCROLL | SWT.V_SCROLL | SWT.FULL_SELECTION | SWT.BORDER)
33
34        String[] titles = { "Método", "Classe Pertencente", "N° de Parâmetros", "N° de Linhas", "N° de Caracteres"
35        int[] bounds = { 200, 150, 120, 100, 100 };
36
37        // Primeira coluna é para o nome do método
38        TableViewerColumn col = createTableViewerColumn(titles[0], bounds[0], 0);
39        col.setLabelProvider(new ColumnLabelProvider() {
40            @Override
41            public String getText(Object element) {
42                IMethod m = (IMethod) element;
43                return m.getElementName();
44            }
45        });
46
47        // Segunda coluna é para nome da classe a qual o método pertence
48        col = createTableViewerColumn(titles[1], bounds[1], 1);
49        col.setLabelProvider(new ColumnLabelProvider() {
50            @Override
51            public String getText(Object element) {
52                IMethod m = (IMethod) element;
53                return m.getCompilationUnit().getParent().getElementName() + "."
54                        + m.getDeclaringType().getElementName();
55            }
56        });
```

```
57
58          // Terceira coluna é o número de parametros do método
59          col = createTableViewerColumn(titles[2], bounds[2], 2);
60          col.setLabelProvider(new ColumnLabelProvider() {
61              @Override
62              public String getText(Object element) {
63                  IMethod m = (IMethod) element;
64                  return Integer.toString(m.getNumberOfParameters());
65              }
66          });
67
68          // Quarta coluna é o número de linhas do método
69          col = createTableViewerColumn(titles[3], bounds[3], 3);
70          col.setLabelProvider(new ColumnLabelProvider() {
71              @Override
72              public String getText(Object element) {
73                  IMethod m = null;
74                  Document doc = null;
75                  try {
76                      m = (IMethod) element;
77                      doc = new Document(m.getSource());
78                  } catch (JavaModelException e) {
79                      e.printStackTrace();
80                  }
81                  return Integer.toString(doc.getNumberOfLines());
82              }
83          });
84
85          // Quinta coluna é o número de caracteres do método
86          col = createTableViewerColumn(titles[4], bounds[4], 4);
```

```
84
85          // Quinta coluna é o número de caracteres do método
86          col = createTableViewerColumn(titles[4], bounds[4], 4);
87          col.setLabelProvider(new ColumnLabelProvider() {
88              @Override
89              public String getText(Object element) {
90                  IMethod m = null;
91                  Document doc;
92                  int numChars = 0;
93                  try {
94                      m = (IMethod) element;
95                      doc = new Document(m.getSource());
96
97                      for (int i = 0; i < doc.getNumberOfLines(); i++) {
98                          numChars += doc.getLineLength(i);
99                      }
100
101                 } catch (JavaModelException e) {
102                     // TODO Auto-generated catch block
103                     e.printStackTrace();
104                 } catch (BadLocationException e) {
105                     // TODO Auto-generated catch block
106                     e.printStackTrace();
107                 }
108
109                 return Integer.toString(numChars);
110             }
111         });
112
113         viewer.refresh();
```

Obrigado!