

# Padrões de projeto estruturais

Saulo Medeiros de Araujo

@sma

[sma@cesar.school](mailto:sma@cesar.school)

# Padrões estruturais

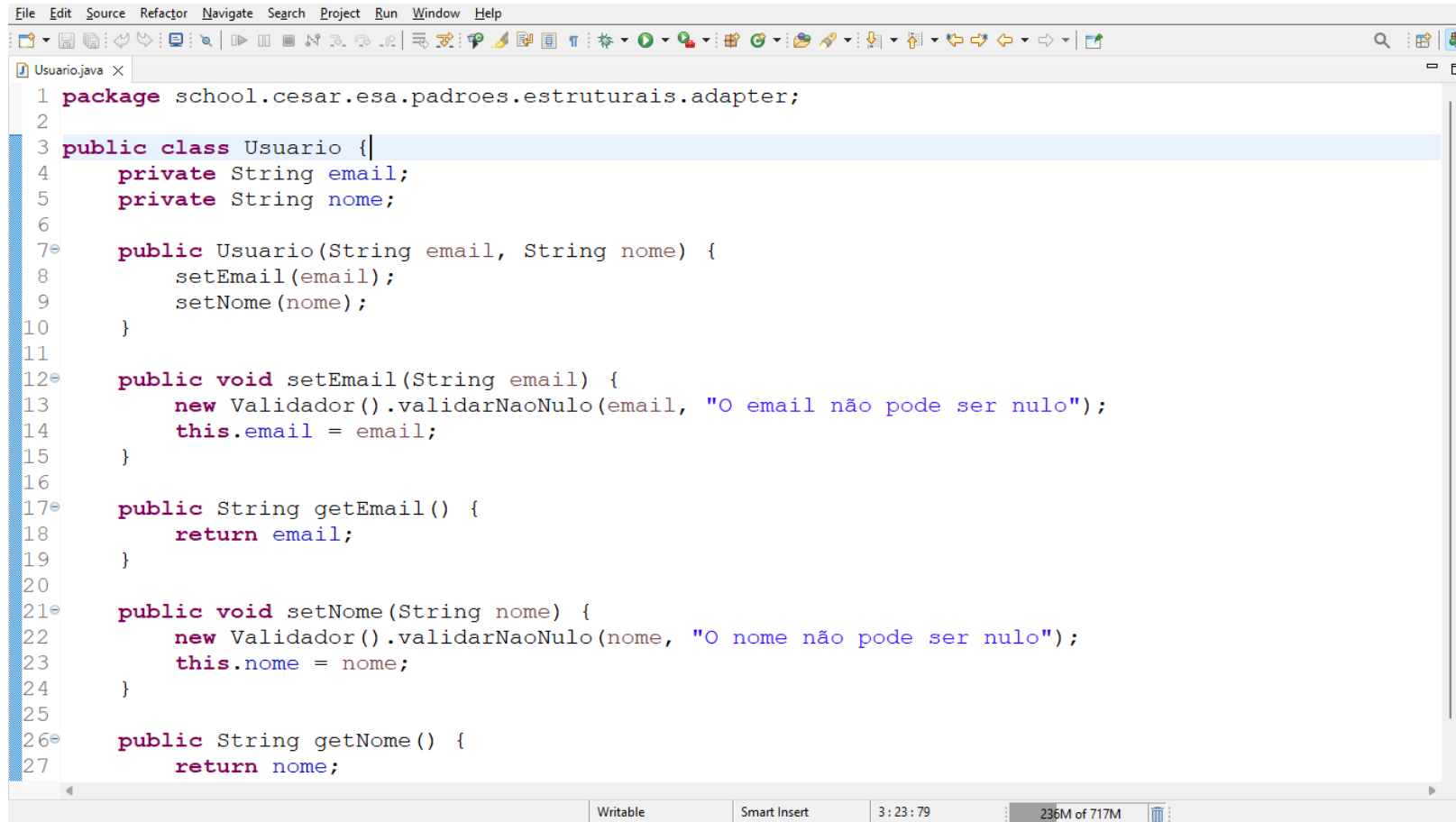
- Adapter
- Bridge
- Composite
- Decorator
- Facade
- Flyweight
- Proxy

Adapter

# Adaptar-se é preciso



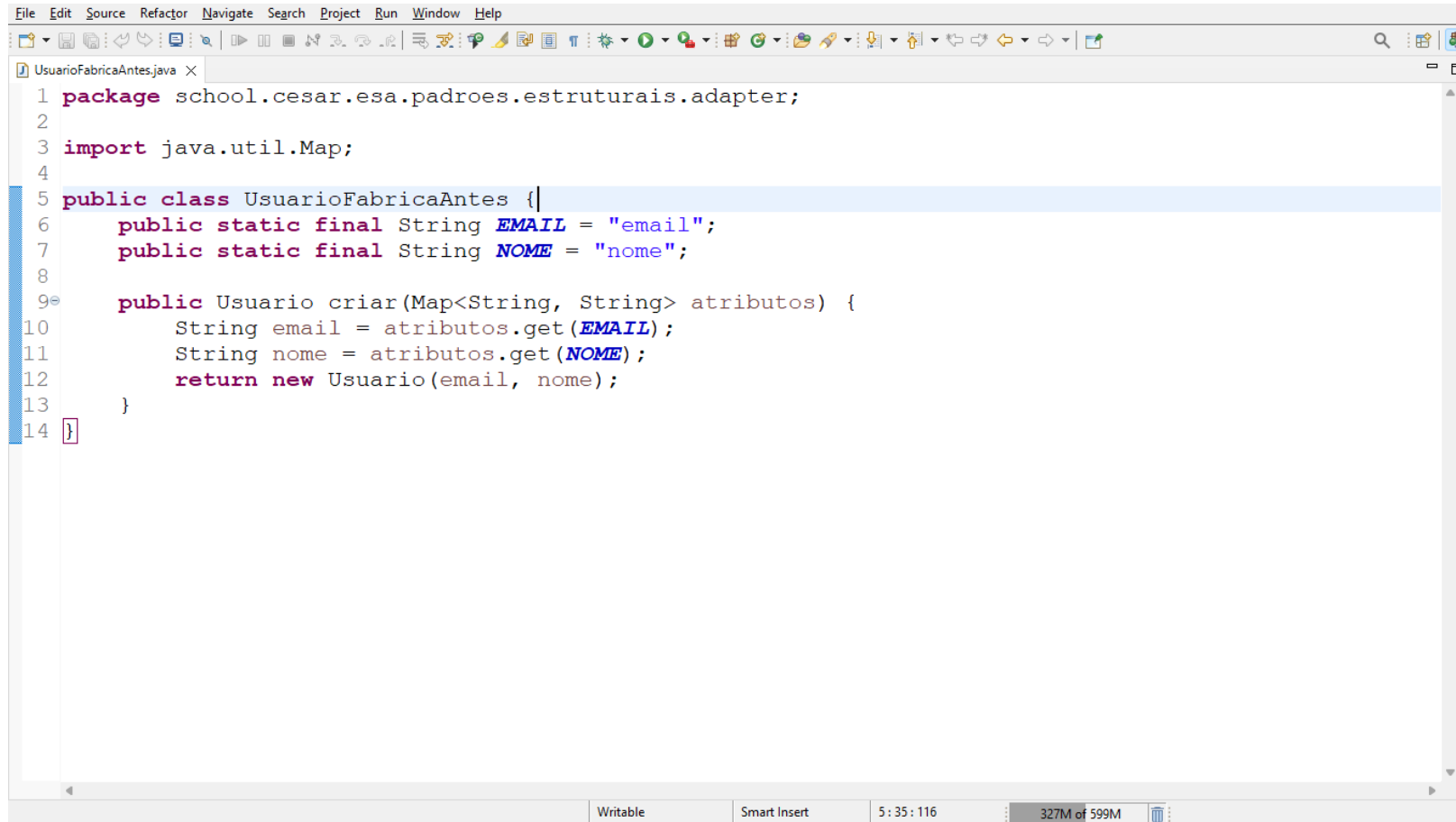
# Adapter



```
File Edit Source Refactor Navigate Search Project Run Window Help
Usuario.java x
1 package school.cesar.esa.padroes.estruturais.adapter;
2
3 public class Usuario {
4     private String email;
5     private String nome;
6
7     public Usuario(String email, String nome) {
8         setEmail(email);
9         setNome(nome);
10    }
11
12    public void setEmail(String email) {
13        new Validador().validarNaoNulo(email, "O email não pode ser nulo");
14        this.email = email;
15    }
16
17    public String getEmail() {
18        return email;
19    }
20
21    public void setNome(String nome) {
22        new Validador().validarNaoNulo(nome, "O nome não pode ser nulo");
23        this.nome = nome;
24    }
25
26    public String getNome() {
27        return nome;
28    }
29 }
```

Writable Smart Insert 3 : 23 : 79 236M of 717M

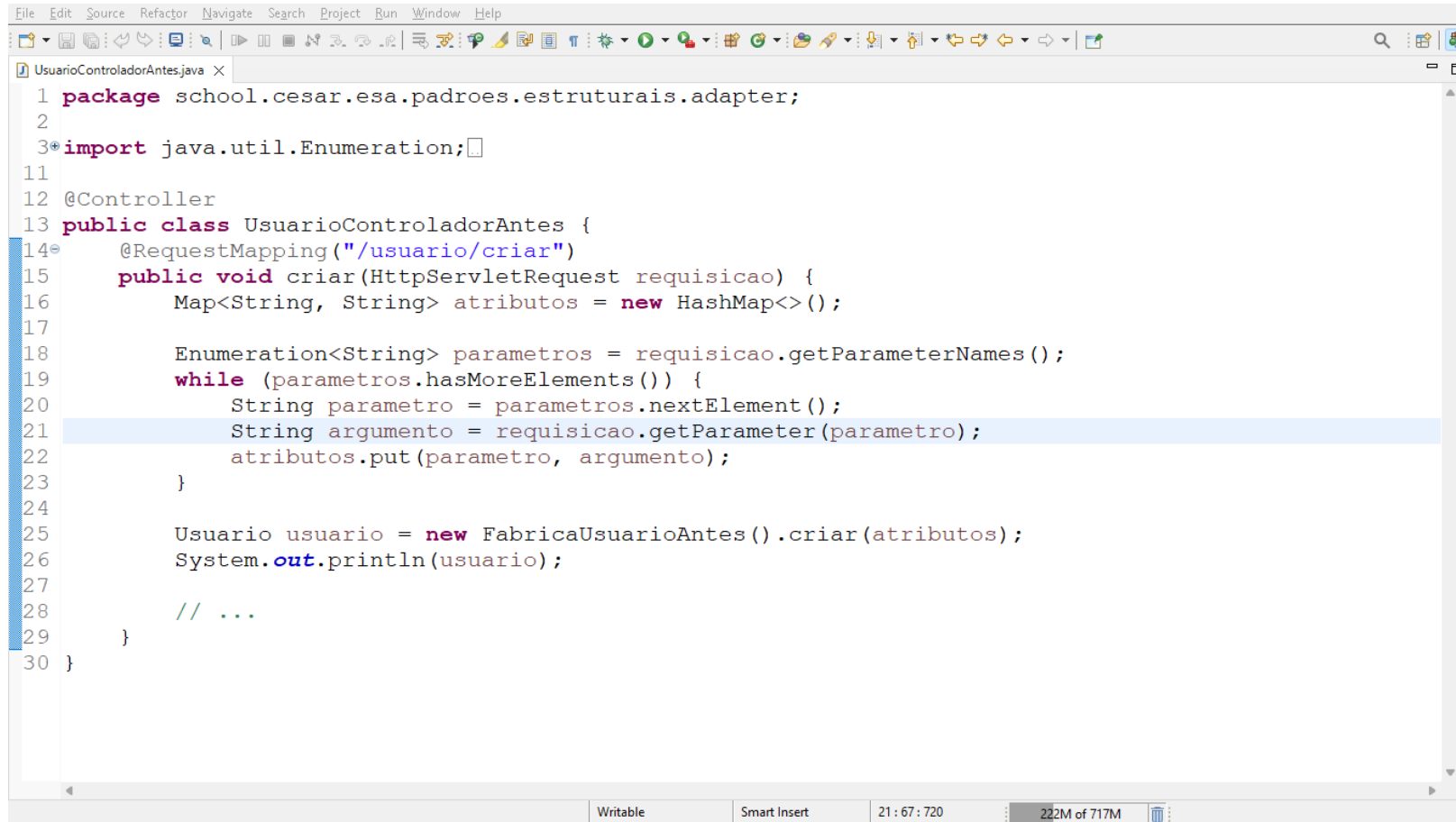
# Adapter



```
File Edit Source Refactor Navigate Search Project Run Window Help
UsuarioFabricaAntes.java
1 package school.cesar.esa.padroes.estruturais.adapter;
2
3 import java.util.Map;
4
5 public class UsuarioFabricaAntes {
6     public static final String EMAIL = "email";
7     public static final String NOME = "nome";
8
9     public Usuario criar(Map<String, String> atributos) {
10         String email = atributos.get(EMAIL);
11         String nome = atributos.get(NOME);
12         return new Usuario(email, nome);
13     }
14 }
```

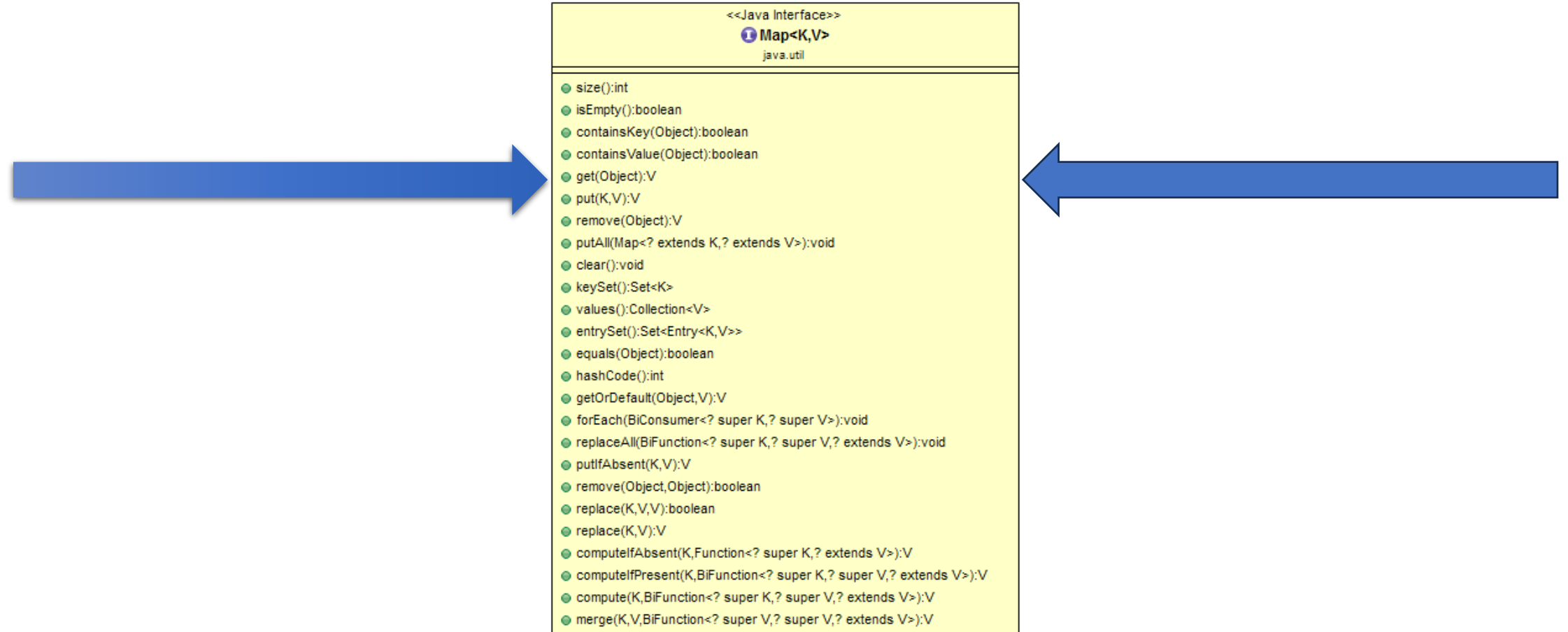
The screenshot shows an IDE window titled 'UsuarioFabricaAntes.java'. The code implements the Adapter pattern. It starts with a package declaration 'school.cesar.esa.padroes.estruturais.adapter' and an import for 'java.util.Map'. The 'UsuarioFabricaAntes' class contains two static final strings, 'EMAIL' and 'NOME', and a 'criar' method that takes a 'Map<String, String>' and returns a 'Usuario' object by looking up the email and name in the map and creating a new 'Usuario' instance with those values. The IDE interface includes a menu bar, a toolbar, and a status bar at the bottom showing 'Writable', 'Smart Insert', '5 : 35 : 116', and '327M of 599M'.

# Adapter

A screenshot of an IDE window showing a Java file named 'UsuarioControladorAntes.java'. The code implements an adapter pattern. It starts with a package declaration 'school.cesar.esa.padroes.estruturais.adapter;' and an import 'java.util.Enumeration;'. The class 'UsuarioControladorAntes' is annotated with '@Controller' and has a '@RequestMapping("/usuario/criar")' annotation. The 'criar' method takes an 'HttpServletRequest requisicao' and creates a 'HashMap' to store attributes. It then iterates through the parameter names of the request, getting each parameter value and putting it into the map. Finally, it calls 'FabricaUsuarioAntes().criar(attributes)' to create a 'Usuario' object and prints it. The code is as follows:

```
1 package school.cesar.esa.padroes.estruturais.adapter;
2
3 import java.util.Enumeration;
4
11
12 @Controller
13 public class UsuarioControladorAntes {
14     @RequestMapping("/usuario/criar")
15     public void criar(HttpServletRequest requisicao) {
16         Map<String, String> atributos = new HashMap<>();
17
18         Enumeration<String> parametros = requisicao.getParameterNames();
19         while (parametros.hasMoreElements()) {
20             String parametro = parametros.nextElement();
21             String argumento = requisicao.getParameter(parametro);
22             atributos.put(parametro, argumento);
23         }
24
25         Usuario usuario = new FabricaUsuarioAntes().criar(atributos);
26         System.out.println(usuario);
27
28         // ...
29     }
30 }
```

# Adapter

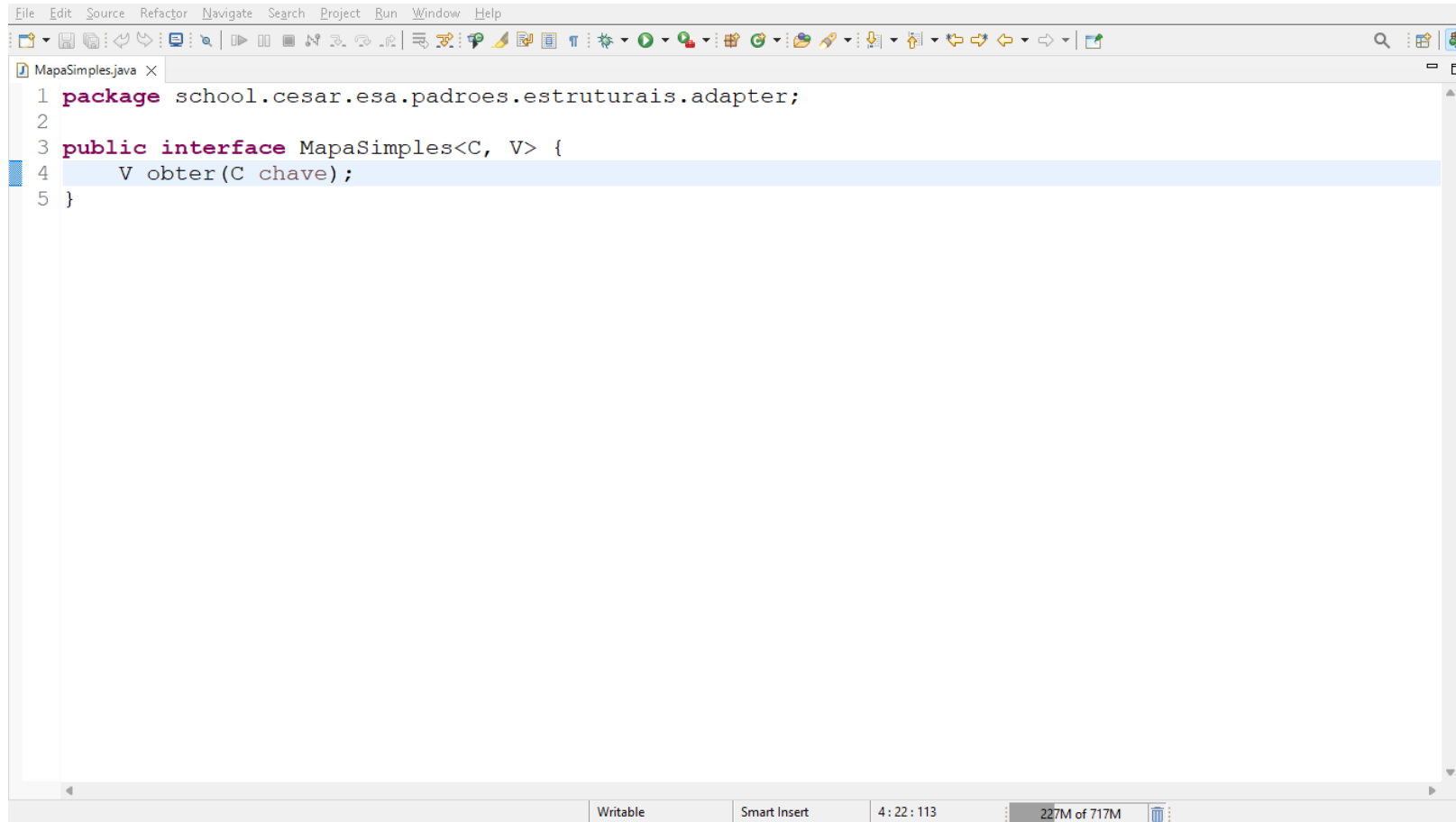




# Adaptadores



# Adapter



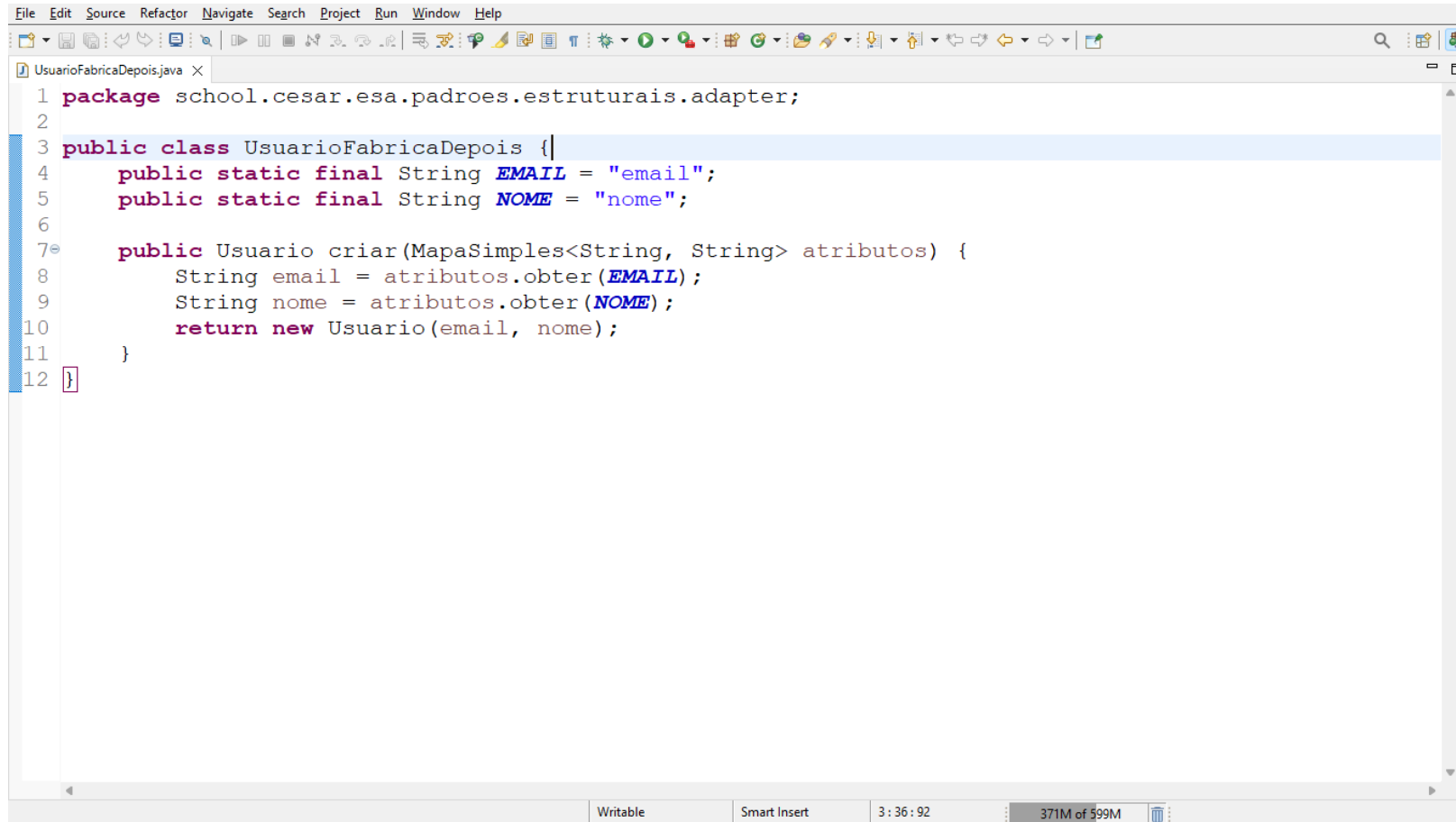
```
1 package school.cesar.esa.padroes.estruturais.adapter;  
2  
3 public interface MapaSimples<C, V> {  
4     V obter(C chave);  
5 }
```

The screenshot shows an IDE window titled 'MapaSimples.java'. The code is as follows:

```
1 package school.cesar.esa.padroes.estruturais.adapter;  
2  
3 public interface MapaSimples<C, V> {  
4     V obter(C chave);  
5 }
```

The interface definition on lines 3-5 is highlighted. The status bar at the bottom indicates 'Writable', 'Smart Insert', '4 : 22 : 113', and '227M of 717M'.

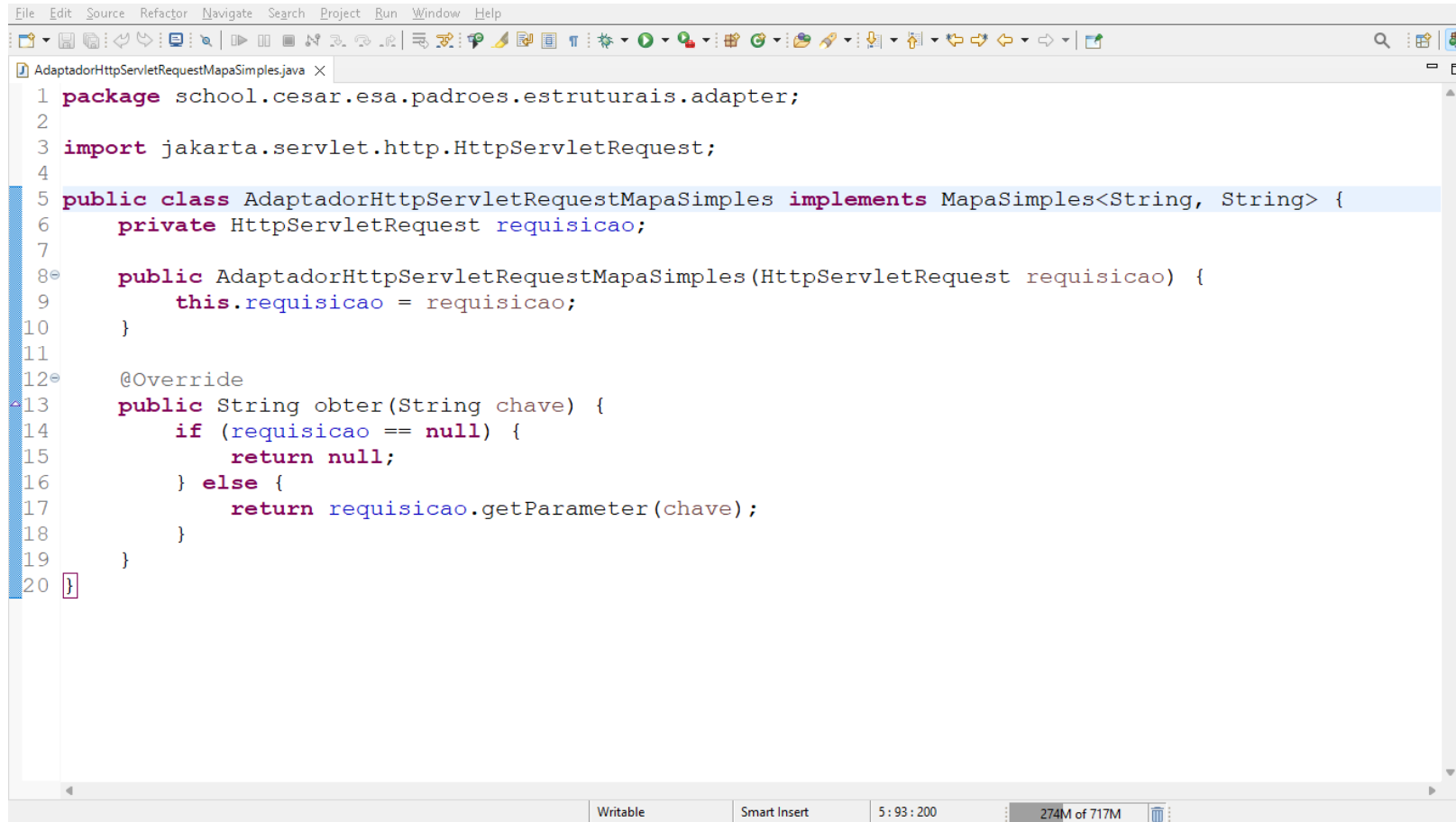
# Adapter



```
File Edit Source Refactor Navigate Search Project Run Window Help
UsuarioFabricaDepois.java
1 package school.cesar.esa.padroes.estruturais.adapter;
2
3 public class UsuarioFabricaDepois {
4     public static final String EMAIL = "email";
5     public static final String NOME = "nome";
6
7     public Usuario criar(MapaSimples<String, String> atributos) {
8         String email = atributos.obter(EMAIL);
9         String nome = atributos.obter(NOME);
10        return new Usuario(email, nome);
11    }
12 }
```

The screenshot shows an IDE window titled 'UsuarioFabricaDepois.java'. The code implements the Adapter pattern. It defines a class 'UsuarioFabricaDepois' with two static final strings, 'EMAIL' and 'NOME', and a 'criar' method that takes a 'MapaSimples' and returns a 'Usuario' object. The IDE interface includes a menu bar, a toolbar, and a status bar at the bottom showing 'Writable', 'Smart Insert', and '3 : 36 : 92'.

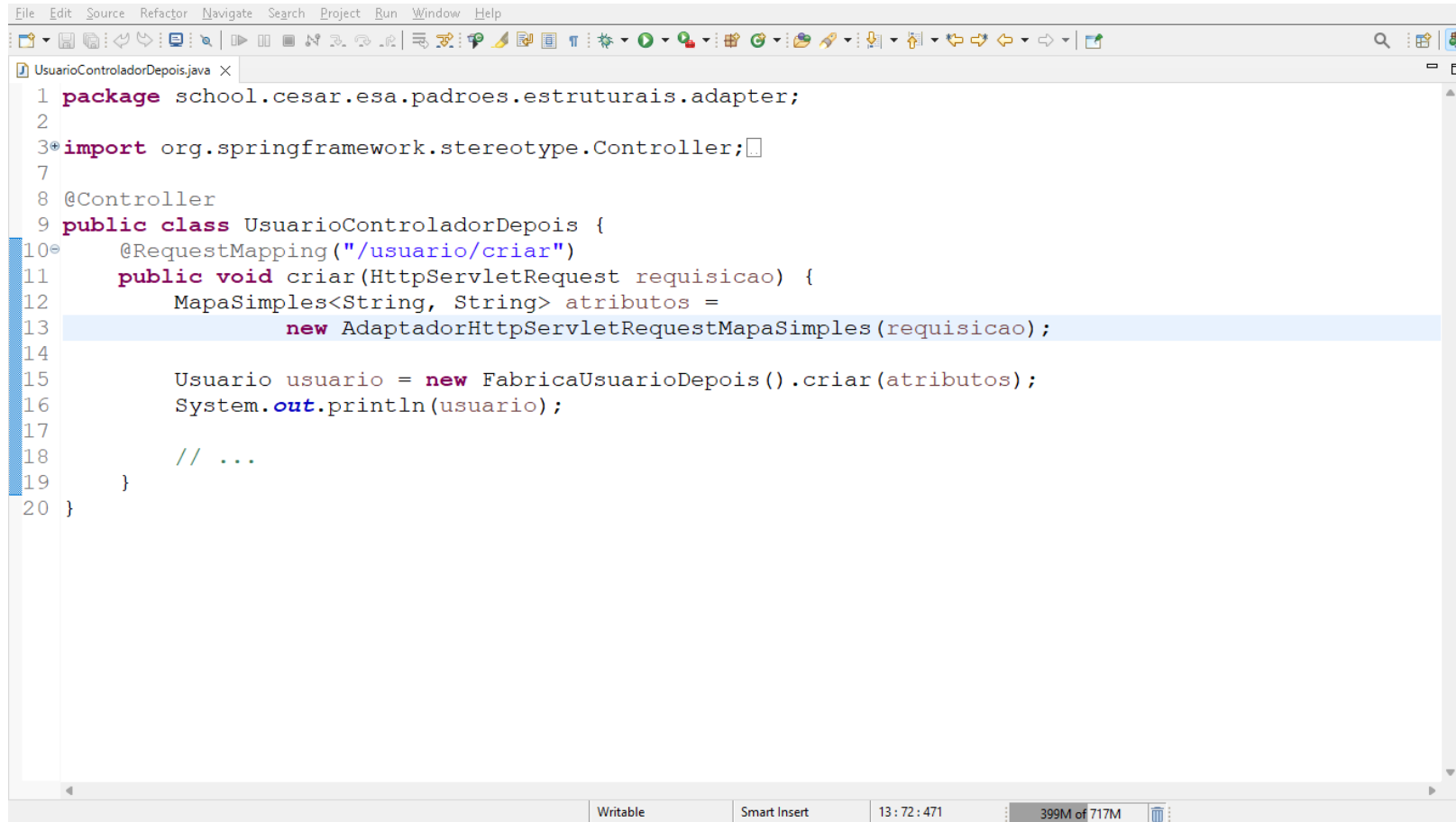
# Adapter



```
File Edit Source Refactor Navigate Search Project Run Window Help
AdaptadorHttpServletRequestMapaSimples.java x
1 package school.cesar.esa.padroes.estruturais.adapter;
2
3 import jakarta.servlet.http.HttpServletRequest;
4
5 public class AdaptadorHttpServletRequestMapaSimples implements MapaSimples<String, String> {
6     private HttpServletRequest requisicao;
7
8     public AdaptadorHttpServletRequestMapaSimples(HttpServletRequest requisicao) {
9         this.requisicao = requisicao;
10    }
11
12    @Override
13    public String obter(String chave) {
14        if (requisicao == null) {
15            return null;
16        } else {
17            return requisicao.getParameter(chave);
18        }
19    }
20 }
```

Writable Smart Insert 5 : 93 : 200 274M of 717M

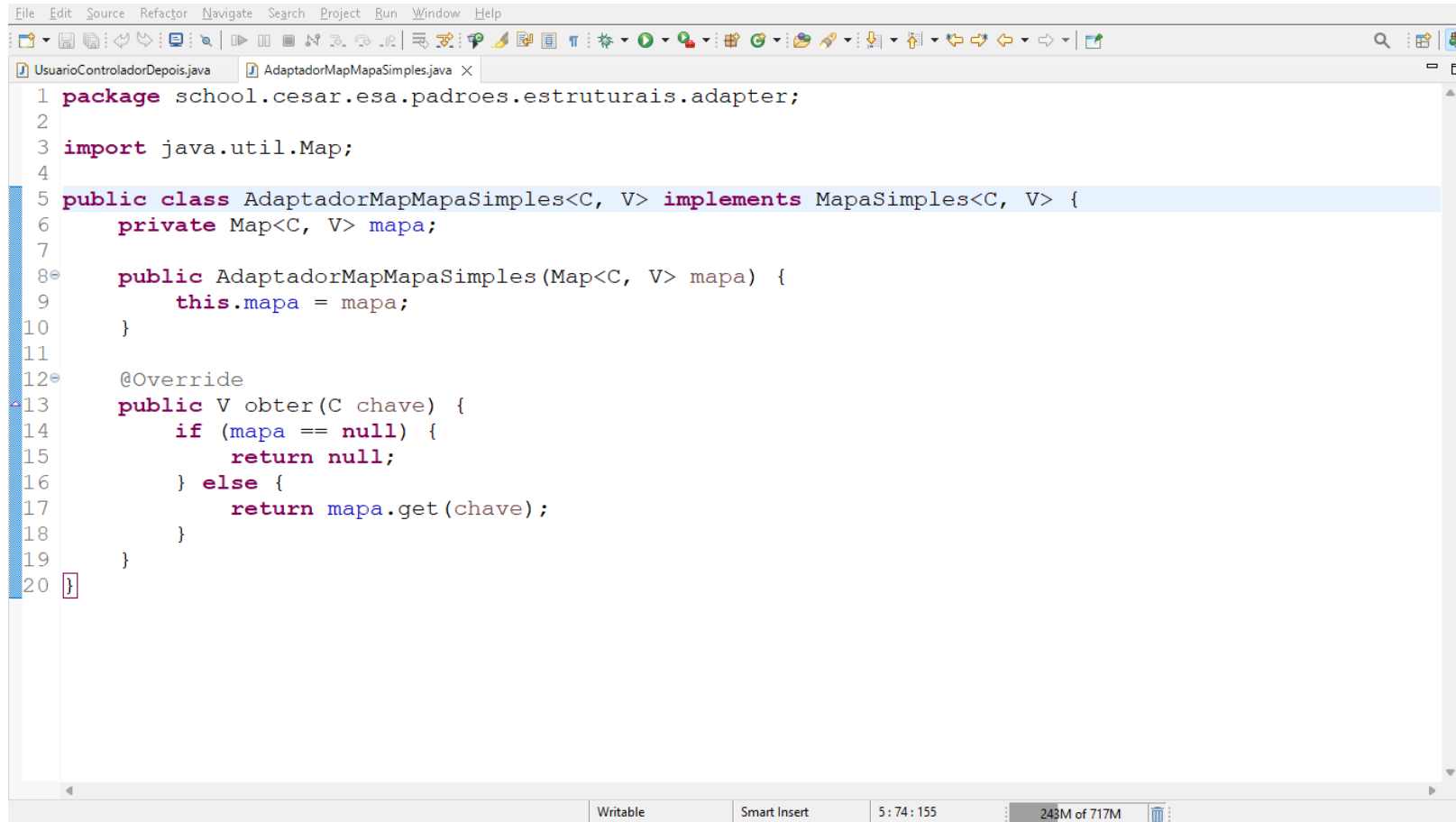
# Adapter



```
File Edit Source Refactor Navigate Search Project Run Window Help
1 package school.cesar.esa.padroes.estruturais.adapter;
2
3 import org.springframework.stereotype.Controller;
4
5 @Controller
6 public class UsuarioControladorDepois {
7     @RequestMapping("/usuario/criar")
8     public void criar(HttpServletRequest request) {
9         Map<String, String> atributos =
10             new AdaptadorHttpServletRequestMapaSimples(request);
11
12         Usuario usuario = new FabricaUsuarioDepois().criar(atributos);
13         System.out.println(usuario);
14
15         // ...
16     }
17 }
```

Writable Smart Insert 13 : 72 : 471 399M of 717M

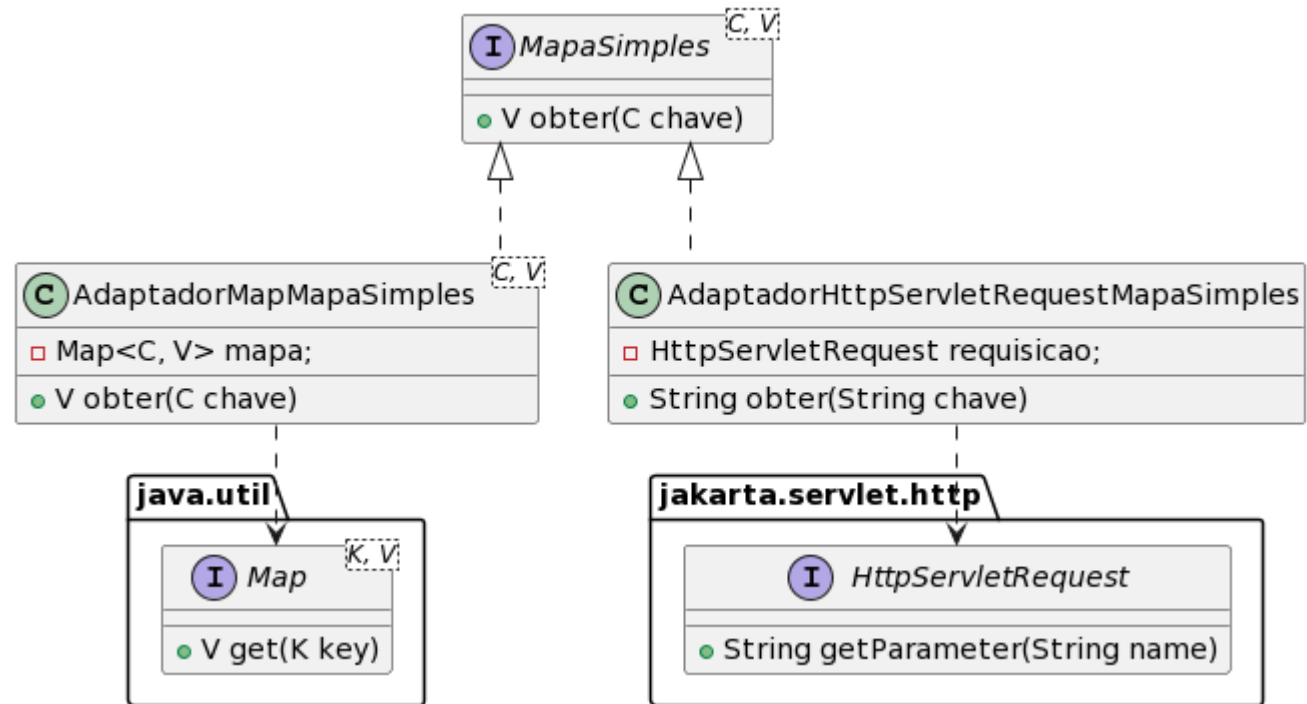
# Adapter



```
File Edit Source Refactor Navigate Search Project Run Window Help
1 package school.cesar.esa.padroes.estruturais.adapter;
2
3 import java.util.Map;
4
5 public class AdaptadorMapMapaSimples<C, V> implements MapaSimples<C, V> {
6     private Map<C, V> mapa;
7
8     public AdaptadorMapMapaSimples(Map<C, V> mapa) {
9         this.mapa = mapa;
10    }
11
12    @Override
13    public V obter(C chave) {
14        if (mapa == null) {
15            return null;
16        } else {
17            return mapa.get(chave);
18        }
19    }
20 }
```

The screenshot shows an IDE window with two tabs: 'UsuarioControladorDepois.java' and 'AdaptadorMapMapaSimples.java'. The 'AdaptadorMapMapaSimples.java' tab is active, displaying the code for the adapter class. The code defines a package, imports the Map interface, and implements the MapaSimples interface. It includes a private Map attribute, a constructor, and an override method for obtaining values from the map. The status bar at the bottom indicates the file is writable, in smart insert mode, at line 5 of 74, with 155 characters, and a memory usage of 243M of 717M.

# Adapter

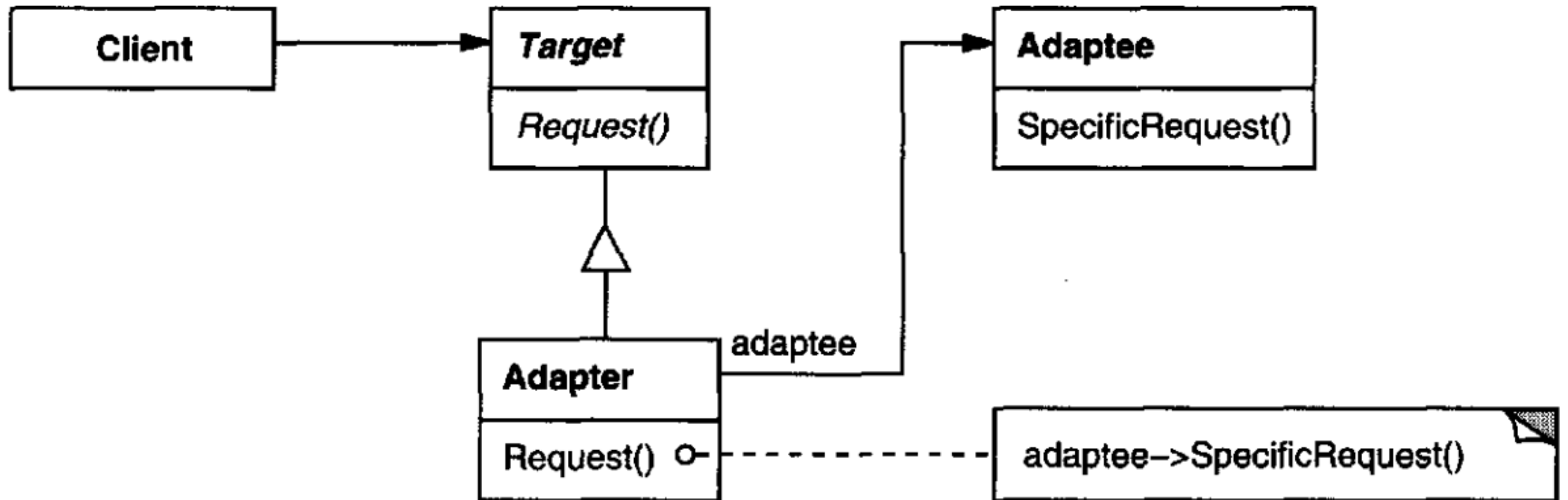


# Adapter

- “Convert the interface of a class into another interface clients expect. Adapter lets classes work together that couldn't otherwise because of incompatible interfaces.”



# Adapter



# Adapter

- SOLID
  - Responsabilidade única (Single responsibility)
  - Aberto-fechado (Open-closed)
  - Substituição de Liskov (Liskob substitution)
  - Segregação de interfaces (Interface segregation)
  - Inversão de dependências (Dependency inversion)
- Prefira composição à herança
- Demeter

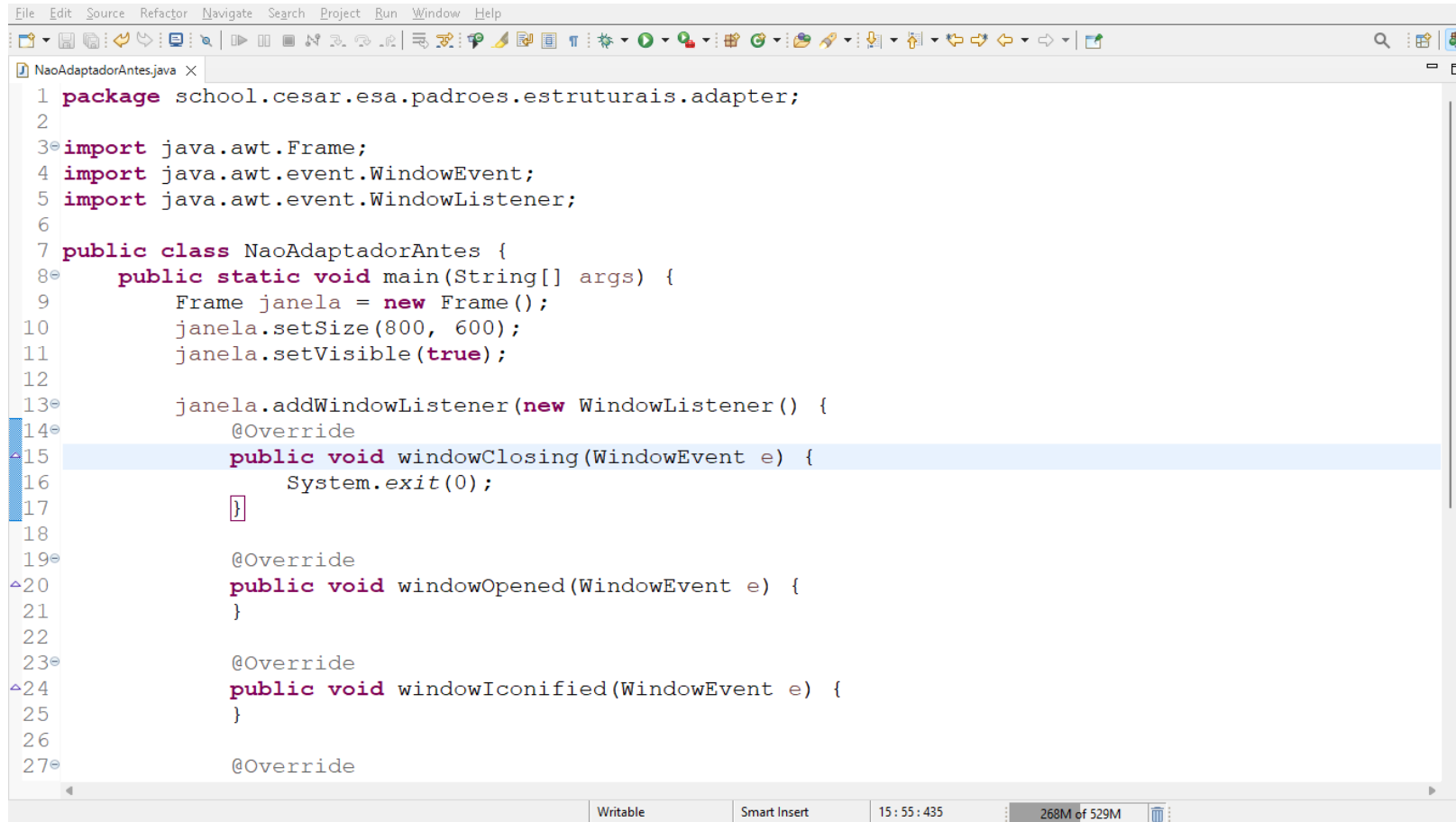
# Adapter

- Integridade conceitual
- (Alta) Coesão
- (Baixo) Acoplamento
- Ocultamento de informações

# Adapter

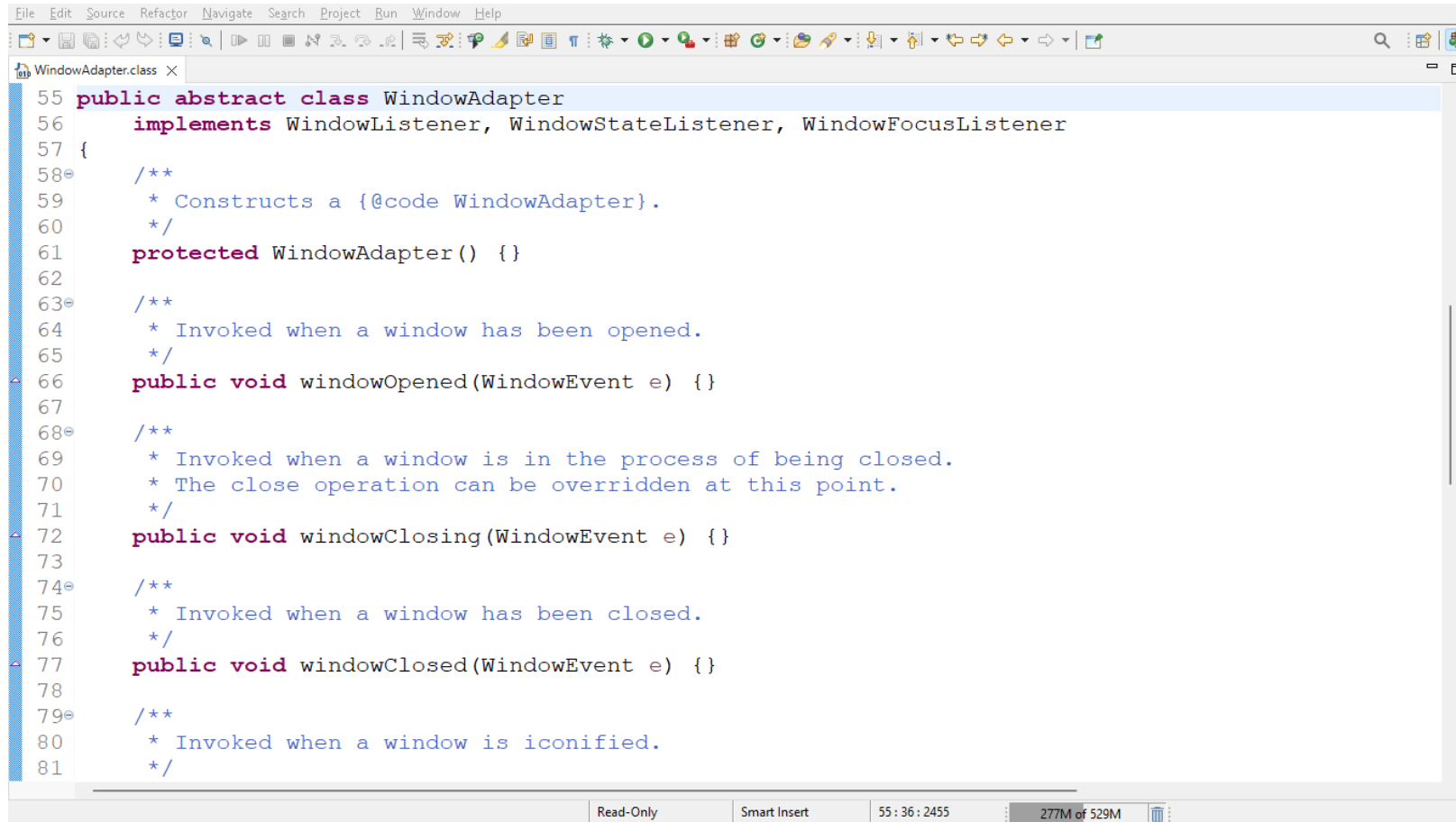


# Adapter



```
1 package school.cesar.esa.padroes.estruturais.adapter;
2
3 import java.awt.Frame;
4 import java.awt.event.WindowEvent;
5 import java.awt.event.WindowListener;
6
7 public class NaoAdaptadorAntes {
8     public static void main(String[] args) {
9         Frame janela = new Frame();
10        janela.setSize(800, 600);
11        janela.setVisible(true);
12
13        janela.addWindowListener(new WindowListener() {
14            @Override
15            public void windowClosing(WindowEvent e) {
16                System.exit(0);
17            }
18
19            @Override
20            public void windowOpened(WindowEvent e) {
21            }
22
23            @Override
24            public void windowIconified(WindowEvent e) {
25            }
26
27            @Override
```

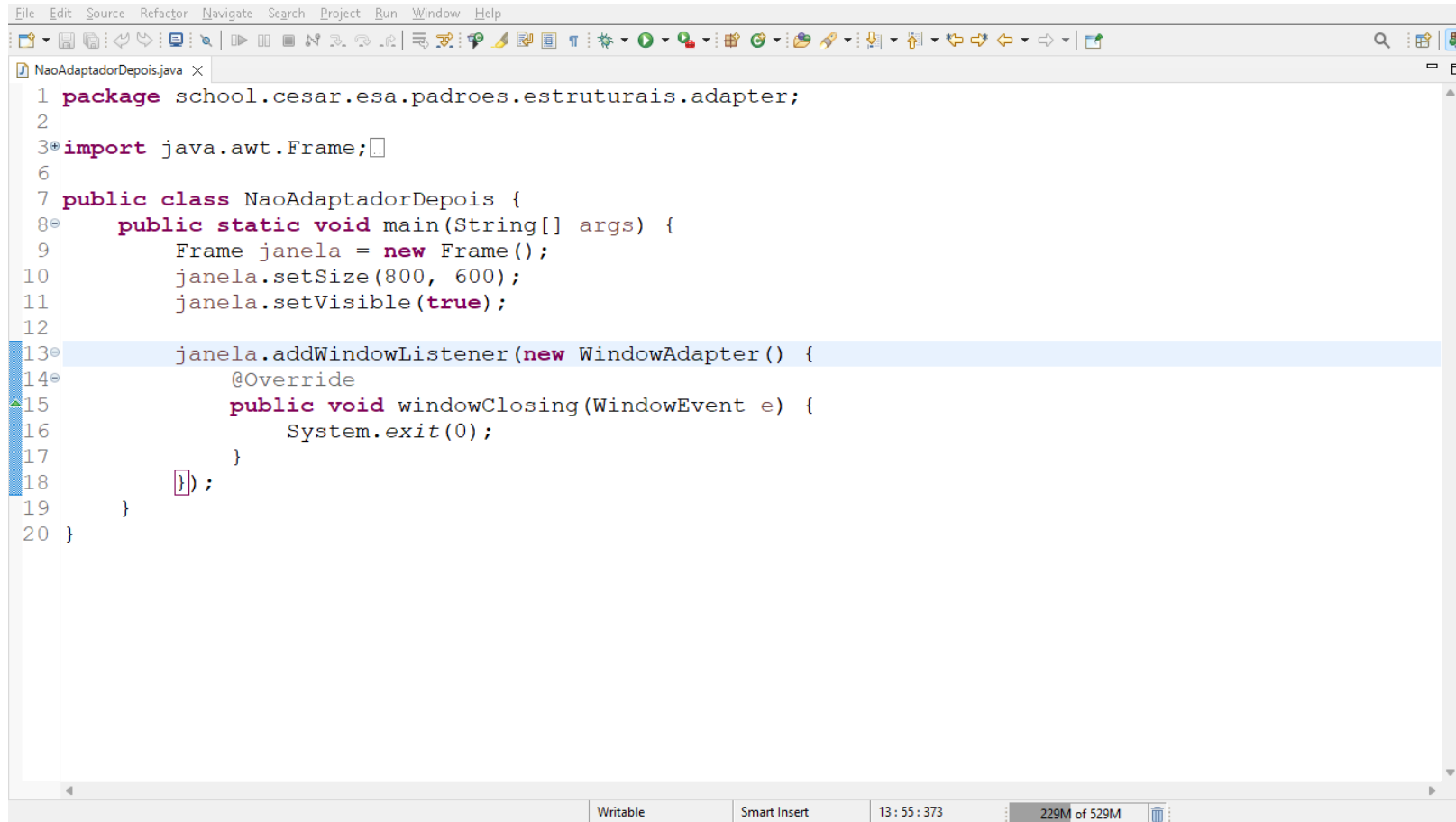
# Adapter



```
File Edit Source Refactor Navigate Search Project Run Window Help
[Icons]
WindowAdapter.class x
55 public abstract class WindowAdapter
56     implements WindowListener, WindowStateListener, WindowFocusListener
57 {
58     /**
59      * Constructs a {@code WindowAdapter}.
60      */
61     protected WindowAdapter() {}
62
63     /**
64      * Invoked when a window has been opened.
65      */
66     public void windowOpened(WindowEvent e) {}
67
68     /**
69      * Invoked when a window is in the process of being closed.
70      * The close operation can be overridden at this point.
71      */
72     public void windowClosing(WindowEvent e) {}
73
74     /**
75      * Invoked when a window has been closed.
76      */
77     public void windowClosed(WindowEvent e) {}
78
79     /**
80      * Invoked when a window is iconified.
81      */
```

Read-Only Smart Insert 55 : 36 : 2455 277M of 529M

# Adapter



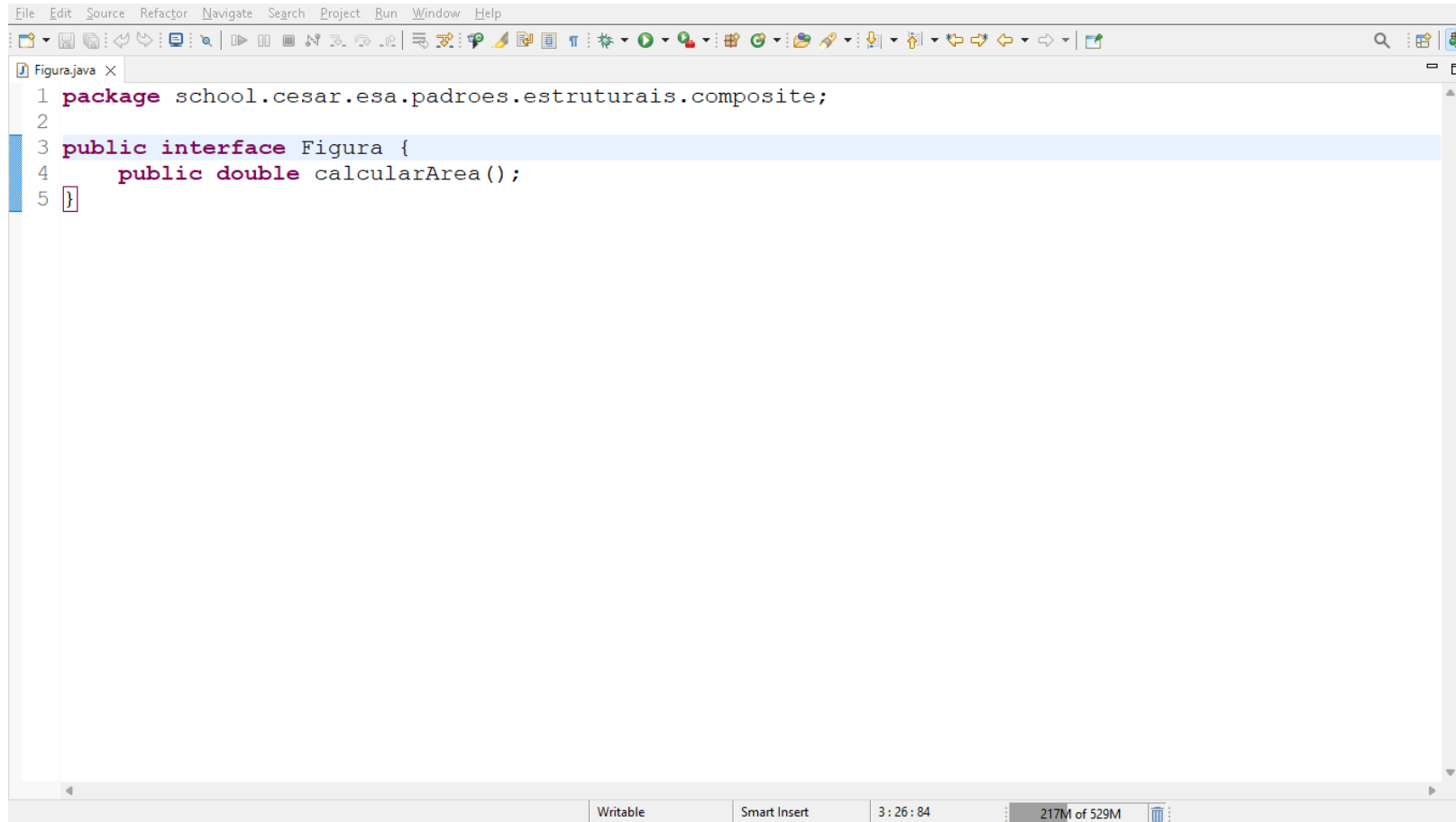
```
1 package school.cesar.esa.padroes.estruturais.adapter;
2
3 import java.awt.Frame;
4
5
6
7 public class NaoAdaptadorDepois {
8     public static void main(String[] args) {
9         Frame janela = new Frame();
10        janela.setSize(800, 600);
11        janela.setVisible(true);
12
13        janela.addWindowListener(new WindowAdapter() {
14            @Override
15            public void windowClosing(WindowEvent e) {
16                System.exit(0);
17            }
18        });
19    }
20 }
```

Writable Smart Insert 13 : 55 : 373 229M of 529M

Composite



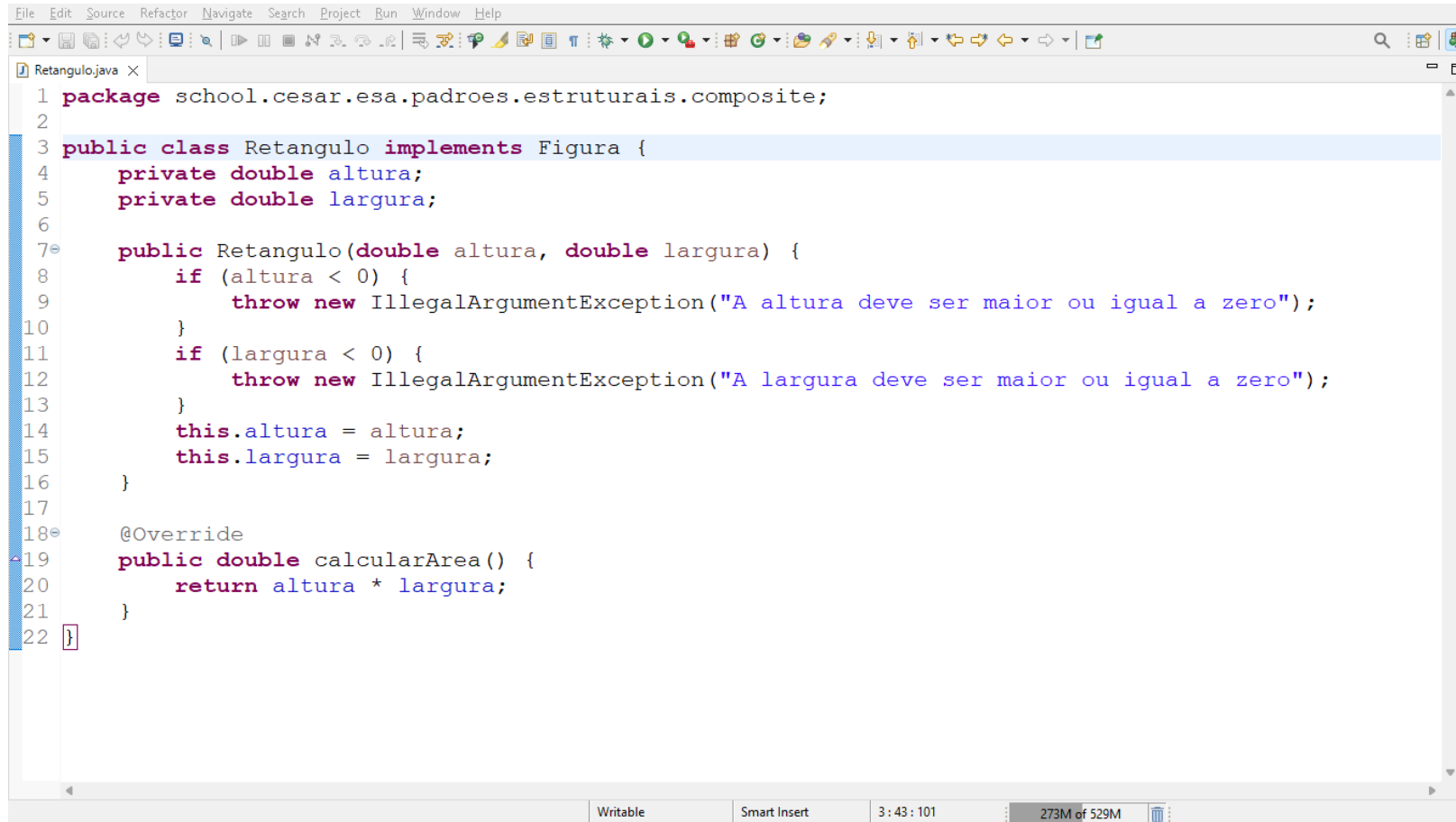
# Composite



```
File Edit Source Refactor Navigate Search Project Run Window Help
1 package school.cesar.esa.padroes.estruturais.composite;
2
3 public interface Figura {
4     public double calcularArea();
5 }
```

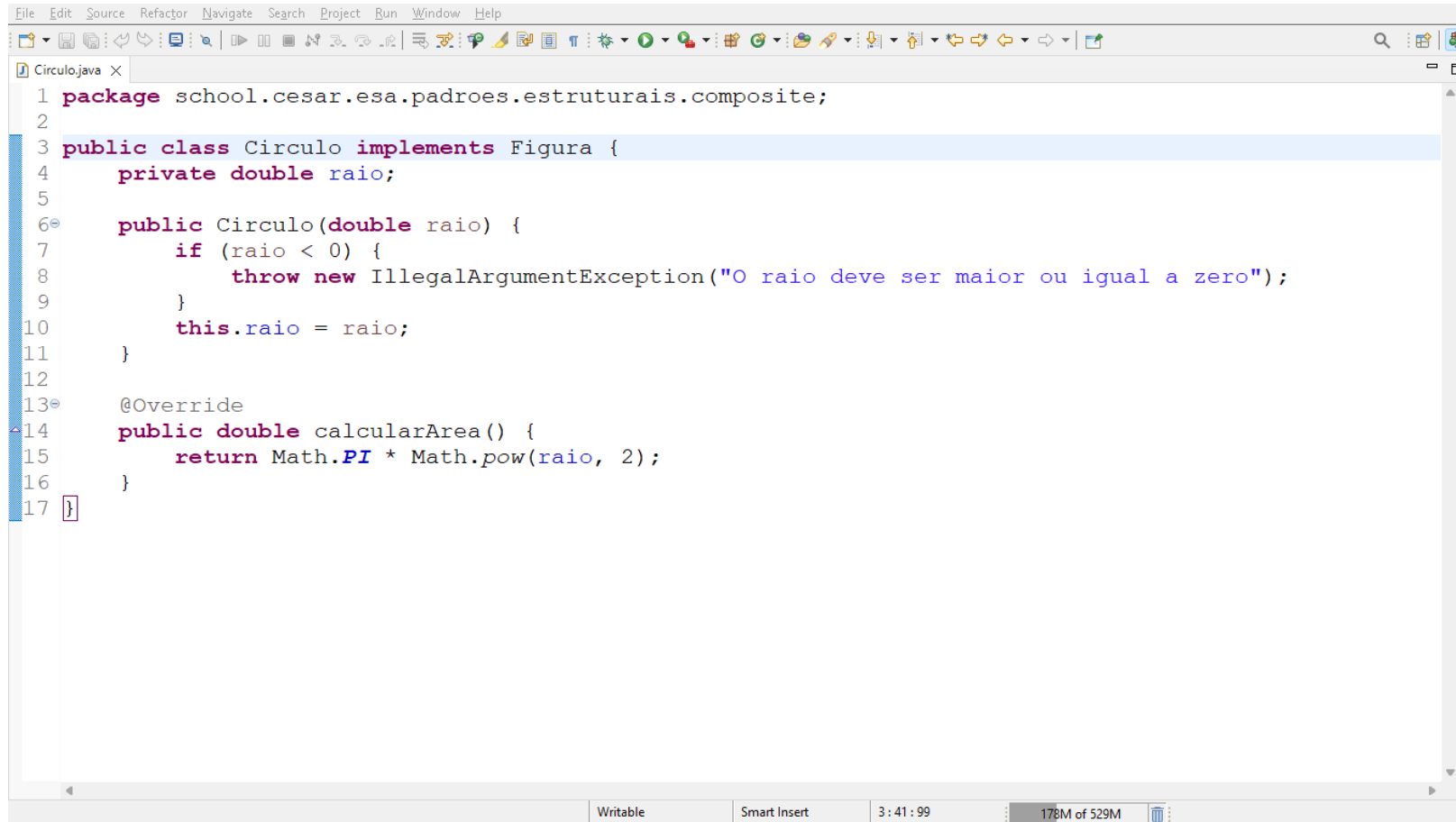
The screenshot shows an IDE window titled 'Figura.java'. The code defines a package 'school.cesar.esa.padroes.estruturais.composite' and a public interface 'Figura'. The interface contains a single method 'calcularArea()' which returns a 'double'. The IDE's status bar at the bottom indicates 'Writable', 'Smart Insert', and '3 : 26 : 84'.

# Composite



```
1 package school.cesar.esa.padroes.estruturais.composite;
2
3 public class Retangulo implements Figura {
4     private double altura;
5     private double largura;
6
7     public Retangulo(double altura, double largura) {
8         if (altura < 0) {
9             throw new IllegalArgumentException("A altura deve ser maior ou igual a zero");
10        }
11        if (largura < 0) {
12            throw new IllegalArgumentException("A largura deve ser maior ou igual a zero");
13        }
14        this.altura = altura;
15        this.largura = largura;
16    }
17
18    @Override
19    public double calcularArea() {
20        return altura * largura;
21    }
22 }
```

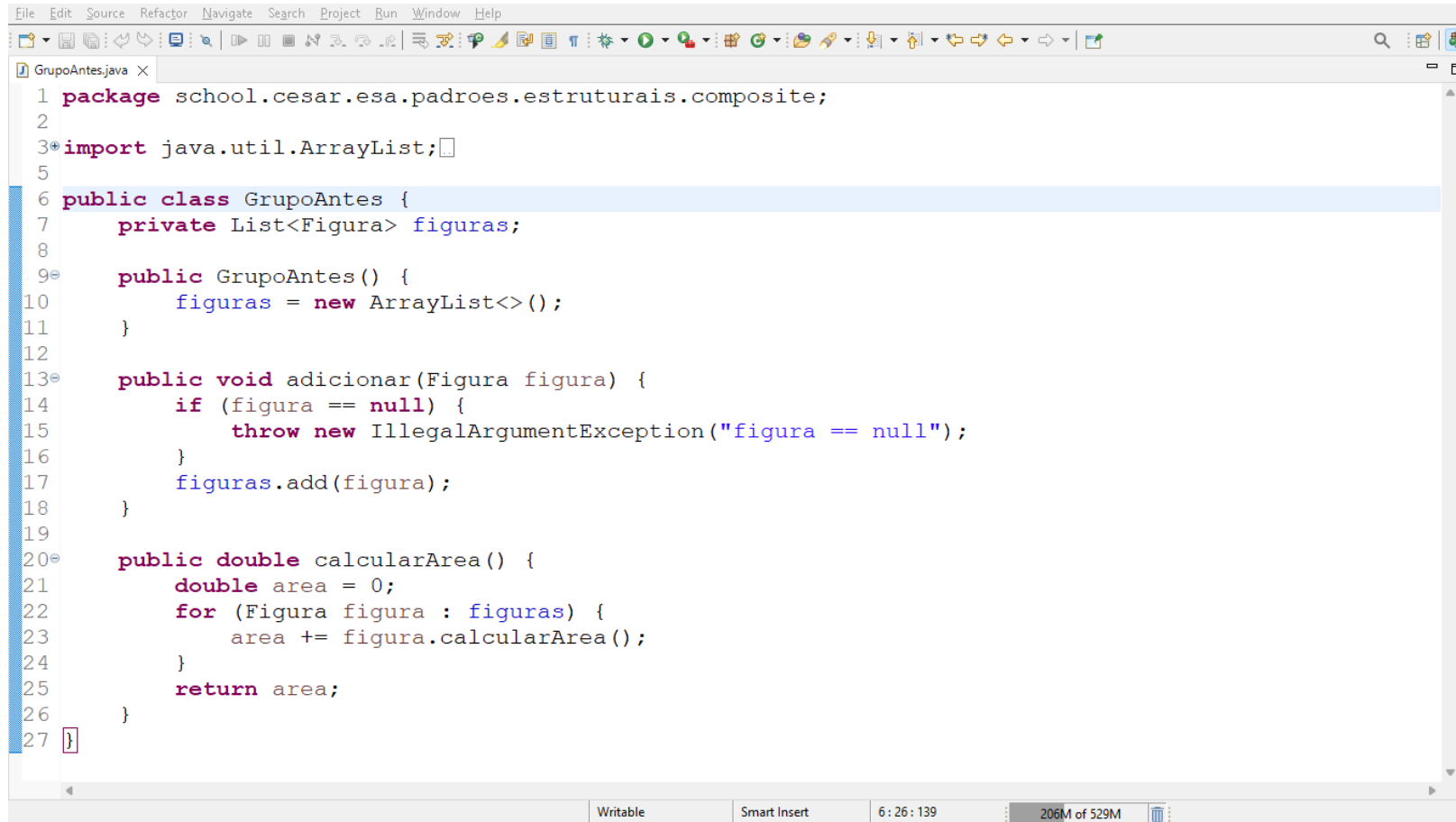
# Composite



```
File Edit Source Refactor Navigate Search Project Run Window Help
Circulo.java x
1 package school.cesar.esa.padroes.estruturais.composite;
2
3 public class Circulo implements Figura {
4     private double raio;
5
6     public Circulo(double raio) {
7         if (raio < 0) {
8             throw new IllegalArgumentException("O raio deve ser maior ou igual a zero");
9         }
10        this.raio = raio;
11    }
12
13    @Override
14    public double calcularArea() {
15        return Math.PI * Math.pow(raio, 2);
16    }
17 }
```

The screenshot shows an IDE window titled 'Circulo.java'. The code implements the Composite design pattern. It starts with a package declaration 'school.cesar.esa.padroes.estruturais.composite;'. Then, a public class 'Circulo' is declared, implementing the 'Figura' interface. A private double attribute 'raio' is defined. The constructor 'Circulo(double raio)' checks if the radius is non-negative, throwing an 'IllegalArgumentException' if it is. The 'calcularArea()' method calculates the area using the formula  $\pi \times \text{raio}^2$ . The IDE interface includes a menu bar, a toolbar, and a status bar at the bottom showing 'Writable', 'Smart Insert', and memory usage '178M of 529M'.

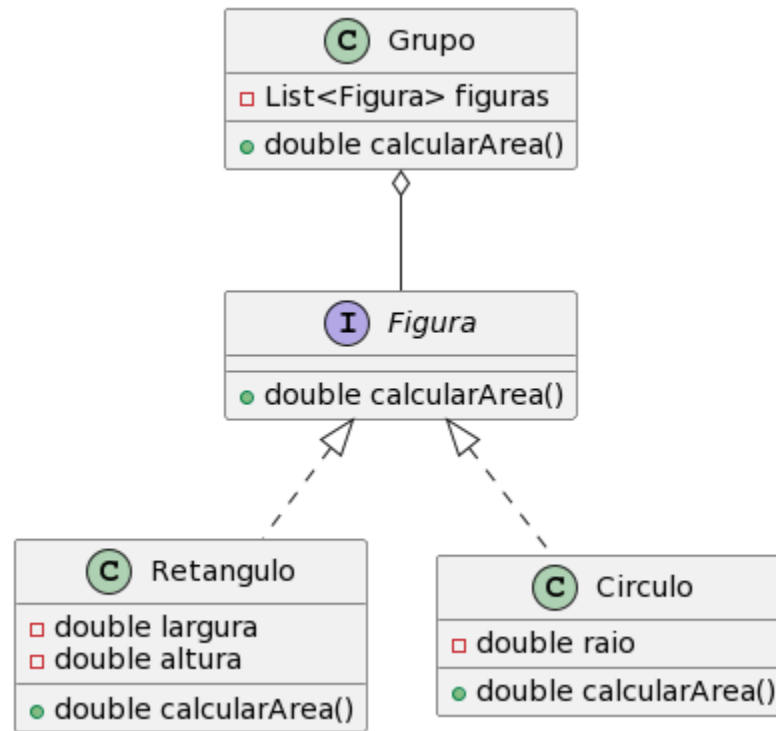
# Composite



```
File Edit Source Refactor Navigate Search Project Run Window Help
GrupoAntes.java x
1 package school.cesar.esa.padroes.estruturais.composite;
2
3 import java.util.ArrayList;
4
5
6 public class GrupoAntes {
7     private List<Figura> figuras;
8
9     public GrupoAntes() {
10         figuras = new ArrayList<>();
11     }
12
13     public void adicionar(Figura figura) {
14         if (figura == null) {
15             throw new IllegalArgumentException("figura == null");
16         }
17         figuras.add(figura);
18     }
19
20     public double calcularArea() {
21         double area = 0;
22         for (Figura figura : figuras) {
23             area += figura.calcularArea();
24         }
25         return area;
26     }
27 }
```

Writable Smart Insert 6 : 26 : 139 206M of 529M

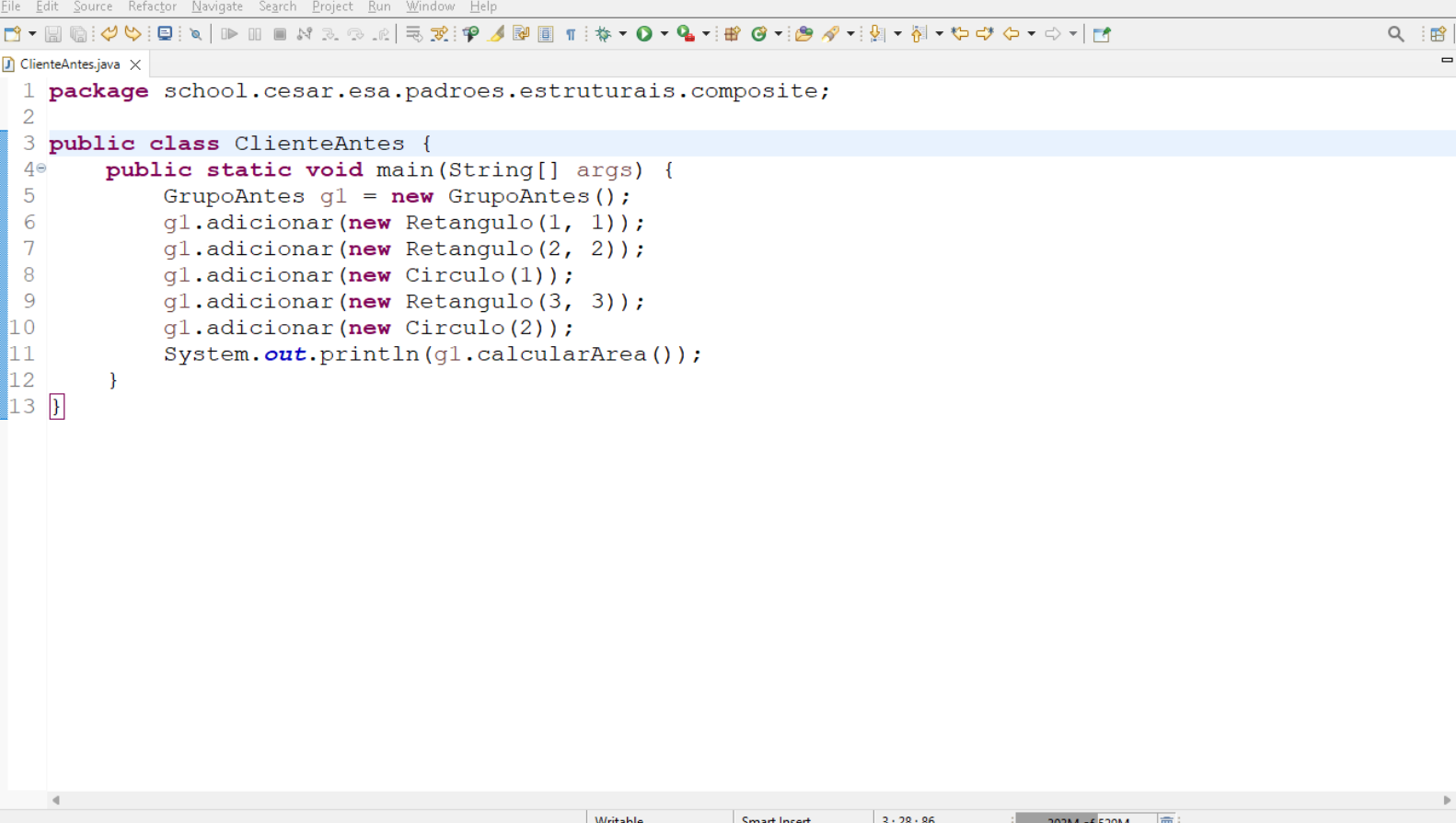
# Composite



# Composite

- g1
  - r1
  - r2
  - c1
  - r3
  - c2

# Composite



```
1 package school.cesar.esa.padroes.estruturais.composite;
2
3 public class ClienteAntes {
4     public static void main(String[] args) {
5         GrupoAntes g1 = new GrupoAntes();
6         g1.adicionar(new Retangulo(1, 1));
7         g1.adicionar(new Retangulo(2, 2));
8         g1.adicionar(new Circulo(1));
9         g1.adicionar(new Retangulo(3, 3));
10        g1.adicionar(new Circulo(2));
11        System.out.println(g1.calcularArea());
12    }
13 }
```

# Composite

- g1
  - r1
  - r2
  - c1
  - r3
  - c2
  - g2
    - c3
    - c4
    - r5



# Composite

The screenshot shows an IDE with a Java file named `ClienteAntes.java`. The code defines a package `school.cesar.esa.padroes.estruturais.composite;` and a class `ClienteAntes` with a `main` method. The `main` method creates two `GrupoAntes` objects, `g1` and `g2`, and adds various shapes to them. Line 16, `g1.adicionar(g2);`, is highlighted with a red error marker. The `Problems` view on the right shows one error and three warnings. A `Properties for Java Problem` dialog is open, displaying the details of the error.

```
1 package school.cesar.esa.padroes.estruturais.composite;
2
3 public class ClienteAntes {
4     public static void main(String[] args) {
5         GrupoAntes g2 = new GrupoAntes();
6         g2.adicionar(new Circulo(3));
7         g2.adicionar(new Circulo(4));
8         g2.adicionar(new Retangulo(5, 5));
9
10        GrupoAntes g1 = new GrupoAntes();
11        g1.adicionar(new Retangulo(1, 1));
12        g1.adicionar(new Retangulo(2, 2));
13        g1.adicionar(new Circulo(1));
14        g1.adicionar(new Retangulo(3, 3));
15        g1.adicionar(new Circulo(2));
16        g1.adicionar(g2);
17        System.out.println(g1.calcularArea());
18    }
19 }
```

**Problems View:** 1 error, 3 warnings, 0 others

Description	Resource	Path	Location
✖ Errors (1 item)			
✖ The method adicionar(Figura) in the type Grup...	ClienteAntes.j...	/composite/src/ma...	line 16
> ⚠ Warnings (3 items)			

**Properties for Java Problem**

**Marker**

Severity: ✖ Error

On element: ClienteAntes.java

Path: composite/src/main/java/school/cesar/esa/padroes/estruturais/composite

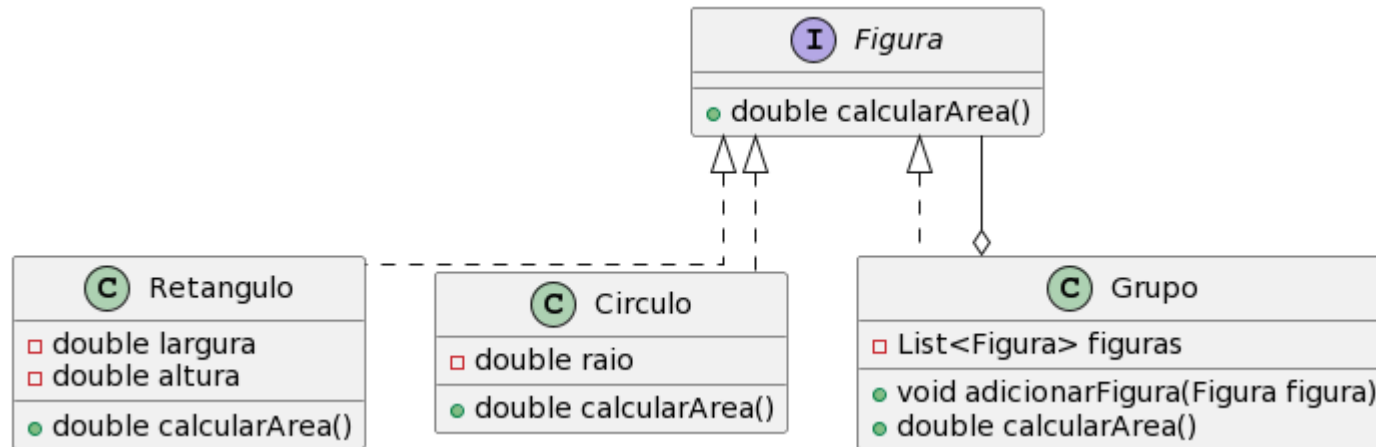
Location: line 16

Creation time: 17 de março de 2024 12:24:23

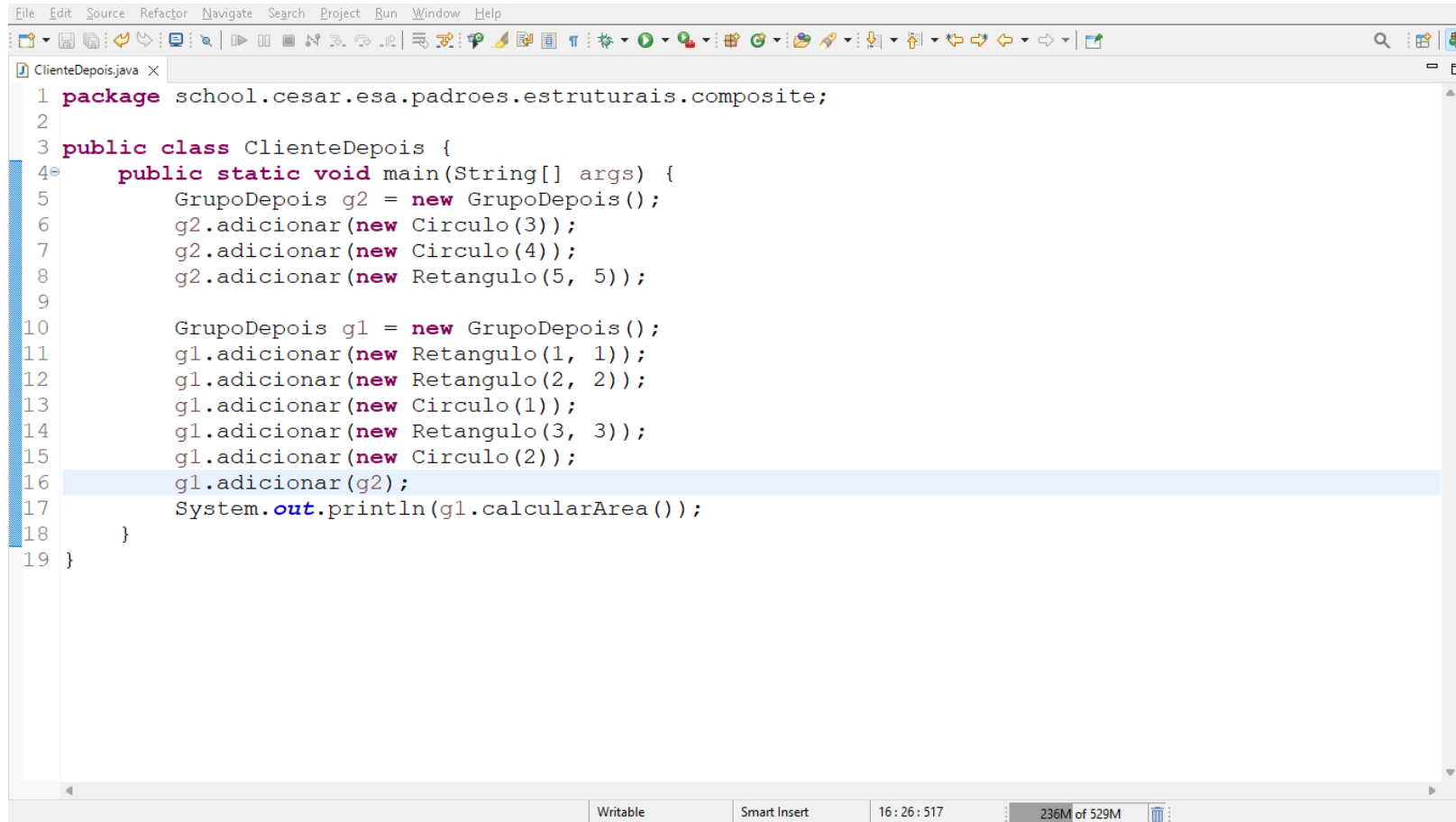
Description: The method adicionar(Figura) in the type GrupoAntes is not applicable for the arguments (GrupoAntes)

Buttons: Apply and Close, Cancel

# Composite



# Composite



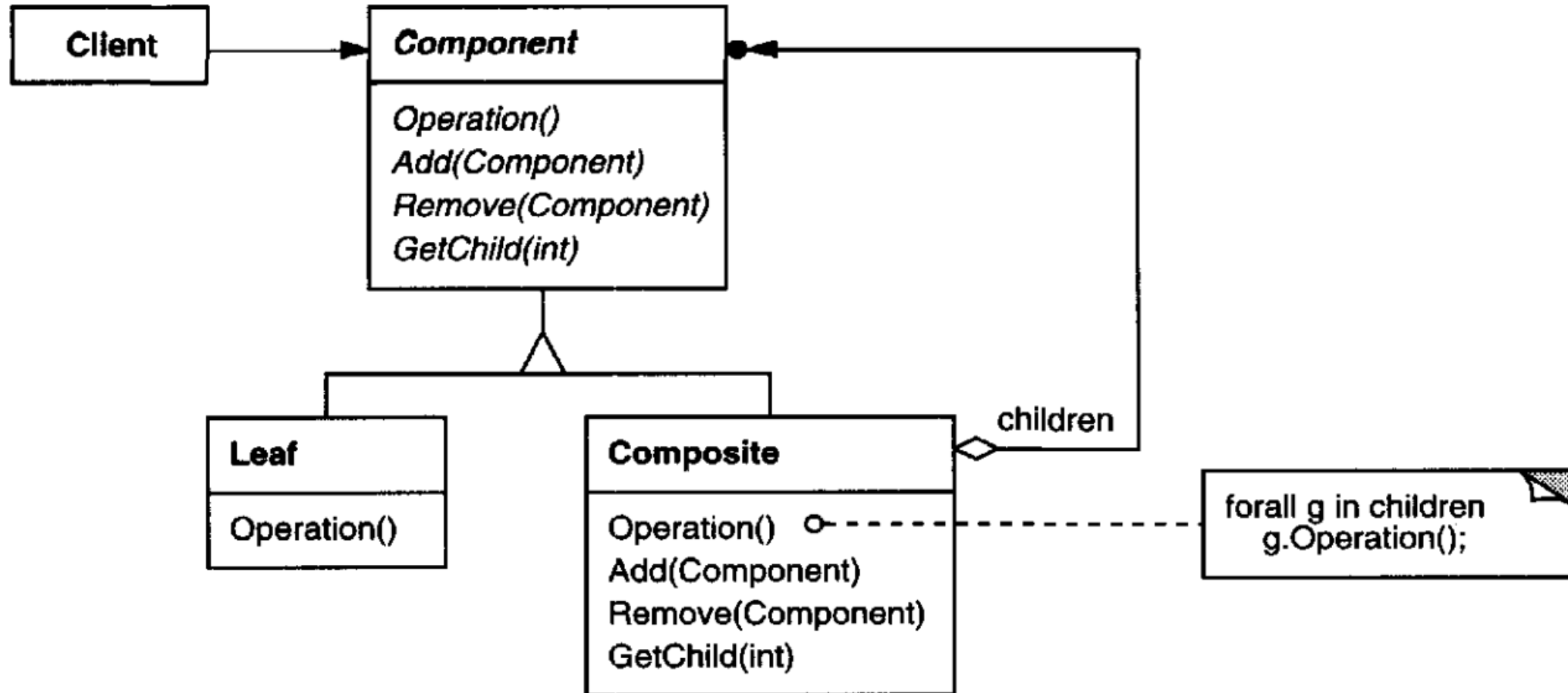
```
File Edit Source Refactor Navigate Search Project Run Window Help
[Icons]
ClienteDepois.java x
1 package school.cesar.esa.padroes.estruturais.composite;
2
3 public class ClienteDepois {
4     public static void main(String[] args) {
5         GrupoDepois g2 = new GrupoDepois();
6         g2.adicionar(new Circulo(3));
7         g2.adicionar(new Circulo(4));
8         g2.adicionar(new Retangulo(5, 5));
9
10        GrupoDepois g1 = new GrupoDepois();
11        g1.adicionar(new Retangulo(1, 1));
12        g1.adicionar(new Retangulo(2, 2));
13        g1.adicionar(new Circulo(1));
14        g1.adicionar(new Retangulo(3, 3));
15        g1.adicionar(new Circulo(2));
16        g1.adicionar(g2);
17        System.out.println(g1.calcularArea());
18    }
19 }
```

Writable Smart Insert 16 : 26 : 517 236M of 529M

# Composite

“Compose objects into tree structures to represent part-whole hierarchies. Composite lets clients treat individual objects and compositions of objects uniformly.”

# Composite



# Composite

- SOLID
  - Responsabilidade única (Single responsibility)
  - Aberto-fechado (Open-closed)
  - Substituição de Liskov (Liskob substitution)
  - Segregação de interfaces (Interface segregation)
  - Inversão de dependências (Dependency inversion)
- Prefira composição à herança
- Demeter

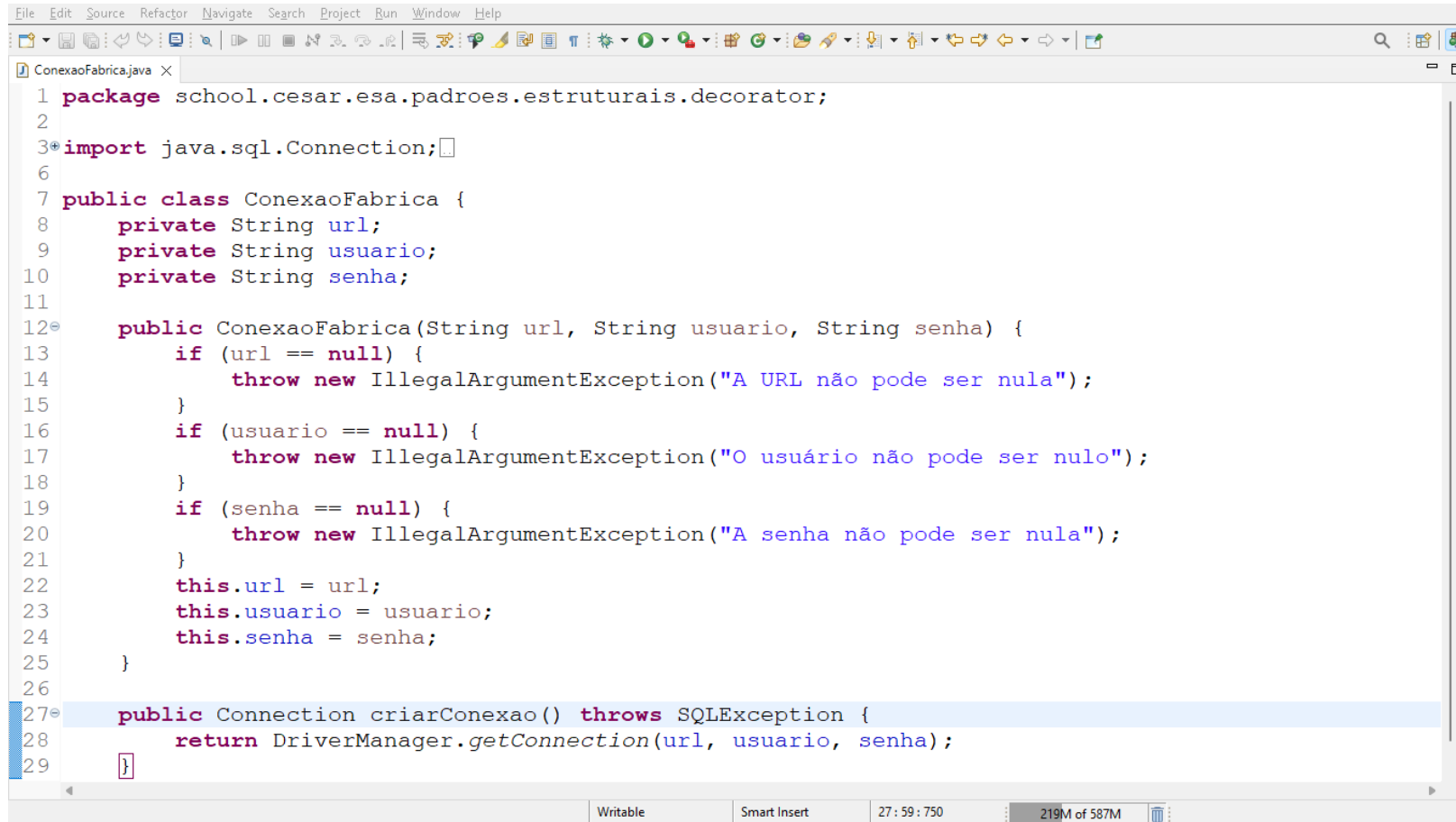
# Composite

- Integridade conceitual
- (Alta) Coesão
- (Baixo) Acoplamento
- Ocultamento de informações

# Decorator



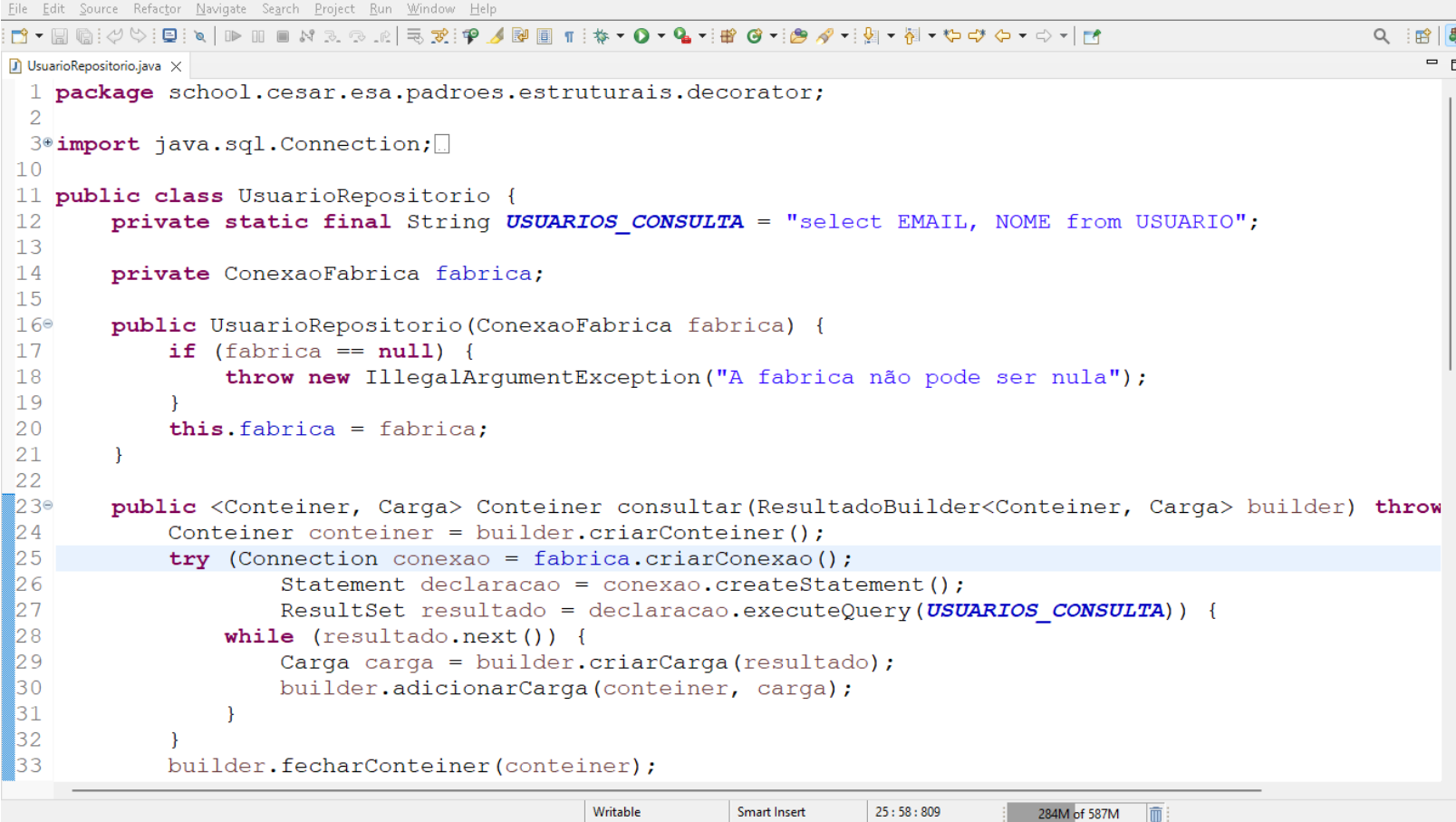
# Decorator



```
File Edit Source Refactor Navigate Search Project Run Window Help
ConexaoFabrica.java X
1 package school.cesar.esa.padroes.estruturais.decorator;
2
3 import java.sql.Connection;
4
5
6
7 public class ConexaoFabrica {
8     private String url;
9     private String usuario;
10    private String senha;
11
12    public ConexaoFabrica(String url, String usuario, String senha) {
13        if (url == null) {
14            throw new IllegalArgumentException("A URL não pode ser nula");
15        }
16        if (usuario == null) {
17            throw new IllegalArgumentException("O usuário não pode ser nulo");
18        }
19        if (senha == null) {
20            throw new IllegalArgumentException("A senha não pode ser nula");
21        }
22        this.url = url;
23        this.usuario = usuario;
24        this.senha = senha;
25    }
26
27    public Connection criarConexao() throws SQLException {
28        return DriverManager.getConnection(url, usuario, senha);
29    }
}
```

Writable Smart Insert 27 : 59 : 750 219M of 587M

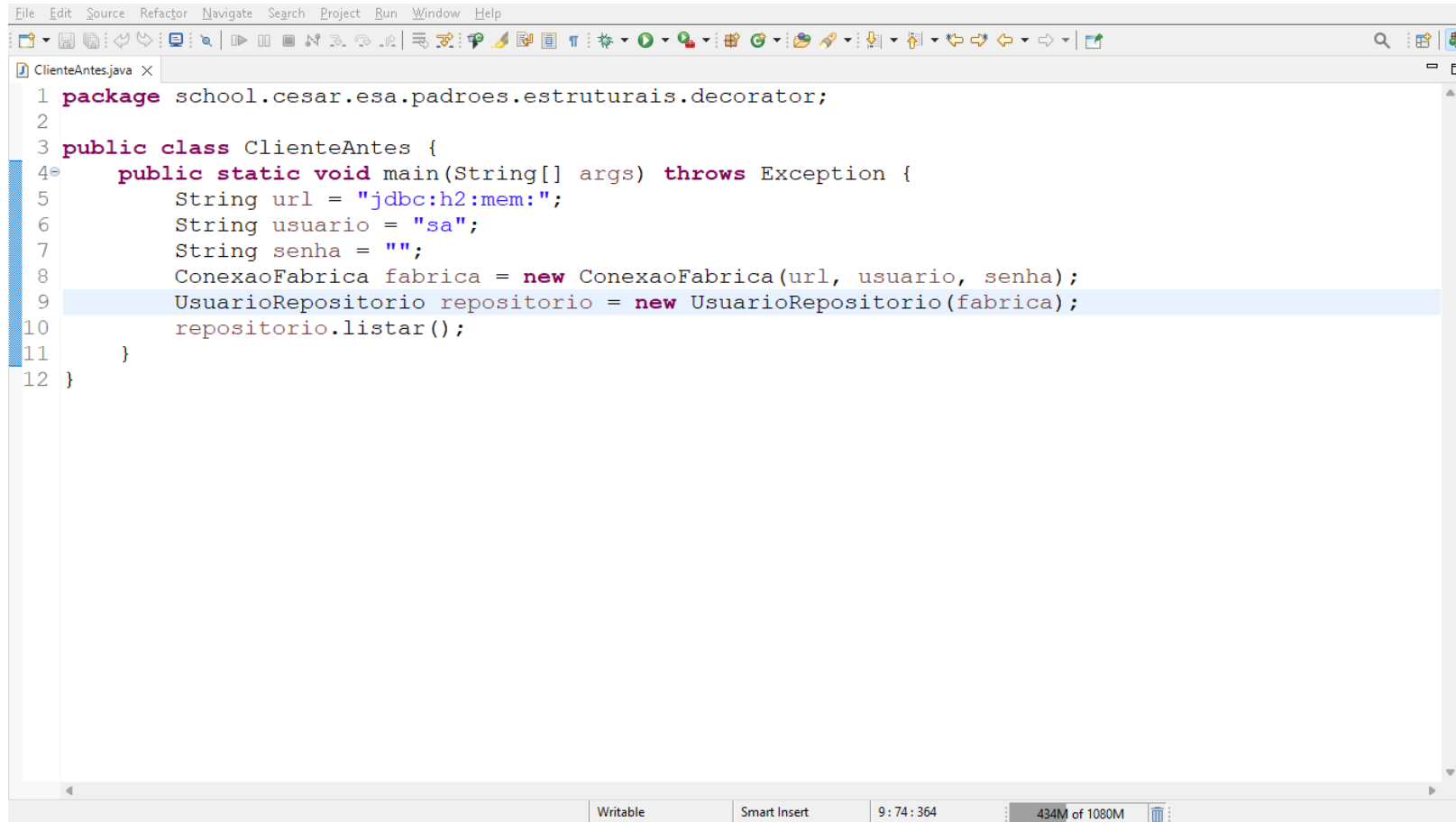
# Decorator



```
File Edit Source Refactor Navigate Search Project Run Window Help
UsuarioRepositorio.java X
1 package school.cesar.esa.padroes.estruturais.decorator;
2
3 import java.sql.Connection;
4
5
6
7
8
9
10
11 public class UsuarioRepositorio {
12     private static final String USUARIOS_CONSULTA = "select EMAIL, NOME from USUARIO";
13
14     private ConexaoFabrica fabrica;
15
16     public UsuarioRepositorio(ConexaoFabrica fabrica) {
17         if (fabrica == null) {
18             throw new IllegalArgumentException("A fabrica não pode ser nula");
19         }
20         this.fabrica = fabrica;
21     }
22
23     public <Container, Carga> Container consultar(ResultadoBuilder<Container, Carga> builder) throw
24         Container container = builder.criarContainer();
25         try (Connection conexao = fabrica.criarConexao();
26             Statement declaracao = conexao.createStatement();
27             ResultSet resultado = declaracao.executeQuery(USUARIOS_CONSULTA)) {
28             while (resultado.next()) {
29                 Carga carga = builder.criarCarga(resultado);
30                 builder.adicionarCarga(container, carga);
31             }
32         }
33         builder.fecharContainer(container);
34     }
35 }
```

Writable Smart Insert 25 : 58 : 809 284M of 587M

# Decorator



```
File Edit Source Refactor Navigate Search Project Run Window Help
1 package school.cesar.esa.padroes.estruturais.decorator;
2
3 public class ClienteAntes {
4     public static void main(String[] args) throws Exception {
5         String url = "jdbc:h2:mem:";
6         String usuario = "sa";
7         String senha = "";
8         ConexaoFabrica fabrica = new ConexaoFabrica(url, usuario, senha);
9         UsuarioRepositorio repositorio = new UsuarioRepositorio(fabrica);
10        repositorio.listar();
11    }
12 }
```

Writable Smart Insert 9 : 74 : 364 434M of 1080M

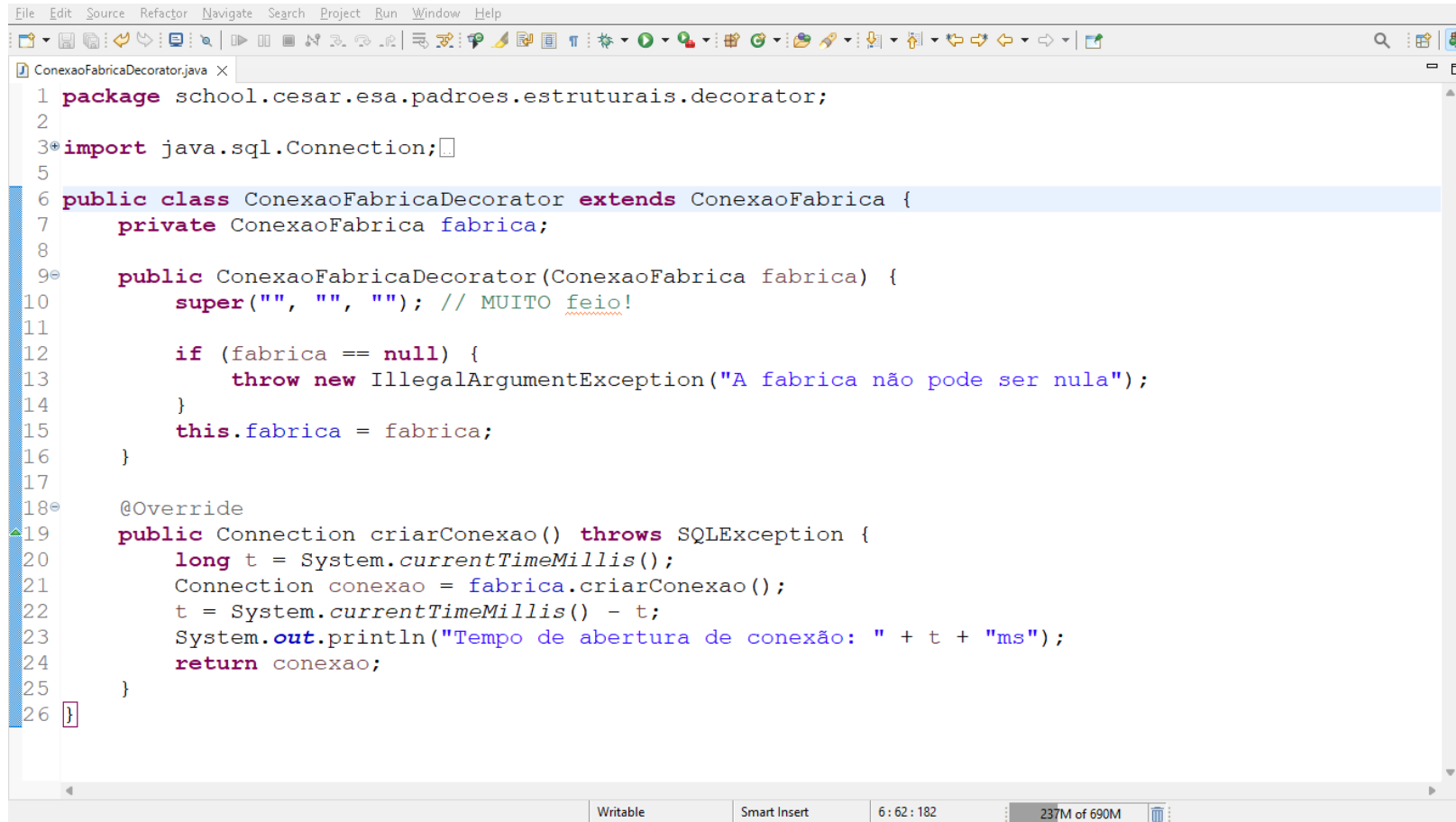
# Decorator

- Alguns SGBDs podem ser configurados para
  - Aceitar um número máximo de conexões abertas simultaneamente
  - Após esse número ser alcançado, um processo que tente abrir uma nova conexão ficará esperando até que alguma das conexões seja fechada
- Como podemos medir e registrar o tempo de abertura de uma conexão sem alterar a classe `ConexaoFabrica`?

# Decorator



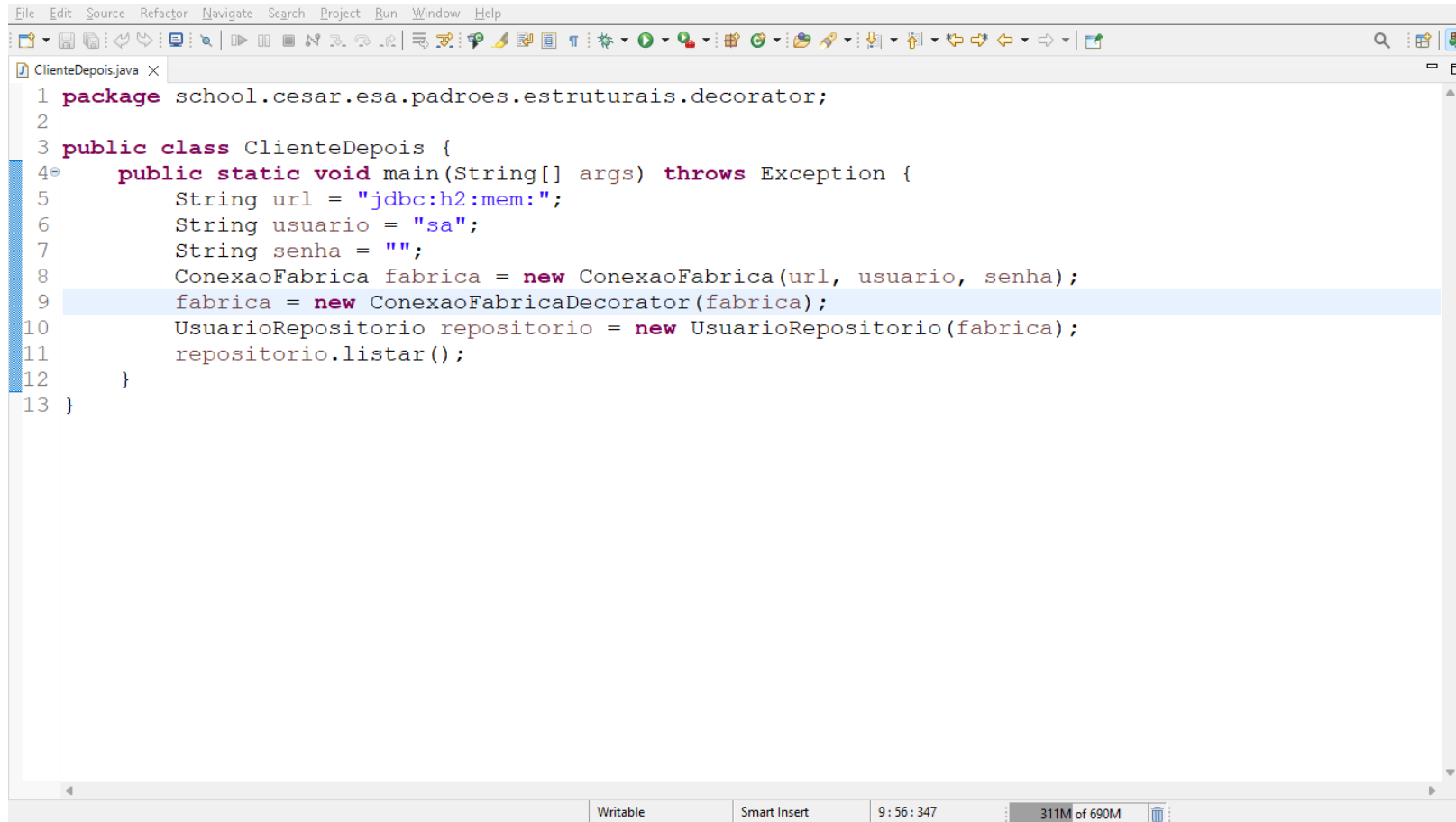
# Decorator



```
File Edit Source Refactor Navigate Search Project Run Window Help
ConexaoFabricaDecorator.java X
1 package school.cesar.esa.padroes.estruturais.decorator;
2
3 import java.sql.Connection;
4
5
6 public class ConexaoFabricaDecorator extends ConexaoFabrica {
7     private ConexaoFabrica fabrica;
8
9     public ConexaoFabricaDecorator(ConexaoFabrica fabrica) {
10         super("", "", ""); // MUITO feio!
11
12         if (fabrica == null) {
13             throw new IllegalArgumentException("A fabrica não pode ser nula");
14         }
15         this.fabrica = fabrica;
16     }
17
18     @Override
19     public Connection criarConexao() throws SQLException {
20         long t = System.currentTimeMillis();
21         Connection conexao = fabrica.criarConexao();
22         t = System.currentTimeMillis() - t;
23         System.out.println("Tempo de abertura de conexão: " + t + "ms");
24         return conexao;
25     }
26 }
```

Writable Smart Insert 6 : 62 : 182 237M of 690M

# Decorator



```
1 package school.cesar.esa.padroes.estruturais.decorator;  
2  
3 public class ClienteDepois {  
4     public static void main(String[] args) throws Exception {  
5         String url = "jdbc:h2:mem:";  
6         String usuario = "sa";  
7         String senha = "";  
8         ConexaoFabrica fabrica = new ConexaoFabrica(url, usuario, senha);  
9         fabrica = new ConexaoFabricaDecorator(fabrica);  
10        UsuarioRepositorio repositorio = new UsuarioRepositorio(fabrica);  
11        repositorio.listar();  
12    }  
13 }
```

Writable | Smart Insert | 9 : 56 : 347 | 311M of 690M

# Decorator

---

## GETTING BETTER

Songs and Music by JOHN LENNON and PAUL McCARTNEY

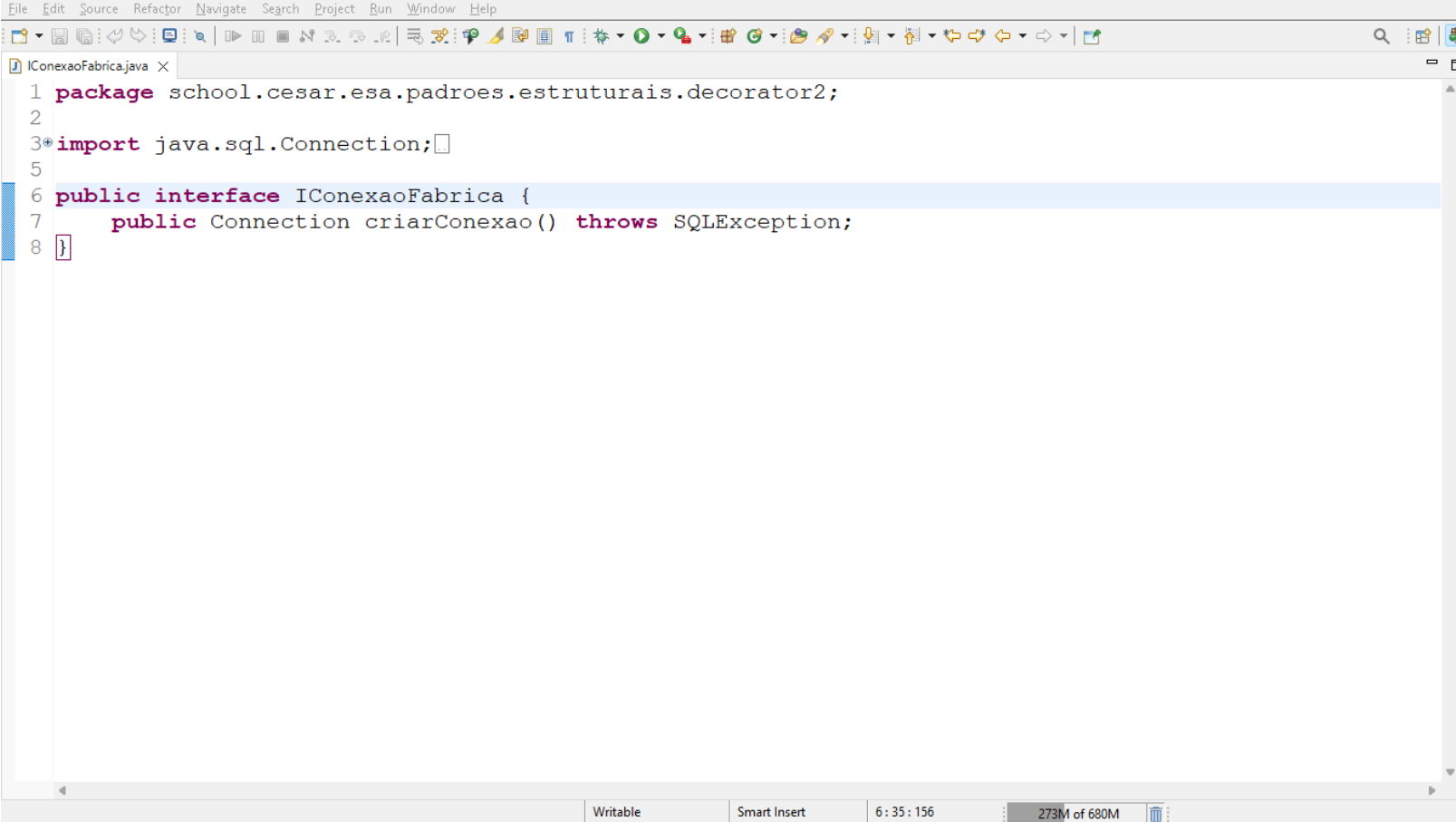


NORTHERN SONGS LIMITED

34



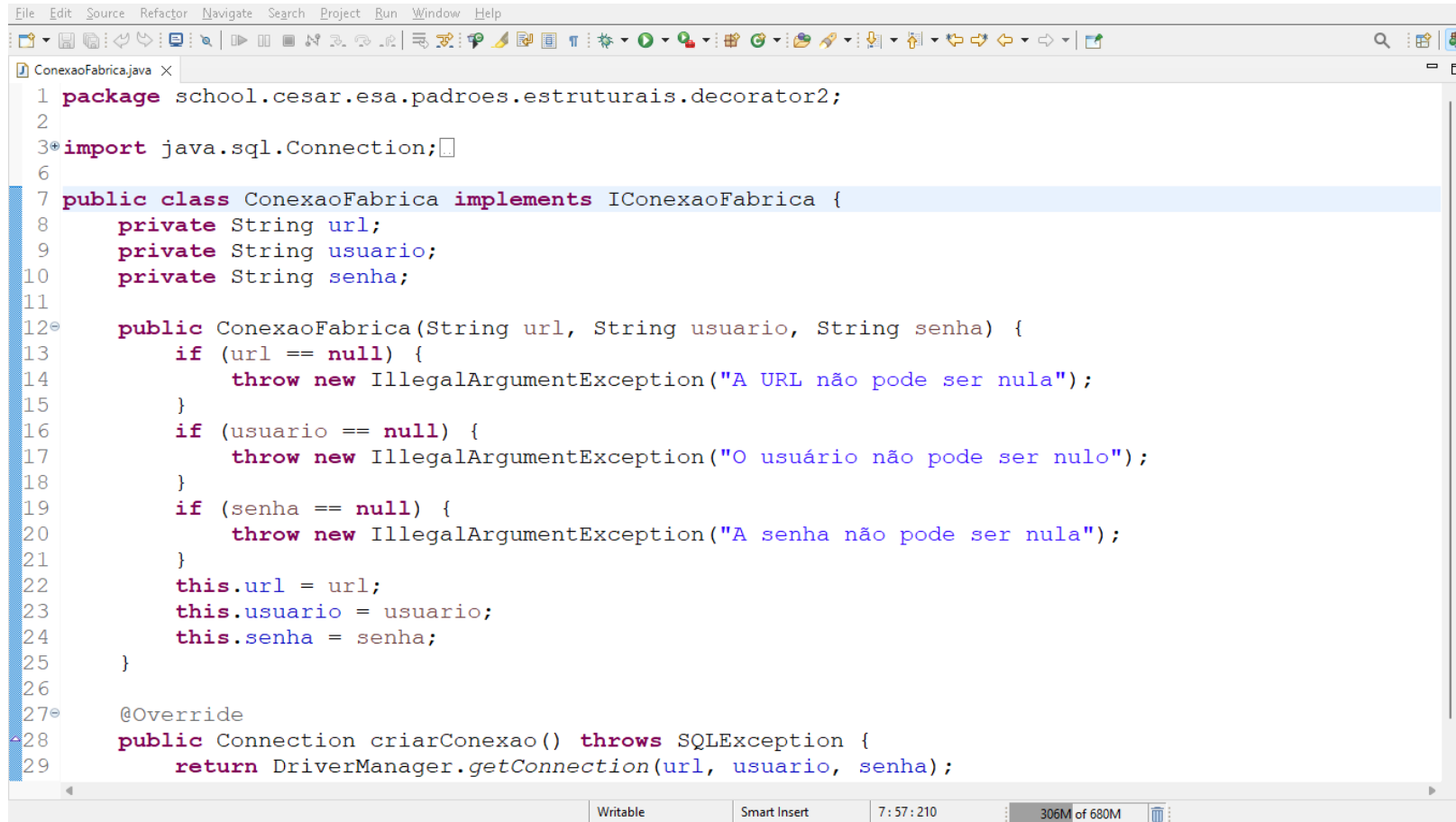
# Decorator



```
1 package school.cesar.esa.padroes.estruturais.decorator2;
2
3 import java.sql.Connection;
4
5
6 public interface IConexaoFabrica {
7     public Connection criarConexao() throws SQLException;
8 }
```

The screenshot shows an IDE window titled 'IConexaoFabrica.java'. The code defines a package 'school.cesar.esa.padroes.estruturais.decorator2', imports 'java.sql.Connection', and declares a public interface 'IConexaoFabrica' with a single method 'criarConexao()' that throws 'SQLException'.

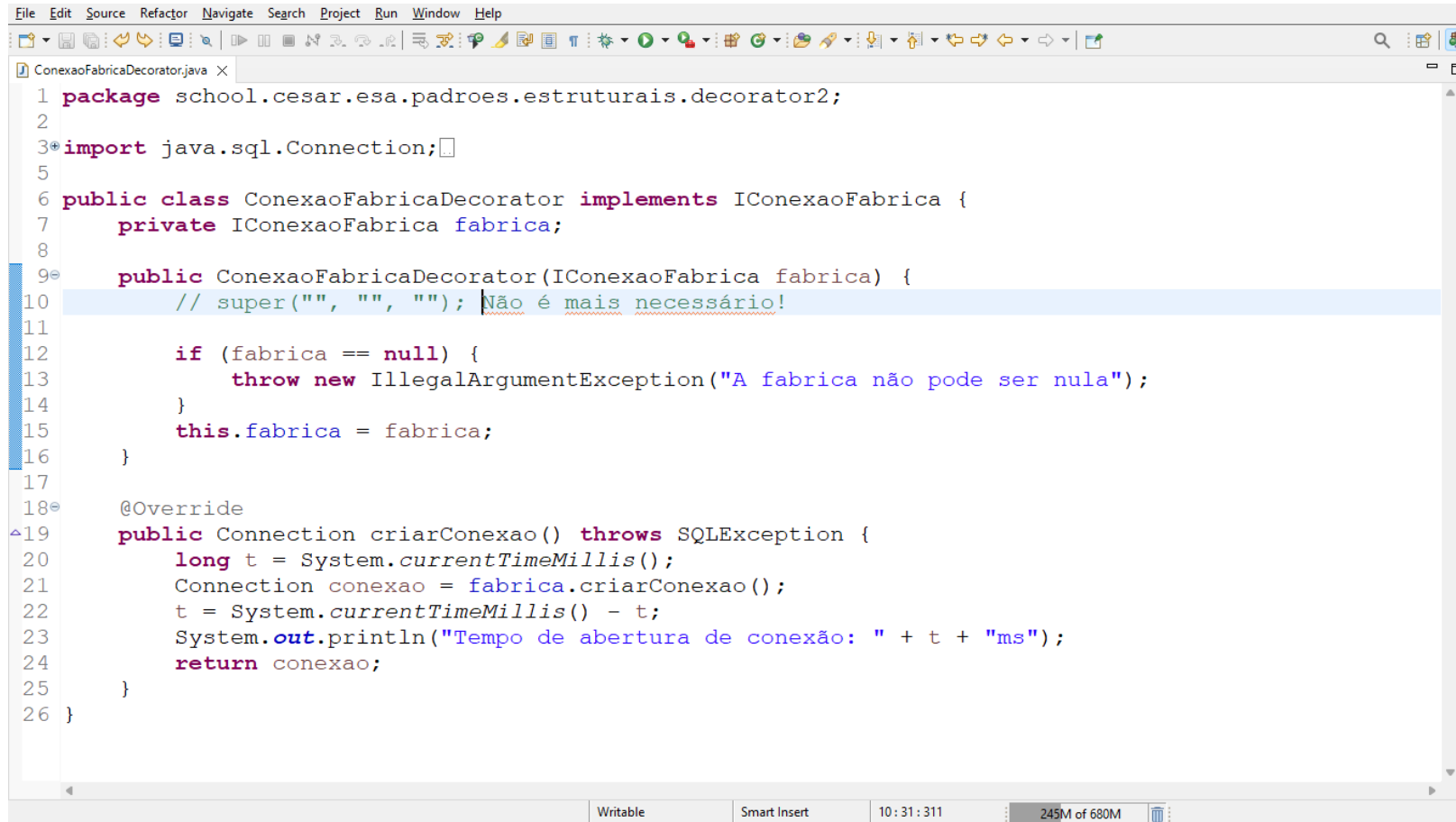
# Decorator



```
File Edit Source Refactor Navigate Search Project Run Window Help
ConexaoFabrica.java X
1 package school.cesar.esa.padroes.estruturais.decorator2;
2
3 import java.sql.Connection;
4
5
6
7 public class ConexaoFabrica implements IConexaoFabrica {
8     private String url;
9     private String usuario;
10    private String senha;
11
12    public ConexaoFabrica(String url, String usuario, String senha) {
13        if (url == null) {
14            throw new IllegalArgumentException("A URL não pode ser nula");
15        }
16        if (usuario == null) {
17            throw new IllegalArgumentException("O usuário não pode ser nulo");
18        }
19        if (senha == null) {
20            throw new IllegalArgumentException("A senha não pode ser nula");
21        }
22        this.url = url;
23        this.usuario = usuario;
24        this.senha = senha;
25    }
26
27    @Override
28    public Connection criarConexao() throws SQLException {
29        return DriverManager.getConnection(url, usuario, senha);
30    }
31}
```

Writable Smart Insert 7: 57: 210 306M of 680M

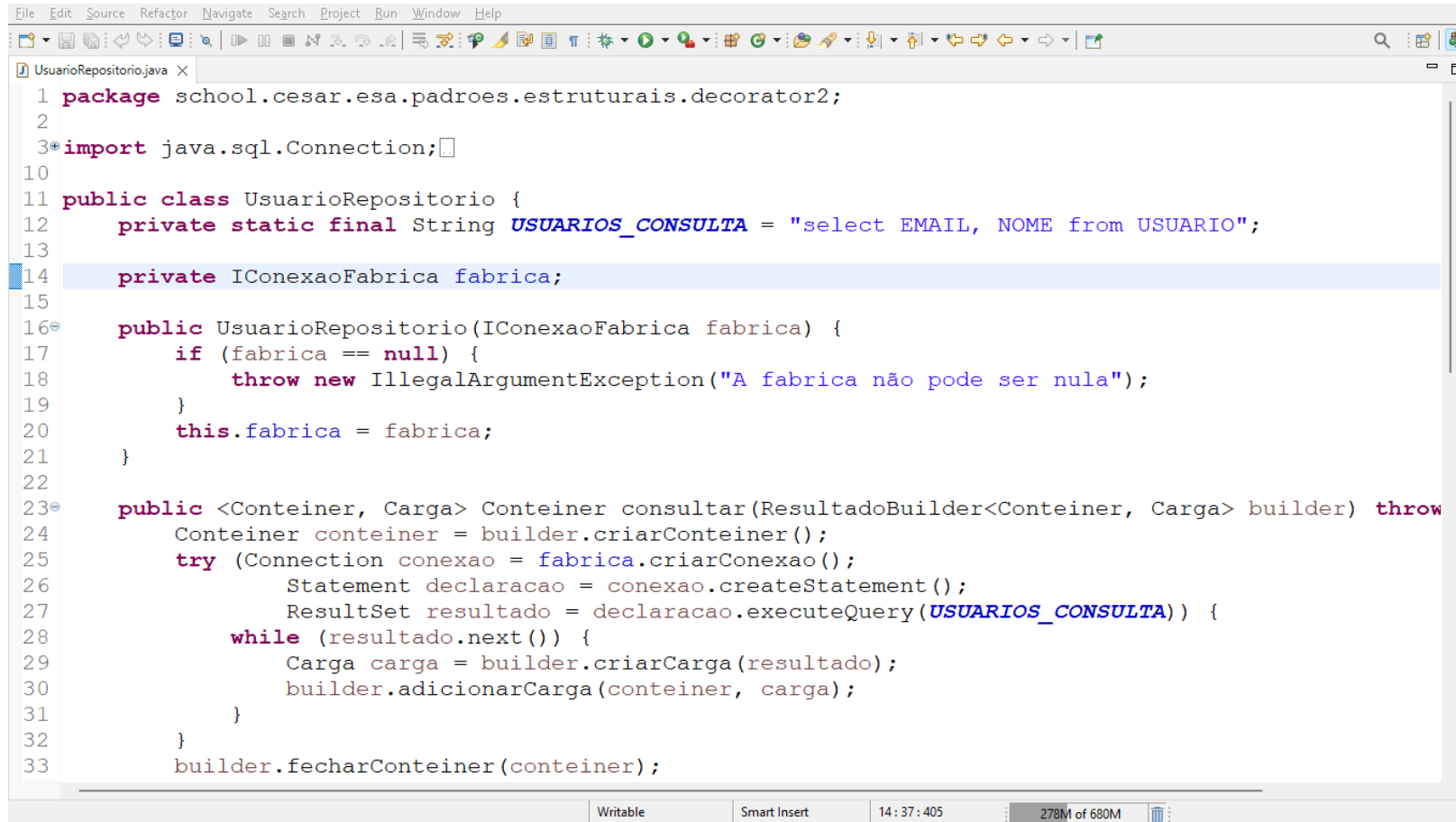
# Decorator



```
File Edit Source Refactor Navigate Search Project Run Window Help
ConexaoFabricaDecorator.java X
1 package school.cesar.esa.padroes.estruturais.decorator2;
2
3 import java.sql.Connection;
4
5
6 public class ConexaoFabricaDecorator implements IConexaoFabrica {
7     private IConexaoFabrica fabrica;
8
9     public ConexaoFabricaDecorator(IConexaoFabrica fabrica) {
10         // super("", "", ""); Não é mais necessário!
11
12         if (fabrica == null) {
13             throw new IllegalArgumentException("A fabrica não pode ser nula");
14         }
15         this.fabrica = fabrica;
16     }
17
18     @Override
19     public Connection criarConexao() throws SQLException {
20         long t = System.currentTimeMillis();
21         Connection conexao = fabrica.criarConexao();
22         t = System.currentTimeMillis() - t;
23         System.out.println("Tempo de abertura de conexão: " + t + "ms");
24         return conexao;
25     }
26 }
```

Writable Smart Insert 10:31:311 245M of 680M

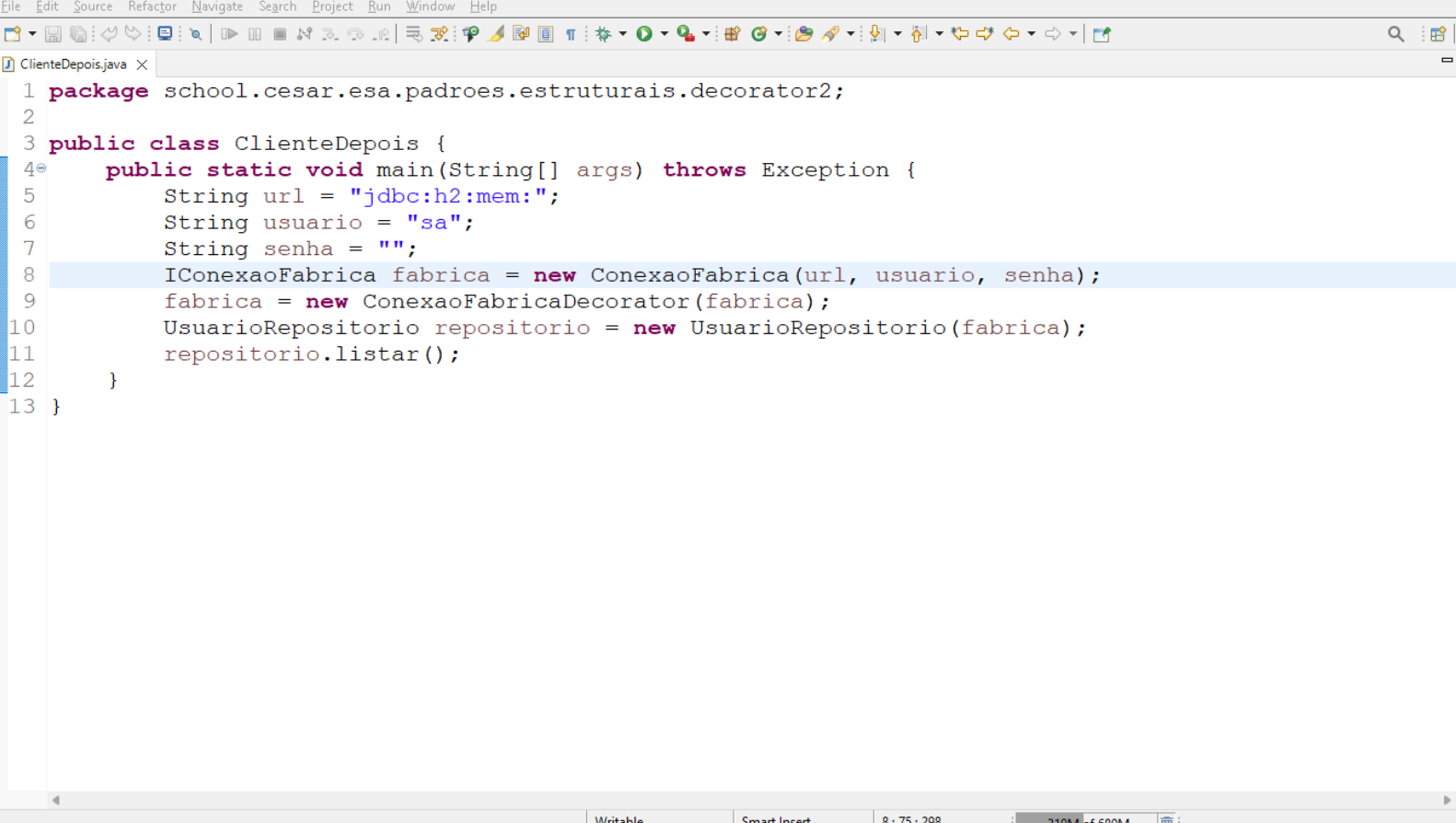
# Decorator



```
File Edit Source Refactor Navigate Search Project Run Window Help
UsuarioRepositorio.java X
1 package school.cesar.esa.padroes.estruturais.decorator2;
2
3 import java.sql.Connection;
10
11 public class UsuarioRepositorio {
12     private static final String USUARIOS_CONSULTA = "select EMAIL, NOME from USUARIO";
13
14     private IConexaoFabrica fabrica;
15
16     public UsuarioRepositorio(IConexaoFabrica fabrica) {
17         if (fabrica == null) {
18             throw new IllegalArgumentException("A fabrica não pode ser nula");
19         }
20         this.fabrica = fabrica;
21     }
22
23     public <Container, Carga> Container consultar(ResultadoBuilder<Container, Carga> builder) throw
24         Container container = builder.criarContainer();
25         try (Connection conexao = fabrica.criarConexao();
26             Statement declaracao = conexao.createStatement();
27             ResultSet resultado = declaracao.executeQuery(USUARIOS_CONSULTA)) {
28             while (resultado.next()) {
29                 Carga carga = builder.criarCarga(resultado);
30                 builder.adicionarCarga(container, carga);
31             }
32         }
33         builder.fecharContainer(container);
```

Writable Smart Insert 14 : 37 : 405 278M of 680M

# Decorator



```

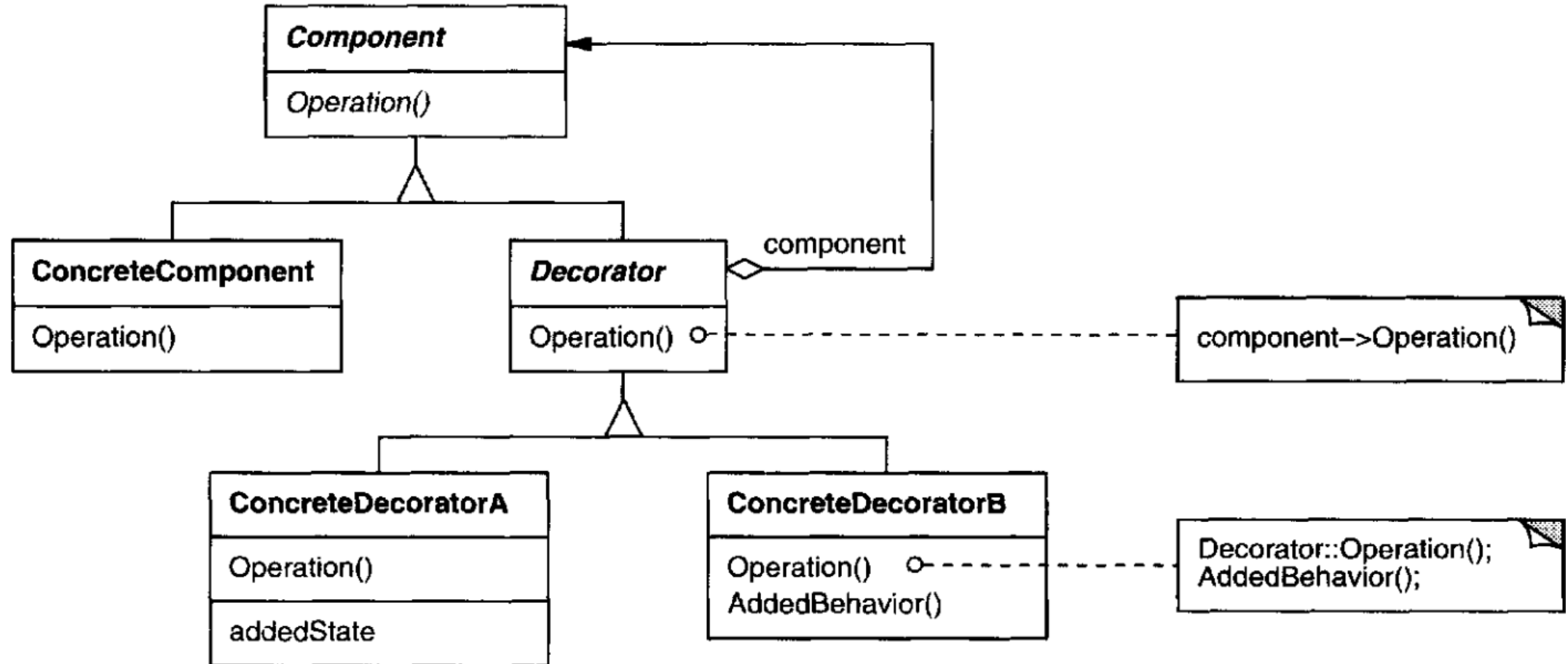
1 package school.cesar.esa.padroes.estruturais.decorator2;
2
3 public class ClienteDepois {
4     public static void main(String[] args) throws Exception {
5         String url = "jdbc:h2:mem:";
6         String usuario = "sa";
7         String senha = "";
8         IConexaoFabrica fabrica = new ConexaoFabrica(url, usuario, senha);
9         fabrica = new ConexaoFabricaDecorator(fabrica);
10        UsuarioRepositorio repositorio = new UsuarioRepositorio(fabrica);
11        repositorio.listar();
12    }
13 }

```

# Decorator

“Attach additional responsibilities to an object dynamically. Decorators provide a flexible alternative to subclassing for extending functionality.”

# Decorator



# Decorator

- SOLID
  - Responsabilidade única (Single responsibility)
  - Aberto-fechado (Open-closed)
  - Substituição de Liskov (Liskob substitution)
  - Segregação de interfaces (Interface segregation)
  - Inversão de dependências (Dependency inversion)
- Prefira composição à herança
- Demeter



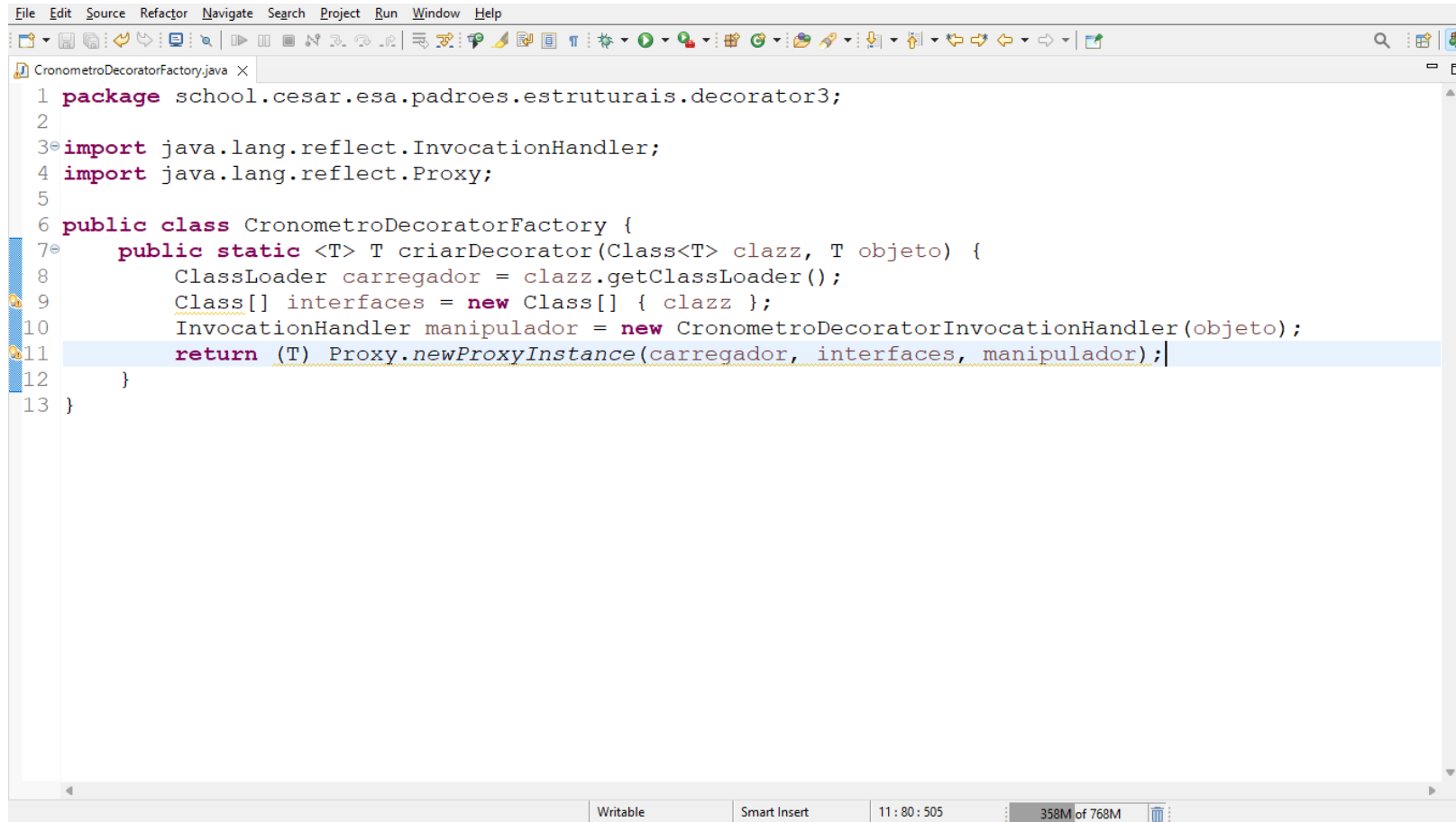
# Decorator

- Integridade conceitual
- (Alta) Coesão
- (Baixo) Acoplamento
- Ocultamento de informações

Decorator



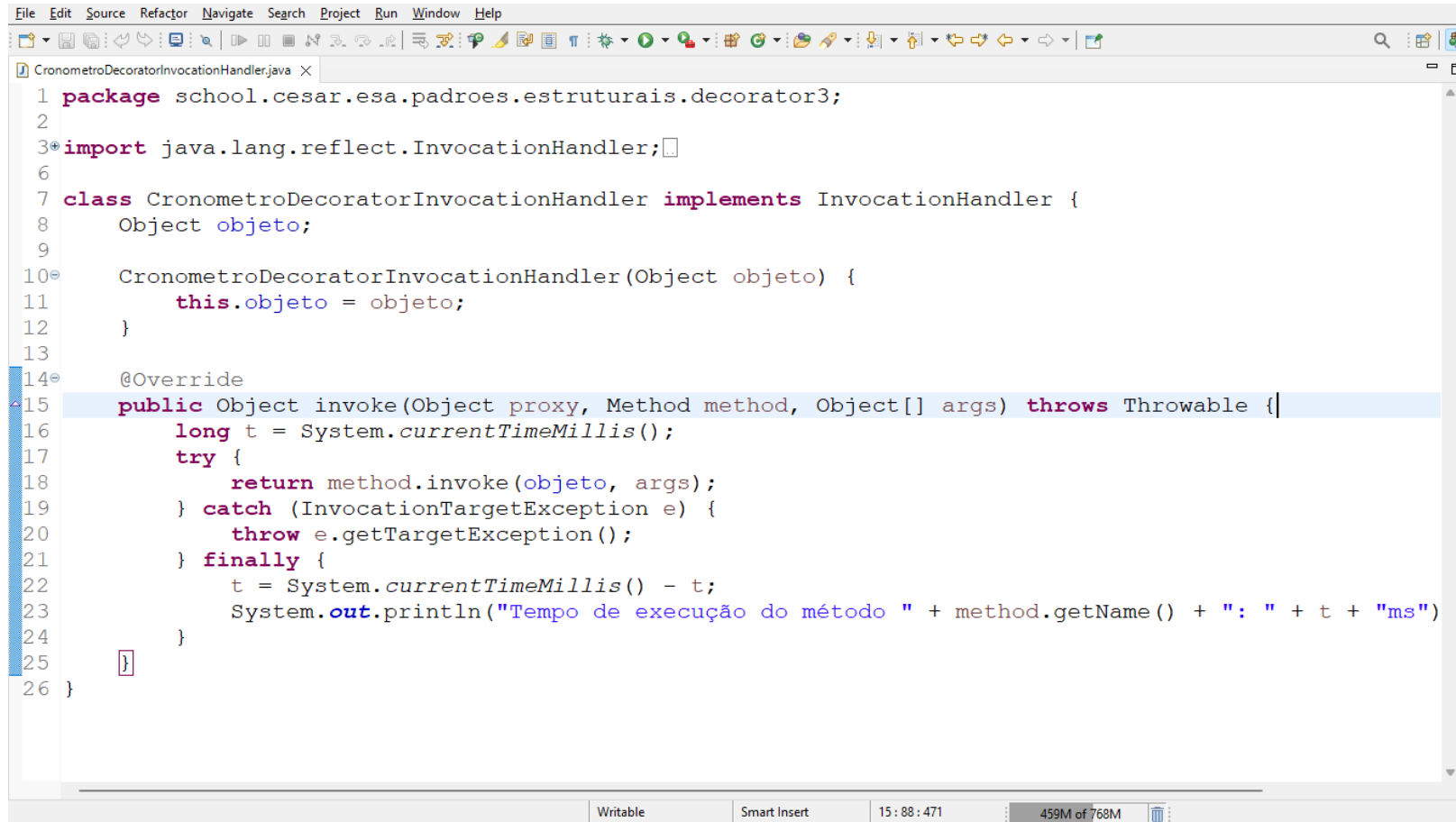
# Decorator



```
1 package school.cesar.esa.padroes.estruturais.decorator3;
2
3 import java.lang.reflect.InvocationHandler;
4 import java.lang.reflect.Proxy;
5
6 public class CronometroDecoratorFactory {
7     public static <T> T criarDecorator(Class<T> clazz, T objeto) {
8         ClassLoader carregador = clazz.getClassLoader();
9         Class[] interfaces = new Class[] { clazz };
10        InvocationHandler manipulador = new CronometroDecoratorInvocationHandler(objeto);
11        return (T) Proxy.newProxyInstance(carregador, interfaces, manipulador);
12    }
13 }
```

Writable Smart Insert 11 : 80 : 505 358M of 768M

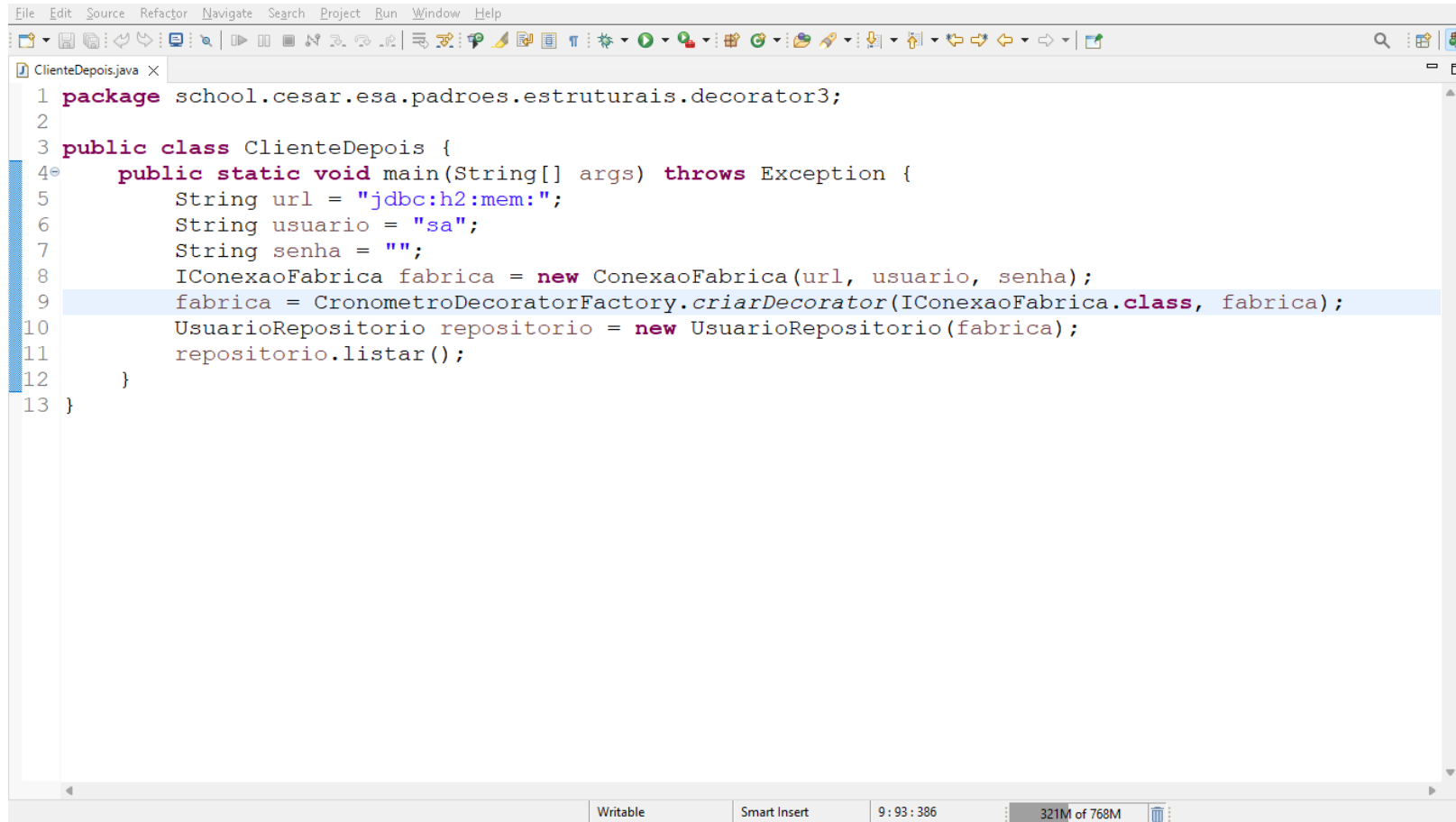
# Decorator



```
File Edit Source Refactor Navigate Search Project Run Window Help
CronometroDecoratorInvocationHandler.java x
1 package school.cesar.esa.padroes.estruturais.decorator3;
2
3 import java.lang.reflect.InvocationHandler;
4
5
6
7 class CronometroDecoratorInvocationHandler implements InvocationHandler {
8     Object objeto;
9
10    CronometroDecoratorInvocationHandler(Object objeto) {
11        this.objeto = objeto;
12    }
13
14    @Override
15    public Object invoke(Object proxy, Method method, Object[] args) throws Throwable {
16        long t = System.currentTimeMillis();
17        try {
18            return method.invoke(objeto, args);
19        } catch (InvocationTargetException e) {
20            throw e.getTargetException();
21        } finally {
22            t = System.currentTimeMillis() - t;
23            System.out.println("Tempo de execução do método " + method.getName() + ": " + t + "ms")
24        }
25    }
26 }
```

Writable Smart Insert 15 : 88 : 471 459M of 768M

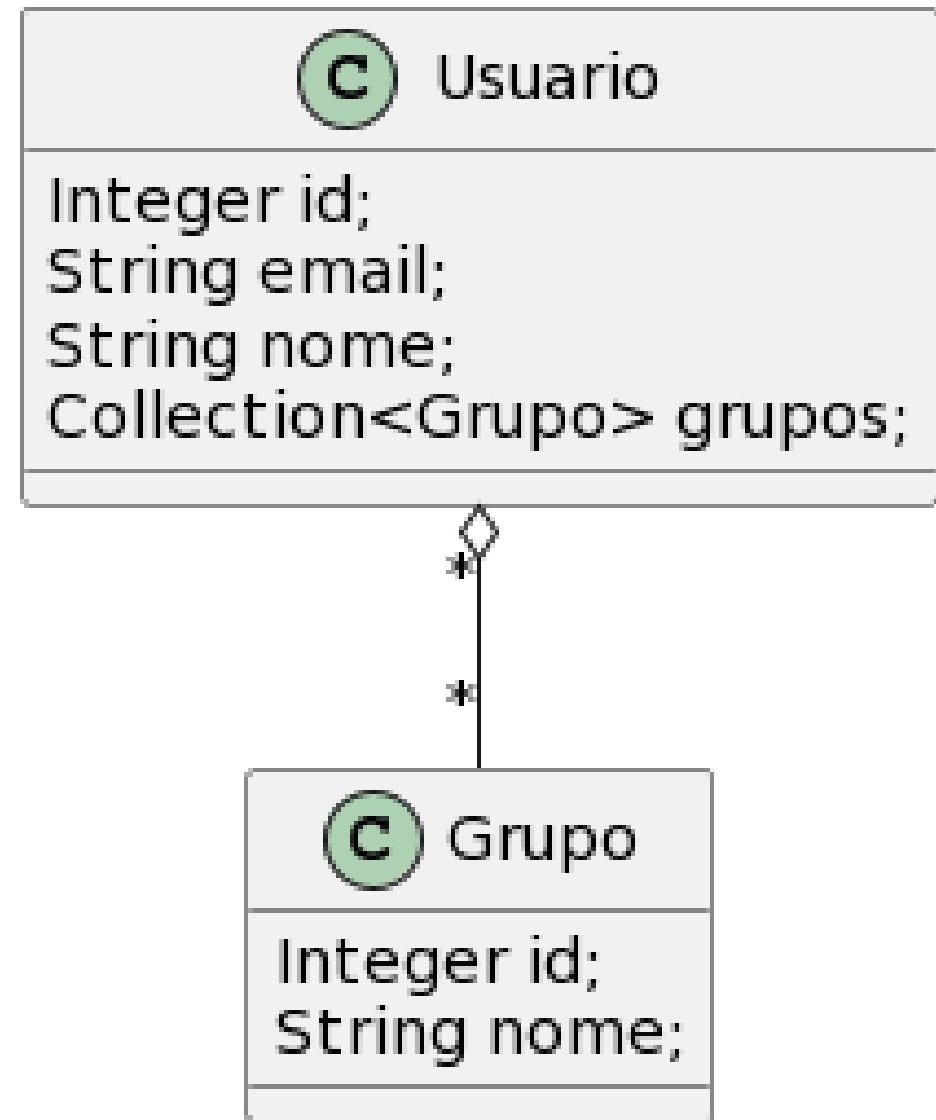
# Decorator



```
1 package school.cesar.esa.padroes.estruturais.decorator3;
2
3 public class ClienteDepois {
4     public static void main(String[] args) throws Exception {
5         String url = "jdbc:h2:mem:";
6         String usuario = "sa";
7         String senha = "";
8         IConexaoFabrica fabrica = new ConexaoFabrica(url, usuario, senha);
9         fabrica = CronometroDecoratorFactory.criarDecorator(IConexaoFabrica.class, fabrica);
10        UsuarioRepositorio repositorio = new UsuarioRepositorio(fabrica);
11        repositorio.listar();
12    }
13 }
```

Proxy

Proxy

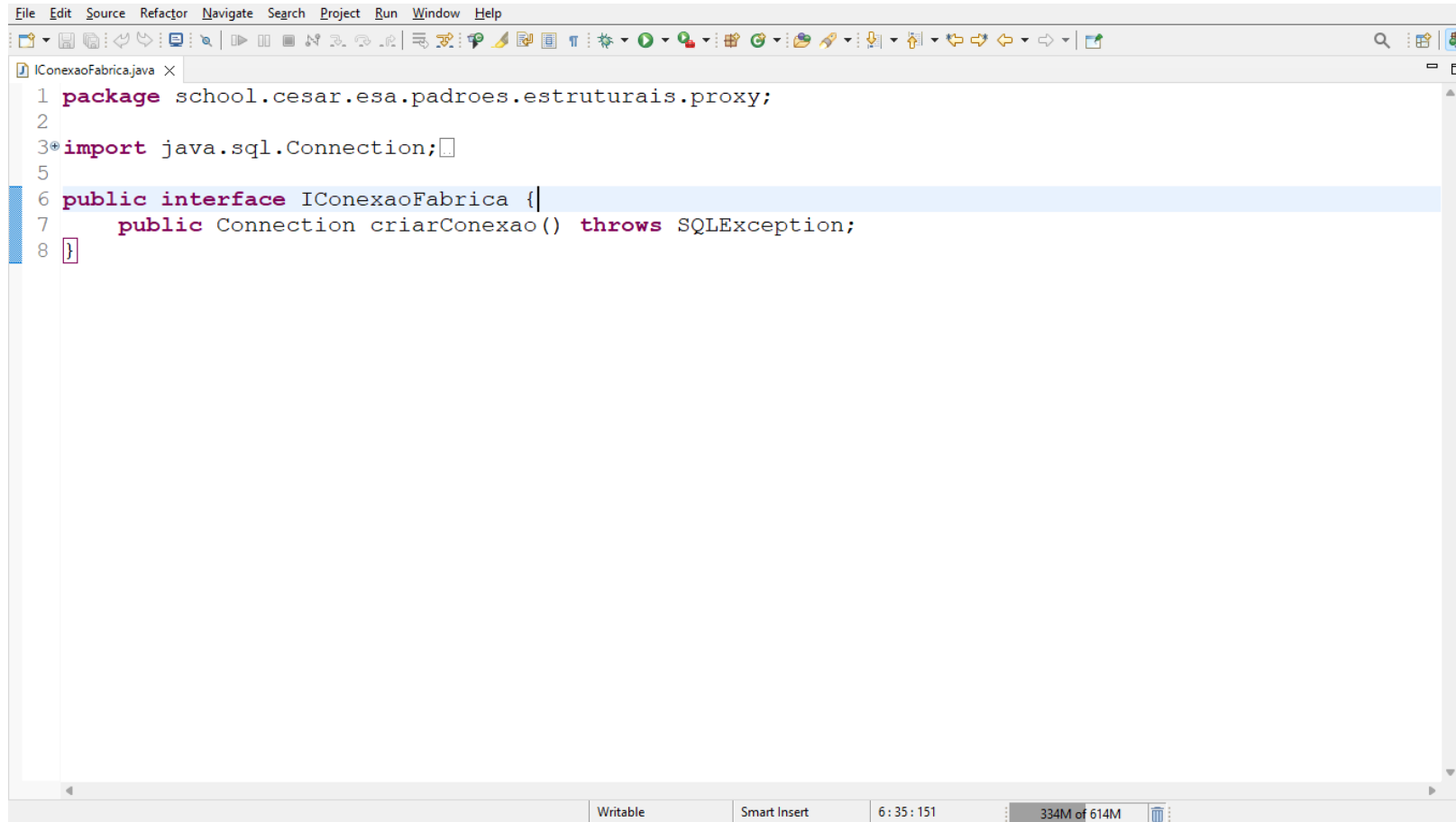


# Proxy





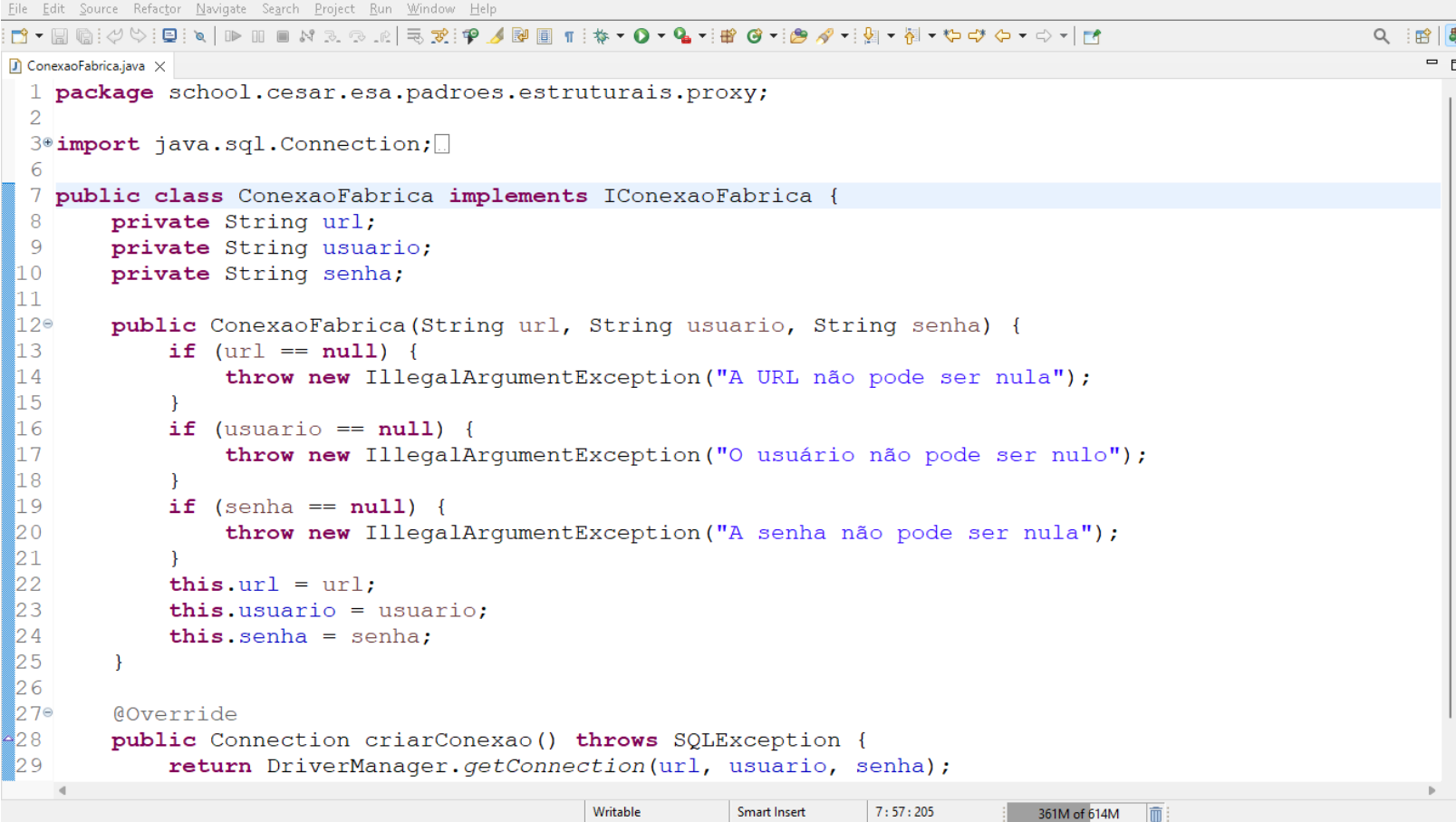
# Proxy



The image shows a screenshot of an IDE window with a single file named 'IConexaoFabrica.java'. The code defines a package, imports a class, and declares a public interface with one method. The interface is highlighted with a blue background. The IDE's status bar at the bottom shows 'Writable', 'Smart Insert', '6 : 35 : 151', and '334M of 614M'.

```
1 package school.cesar.esa.padroes.estruturais.proxy;
2
3 import java.sql.Connection;
4
5
6 public interface IConexaoFabrica {
7     public Connection criarConexao() throws SQLException;
8 }
```

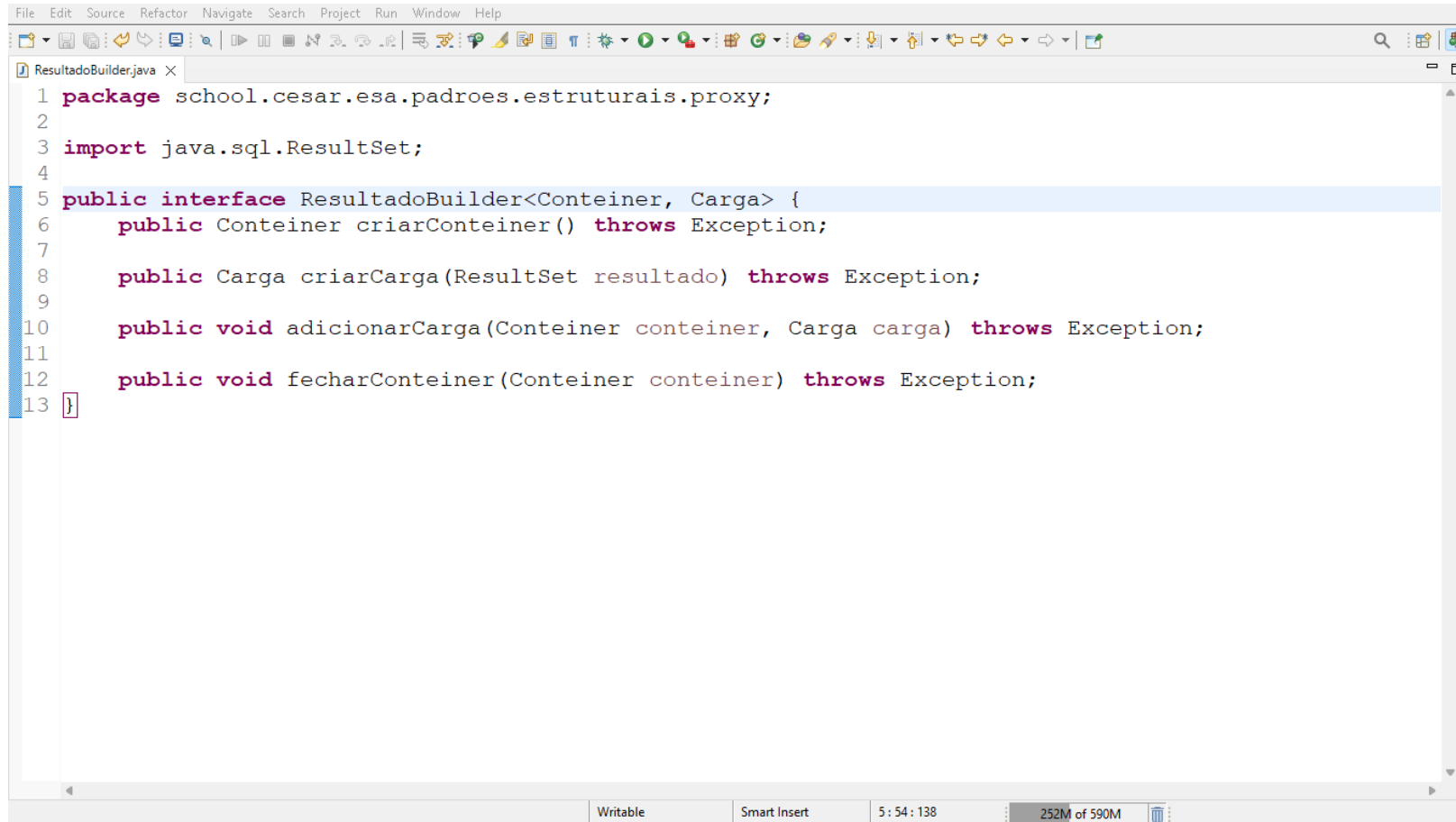
# Proxy



```
File Edit Source Refactor Navigate Search Project Run Window Help
ConexaoFabrica.java X
1 package school.cesar.esa.padroes.estruturais.proxy;
2
3 import java.sql.Connection;
4
5
6
7 public class ConexaoFabrica implements IConexaoFabrica {
8     private String url;
9     private String usuario;
10    private String senha;
11
12    public ConexaoFabrica(String url, String usuario, String senha) {
13        if (url == null) {
14            throw new IllegalArgumentException("A URL não pode ser nula");
15        }
16        if (usuario == null) {
17            throw new IllegalArgumentException("O usuário não pode ser nulo");
18        }
19        if (senha == null) {
20            throw new IllegalArgumentException("A senha não pode ser nula");
21        }
22        this.url = url;
23        this.usuario = usuario;
24        this.senha = senha;
25    }
26
27    @Override
28    public Connection criarConexao() throws SQLException {
29        return DriverManager.getConnection(url, usuario, senha);
30    }
31}
```

Writable Smart Insert 7:57:205 361M of 614M

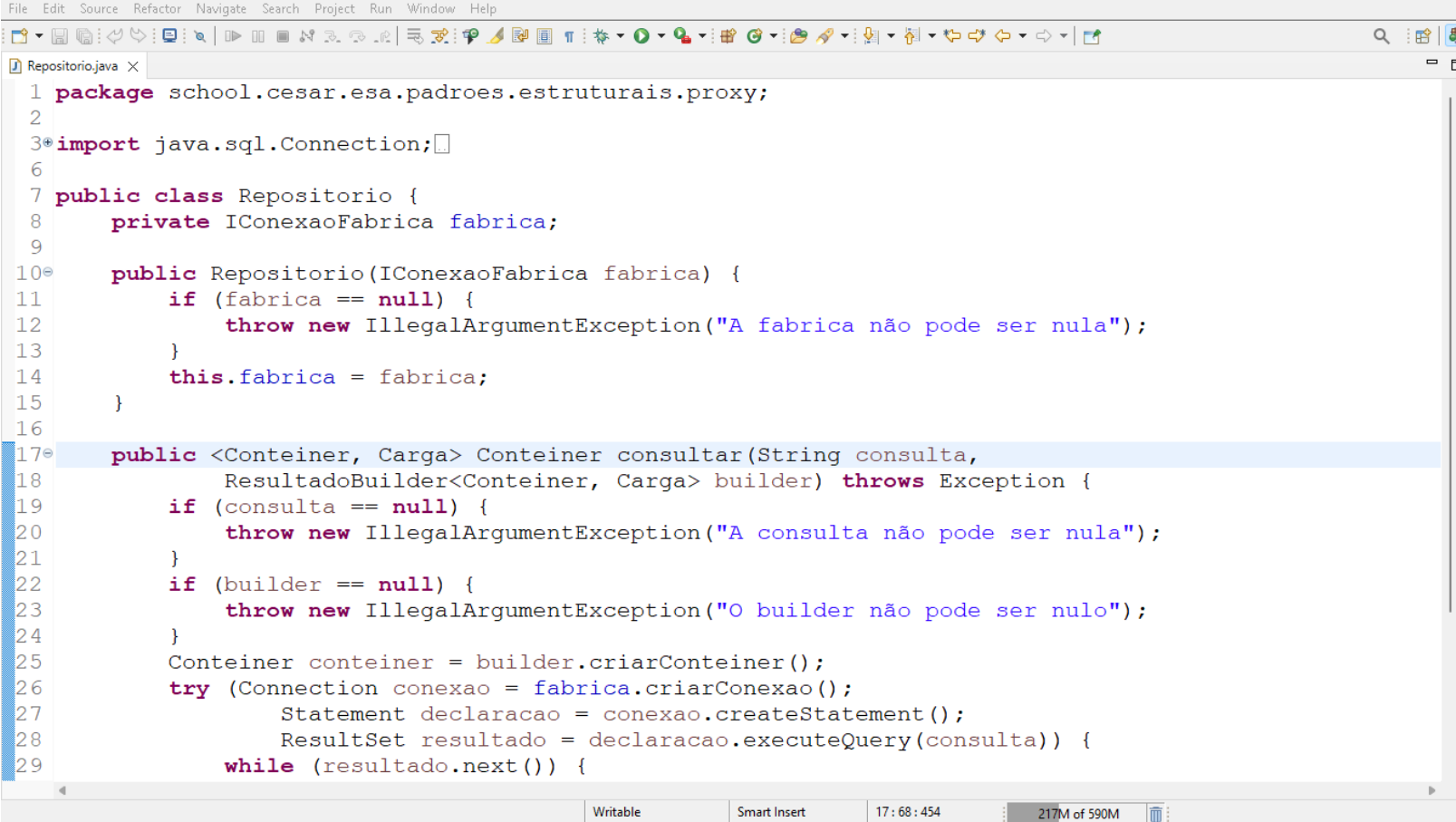
# Proxy



```
File Edit Source Refactor Navigate Search Project Run Window Help
ResultadoBuilder.java x
1 package school.cesar.esa.padroes.estruturais.proxy;
2
3 import java.sql.ResultSet;
4
5 public interface ResultadoBuilder<Container, Carga> {
6     public Container criarContainer() throws Exception;
7
8     public Carga criarCarga(ResultSet resultado) throws Exception;
9
10    public void adicionarCarga(Container container, Carga carga) throws Exception;
11
12    public void fecharContainer(Container container) throws Exception;
13 }
```

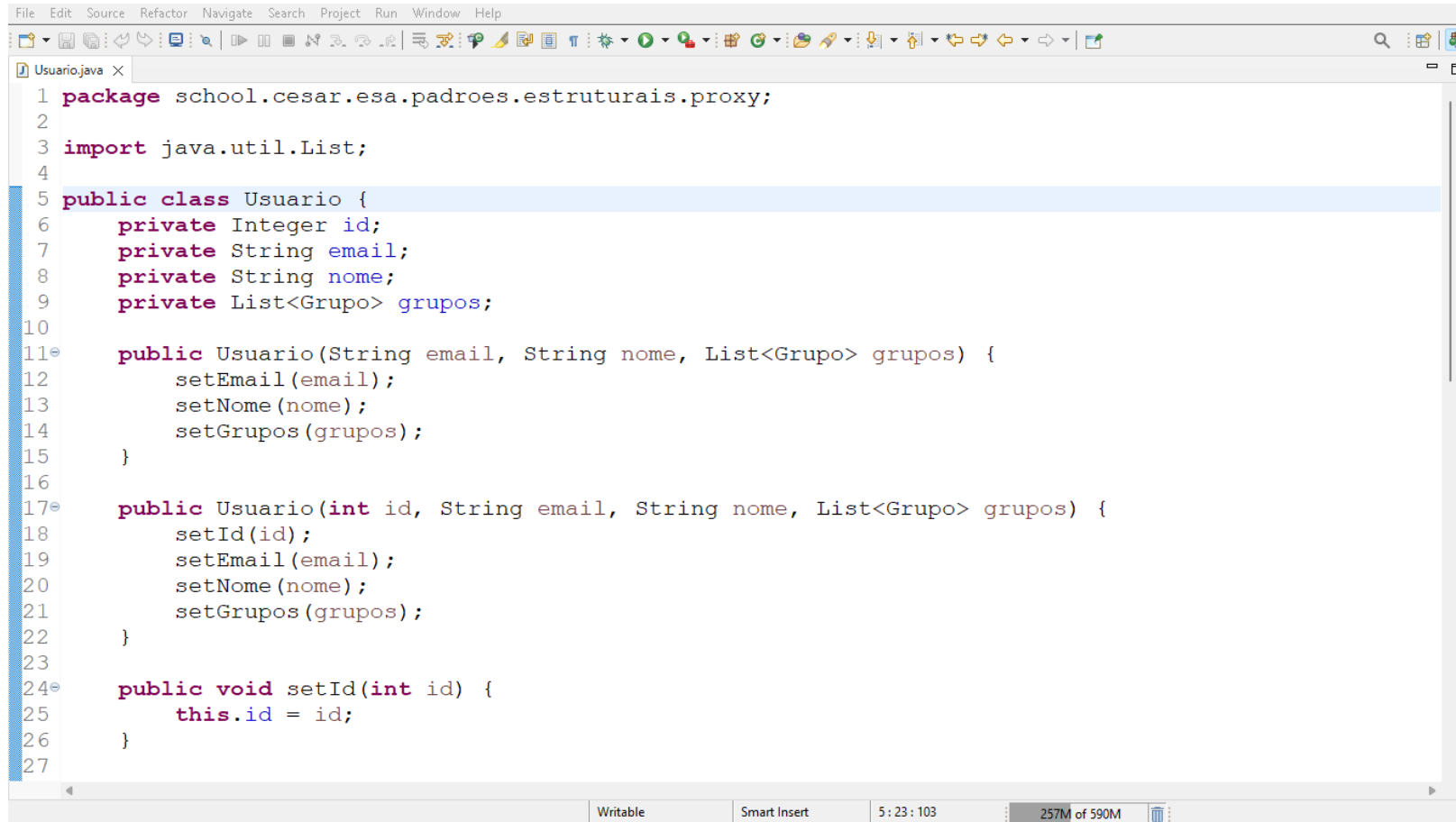
Writable Smart Insert 5 : 54 : 138 252M of 590M

# Proxy



```
1 package school.cesar.esa.padroes.estruturais.proxy;
2
3 import java.sql.Connection;
4
5
6
7 public class Repositorio {
8     private IConexaoFabrica fabrica;
9
10    public Repositorio(IConexaoFabrica fabrica) {
11        if (fabrica == null) {
12            throw new IllegalArgumentException("A fabrica não pode ser nula");
13        }
14        this.fabrica = fabrica;
15    }
16
17    public <Container, Carga> Container consultar(String consulta,
18        ResultadoBuilder<Container, Carga> builder) throws Exception {
19        if (consulta == null) {
20            throw new IllegalArgumentException("A consulta não pode ser nula");
21        }
22        if (builder == null) {
23            throw new IllegalArgumentException("O builder não pode ser nulo");
24        }
25        Container container = builder.criarContainer();
26        try (Connection conexao = fabrica.criarConexao();
27            Statement declaracao = conexao.createStatement();
28            ResultSet resultado = declaracao.executeQuery(consulta)) {
29            while (resultado.next()) {
```

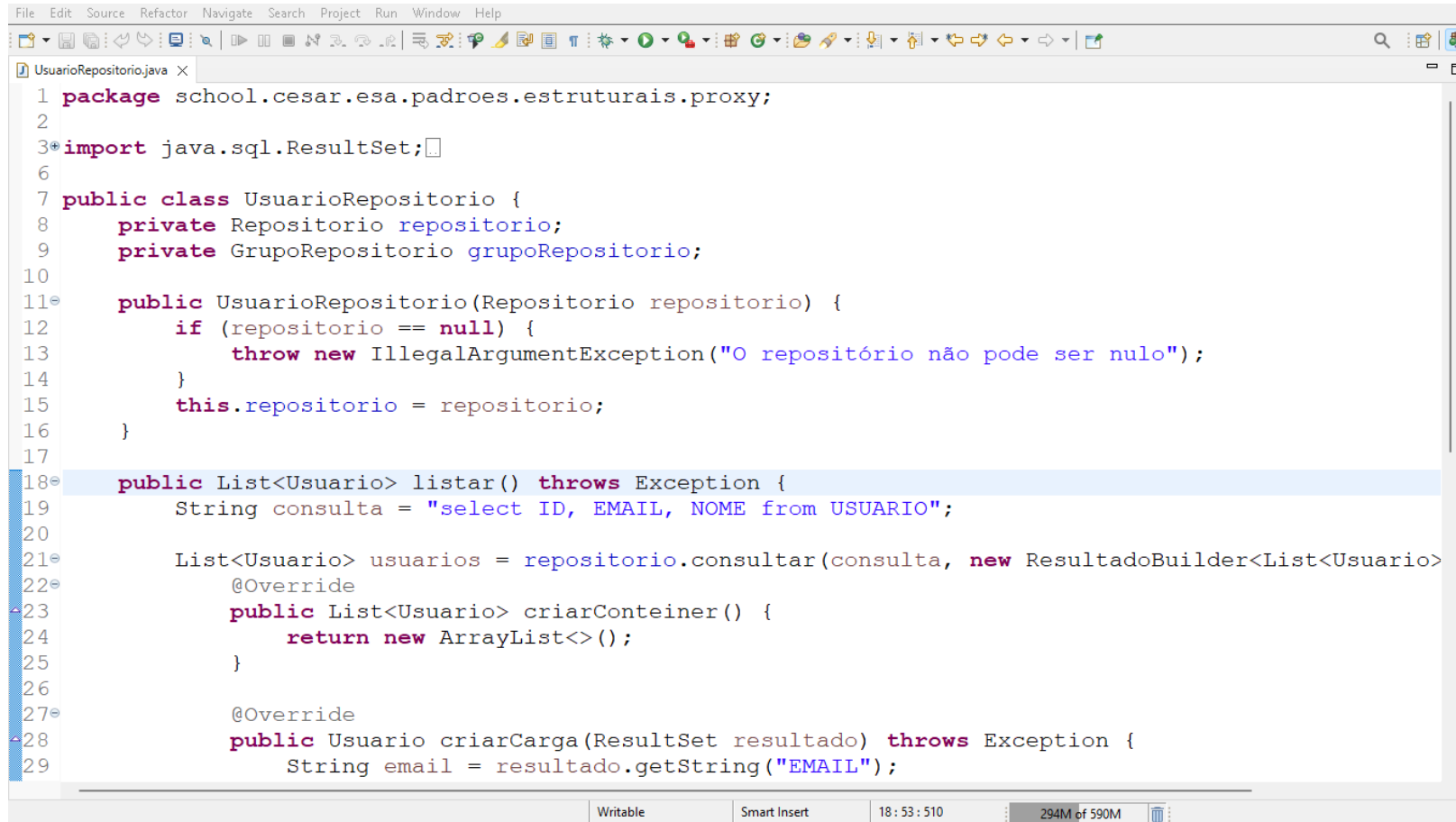
# Proxy



```
File Edit Source Refactor Navigate Search Project Run Window Help
Usuario.java x
1 package school.cesar.esa.padroes.estruturais.proxy;
2
3 import java.util.List;
4
5 public class Usuario {
6     private Integer id;
7     private String email;
8     private String nome;
9     private List<Grupo> grupos;
10
11     public Usuario(String email, String nome, List<Grupo> grupos) {
12         setEmail(email);
13         setNome(nome);
14         setGrupos(grupos);
15     }
16
17     public Usuario(int id, String email, String nome, List<Grupo> grupos) {
18         setId(id);
19         setEmail(email);
20         setNome(nome);
21         setGrupos(grupos);
22     }
23
24     public void setId(int id) {
25         this.id = id;
26     }
27
```

Writable Smart Insert 5 : 23 : 103 257M of 590M

# Proxy



```
1 package school.cesar.esa.padroes.estruturais.proxy;
2
3 import java.sql.ResultSet;
4
5
6
7 public class UsuarioRepositorio {
8     private Repositorio repositorio;
9     private GrupoRepositorio grupoRepositorio;
10
11     public UsuarioRepositorio(Repositorio repositorio) {
12         if (repositorio == null) {
13             throw new IllegalArgumentException("O repositório não pode ser nulo");
14         }
15         this.repositorio = repositorio;
16     }
17
18     public List<Usuario> listar() throws Exception {
19         String consulta = "select ID, EMAIL, NOME from USUARIO";
20
21         List<Usuario> usuarios = repositorio.consultar(consulta, new ResultadoBuilder<List<Usuario>>() {
22             @Override
23             public List<Usuario> criarContainer() {
24                 return new ArrayList<>();
25             }
26
27             @Override
28             public Usuario criarCarga(ResultSet resultado) throws Exception {
29                 String email = resultado.getString("EMAIL");
```

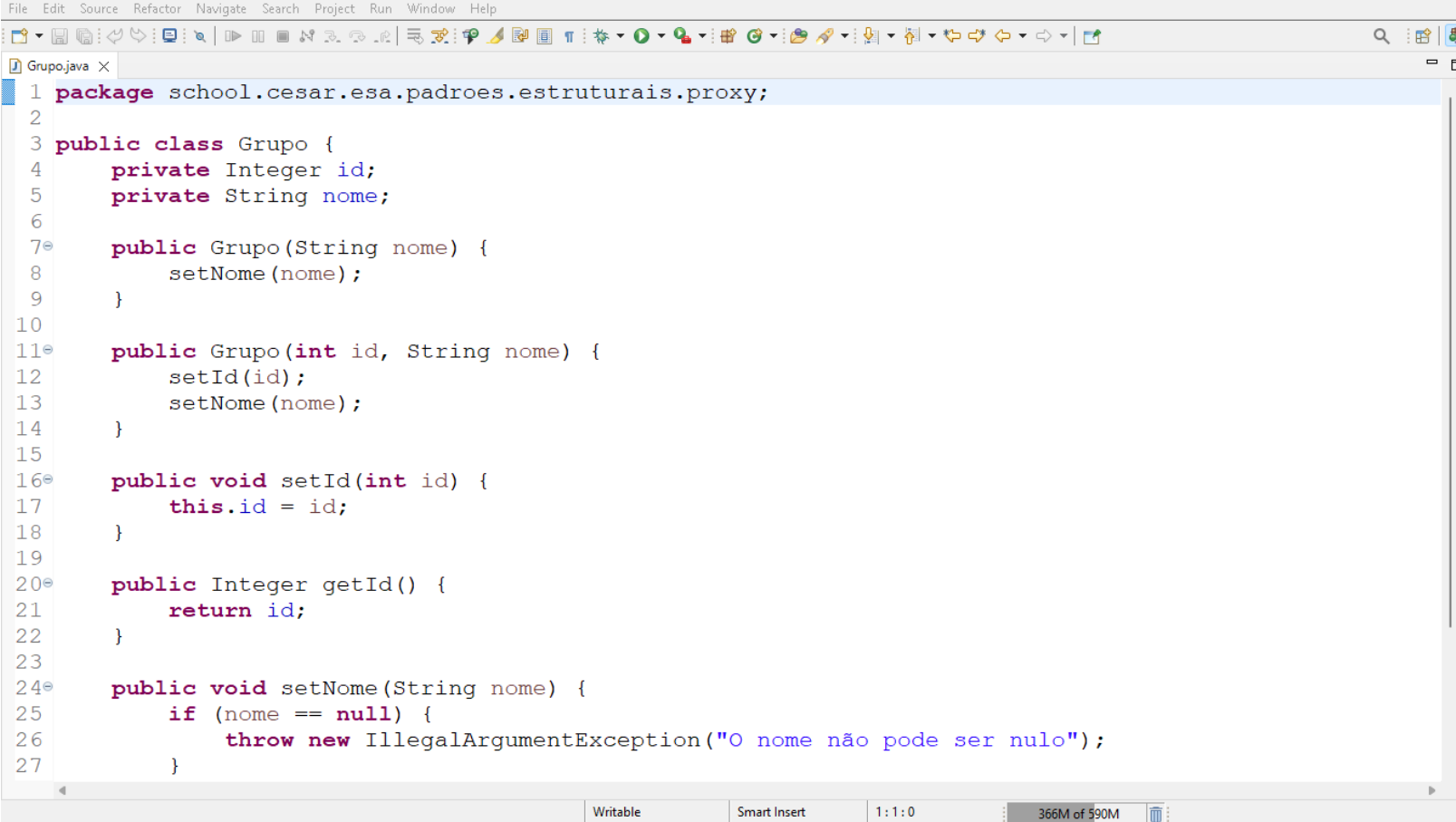
# Proxy

```

27 @Override
28 public Usuario criarCarga(ResultSet resultado) throws Exception {
29     String email = resultado.getString("EMAIL");
30     String nome = resultado.getString("NOME");
31     List<Grupo> grupos = new ArrayList<>();
32     return new Usuario(email, nome, grupos);
33 }
34
35 @Override
36 public void adicionarCarga(List<Usuario> container, Usuario carga) {
37     container.add(carga);
38 }
39
40 @Override
41 public void fecharContainer(List<Usuario> container) throws Exception {
42 }
43
44 });
45
46 for (Usuario usuario : usuarios) {
47     int id = usuario.getId();
48     List<Grupo> grupos = grupoRepositorio.consultarGrupos(id);
49     usuario.setGrupos(grupos);
50 }
51
52 return usuarios;
53 }

```

# Proxy

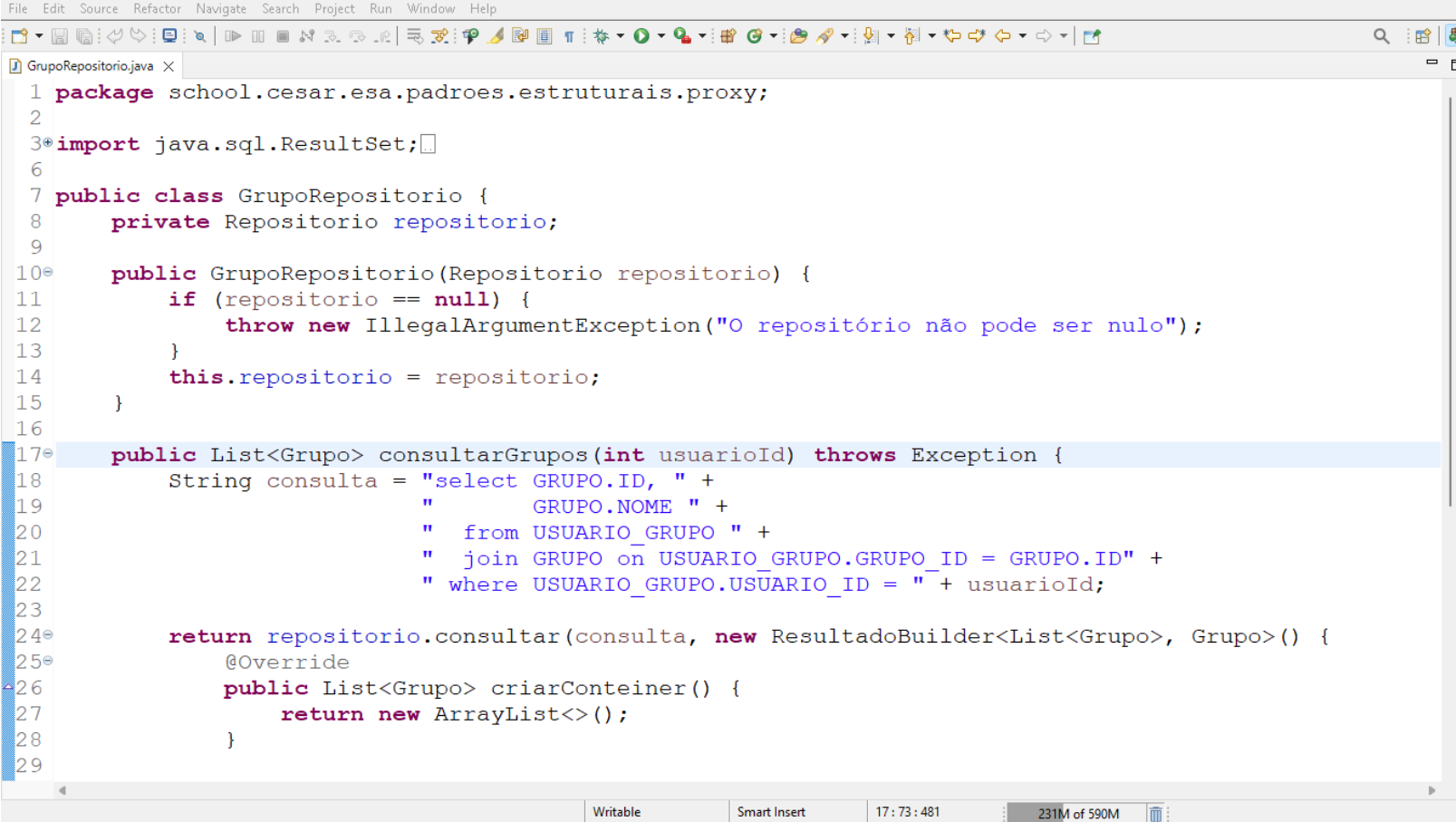


```
File Edit Source Refactor Navigate Search Project Run Window Help
Grupo.java x
1 package school.cesar.esa.padroes.estruturais.proxy;
2
3 public class Grupo {
4     private Integer id;
5     private String nome;
6
7     public Grupo(String nome) {
8         setNome(nome);
9     }
10
11    public Grupo(int id, String nome) {
12        setId(id);
13        setNome(nome);
14    }
15
16    public void setId(int id) {
17        this.id = id;
18    }
19
20    public Integer getId() {
21        return id;
22    }
23
24    public void setNome(String nome) {
25        if (nome == null) {
26            throw new IllegalArgumentException("O nome não pode ser nulo");
27        }
28    }
29 }
```

Writable Smart Insert 1:1:0 366M of 590M



# Proxy

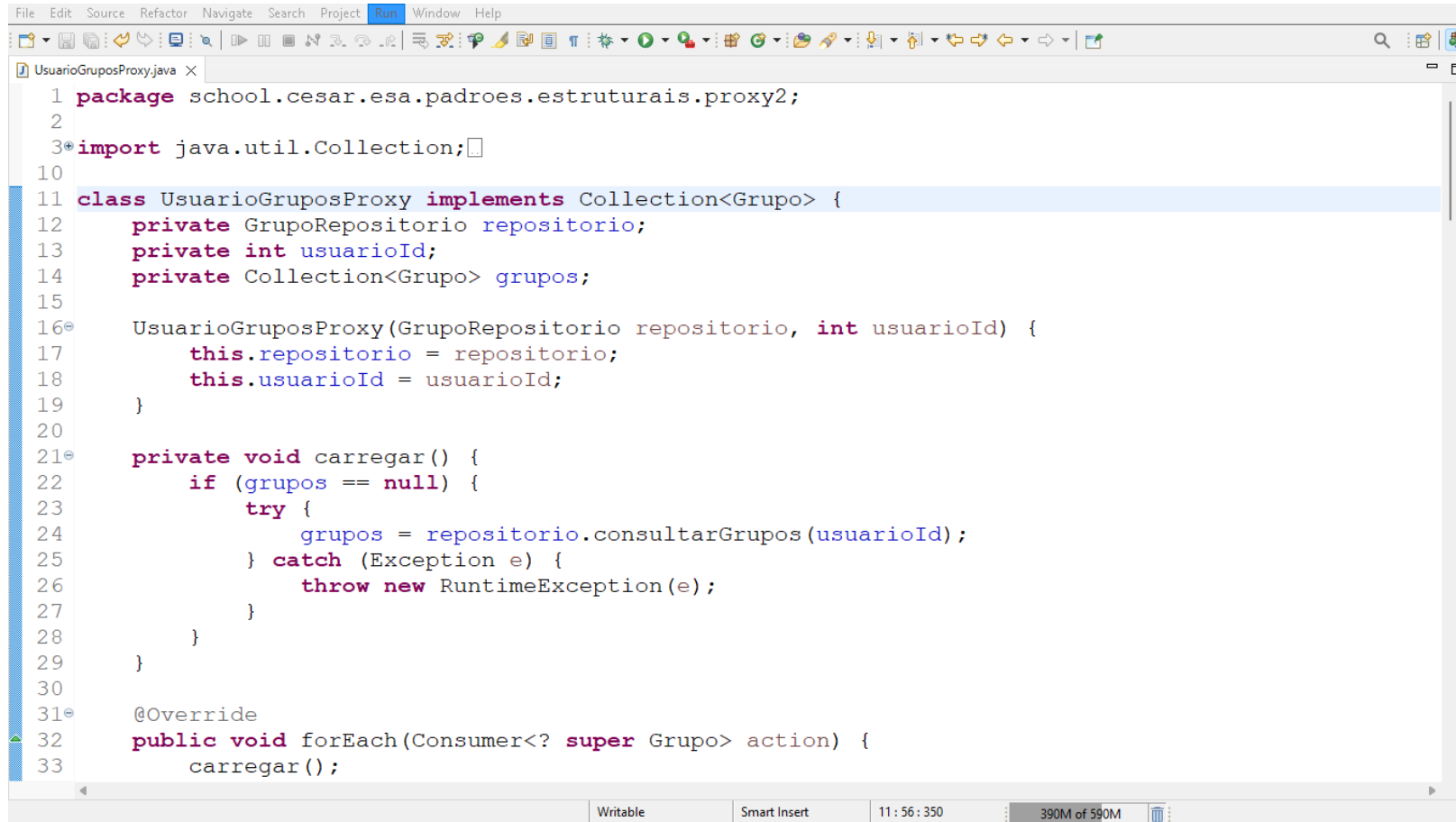


```
1 package school.cesar.esa.padroes.estruturais.proxy;
2
3 import java.sql.ResultSet;
4
5
6
7 public class GrupoRepositorio {
8     private Repositorio repositorio;
9
10    public GrupoRepositorio(Repositorio repositorio) {
11        if (repositorio == null) {
12            throw new IllegalArgumentException("O repositório não pode ser nulo");
13        }
14        this.repositorio = repositorio;
15    }
16
17    public List<Grupo> consultarGrupos(int usuarioId) throws Exception {
18        String consulta = "select GRUPO.ID, " +
19                          "      GRUPO.NOME " +
20                          " from USUARIO_GRUPO " +
21                          " join GRUPO on USUARIO_GRUPO.GRUPO_ID = GRUPO.ID" +
22                          " where USUARIO_GRUPO.USUARIO_ID = " + usuarioId;
23
24        return repositorio.consultar(consulta, new ResultadoBuilder<List<Grupo>, Grupo>() {
25            @Override
26            public List<Grupo> criarContainer() {
27                return new ArrayList<>();
28            }
29        });
30    }
31}
```

# Proxy



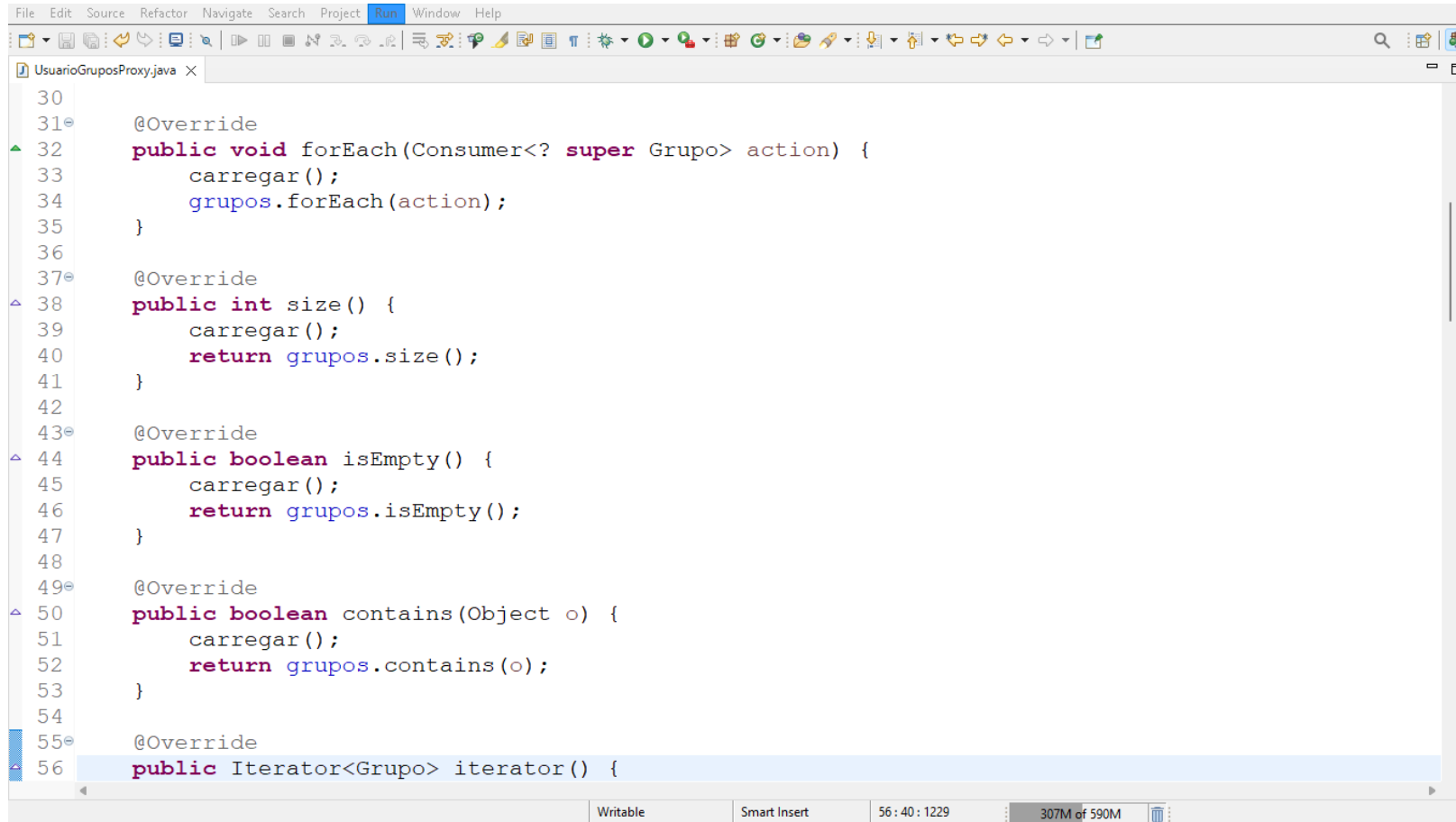
# Proxy



```
File Edit Source Refactor Navigate Search Project Run Window Help
UsuarioGruposProxy.java x
1 package school.cesar.esa.padroes.estruturais.proxy2;
2
3 import java.util.Collection;
4
10
11 class UsuarioGruposProxy implements Collection<Grupo> {
12     private GrupoRepositorio repositorio;
13     private int usuarioId;
14     private Collection<Grupo> grupos;
15
16     UsuarioGruposProxy(GupoRepositorio repositorio, int usuarioId) {
17         this.repositorio = repositorio;
18         this.usuarioId = usuarioId;
19     }
20
21     private void carregar() {
22         if (grupos == null) {
23             try {
24                 grupos = repositorio.consultarGrupos(usuarioId);
25             } catch (Exception e) {
26                 throw new RuntimeException(e);
27             }
28         }
29     }
30
31     @Override
32     public void forEach(Consumer<? super Grupo> action) {
33         carregar();
```

Writable Smart Insert 11:56:35 390M of 590M

# Proxy



The screenshot shows an IDE window titled 'UsuarioGruposProxy.java'. The code implements a Proxy pattern for a 'Grupo' interface. The 'UsuarioGruposProxy' class has several methods that delegate calls to a 'Grupo' object (referred to as 'grupos'). Each method call is preceded by a 'carregar()' method call, likely for lazy loading. The methods shown are: 'forEach', 'size', 'isEmpty', 'contains', and 'iterator'. The 'iterator' method is currently selected and highlighted in blue.

```
30
31 @Override
32 public void forEach(Consumer<? super Grupo> action) {
33     carregar();
34     grupos.forEach(action);
35 }
36
37 @Override
38 public int size() {
39     carregar();
40     return grupos.size();
41 }
42
43 @Override
44 public boolean isEmpty() {
45     carregar();
46     return grupos.isEmpty();
47 }
48
49 @Override
50 public boolean contains(Object o) {
51     carregar();
52     return grupos.contains(o);
53 }
54
55 @Override
56 public Iterator<Grupo> iterator() {
```

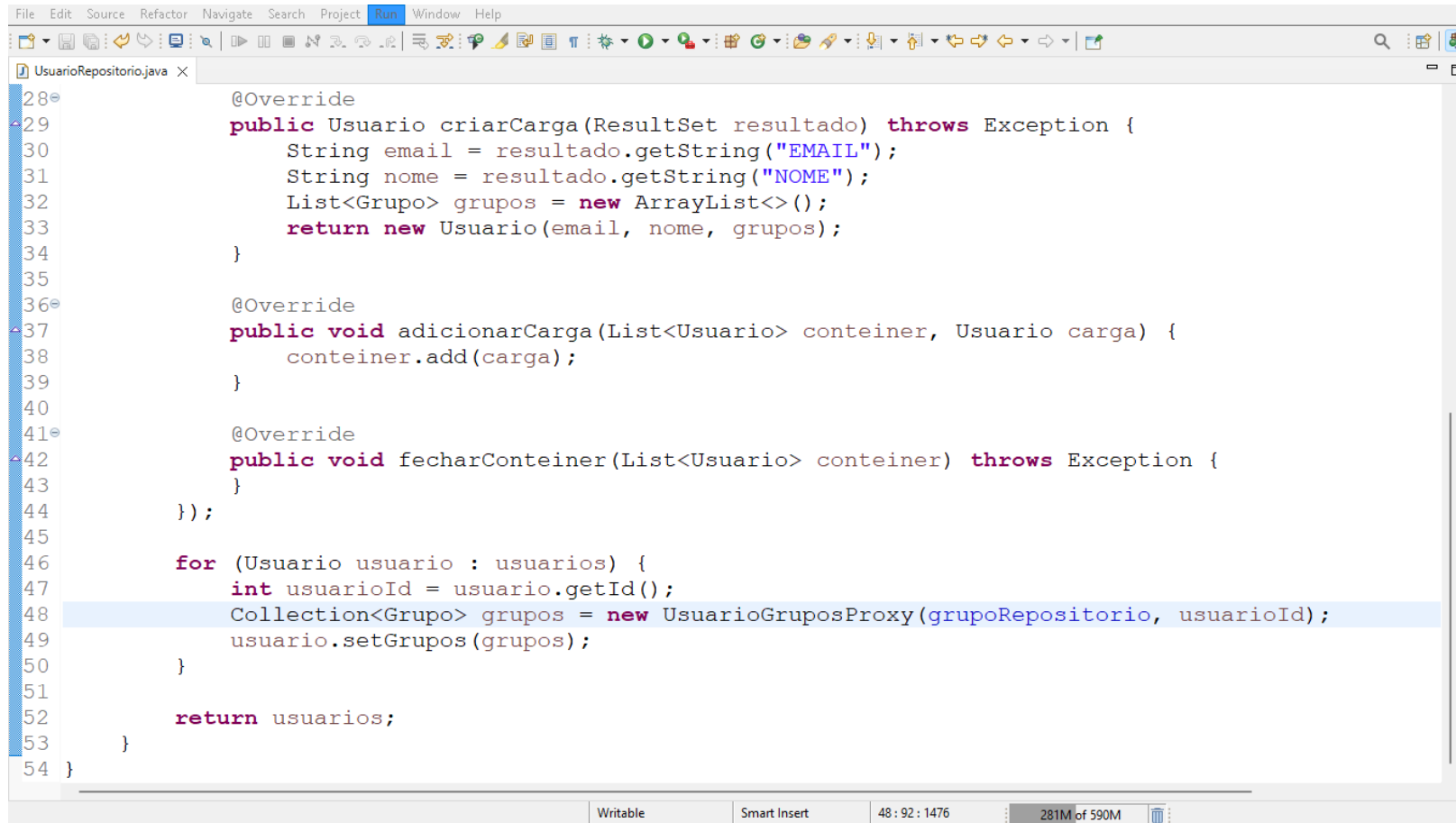
Writable Smart Insert 56 : 40 : 1229 307M of 590M

# Proxy

```
File Edit Source Refactor Navigate Search Project Run Window Help
UsuarioRepositorio.java X
1 package school.cesar.esa.padroes.estruturais.proxy2;
2
3 import java.sql.ResultSet;
4
5
6
7
8 public class UsuarioRepositorio {
9     private Repositorio repositorio;
10    private GrupoRepositorio grupoRepositorio;
11
12    public UsuarioRepositorio(Repositorio repositorio) {
13        if (repositorio == null) {
14            throw new IllegalArgumentException("O repositório não pode ser nulo");
15        }
16        this.repositorio = repositorio;
17    }
18
19    public List<Usuario> listar() throws Exception {
20        String consulta = "select ID, EMAIL, NOME from USUARIO";
21
22        List<Usuario> usuarios = repositorio.consultar(consulta, new ResultadoBuilder<List<Usuario>>
23            @Override
24            public List<Usuario> criarContainer() {
25                return new ArrayList<>();
26            }
27
28            @Override
29            public Usuario criarCarga(ResultSet resultado) throws Exception {
30                String email = resultado.getString("EMAIL");
```

Writable Smart Insert 8:34:202 280M of 590M

# Proxy



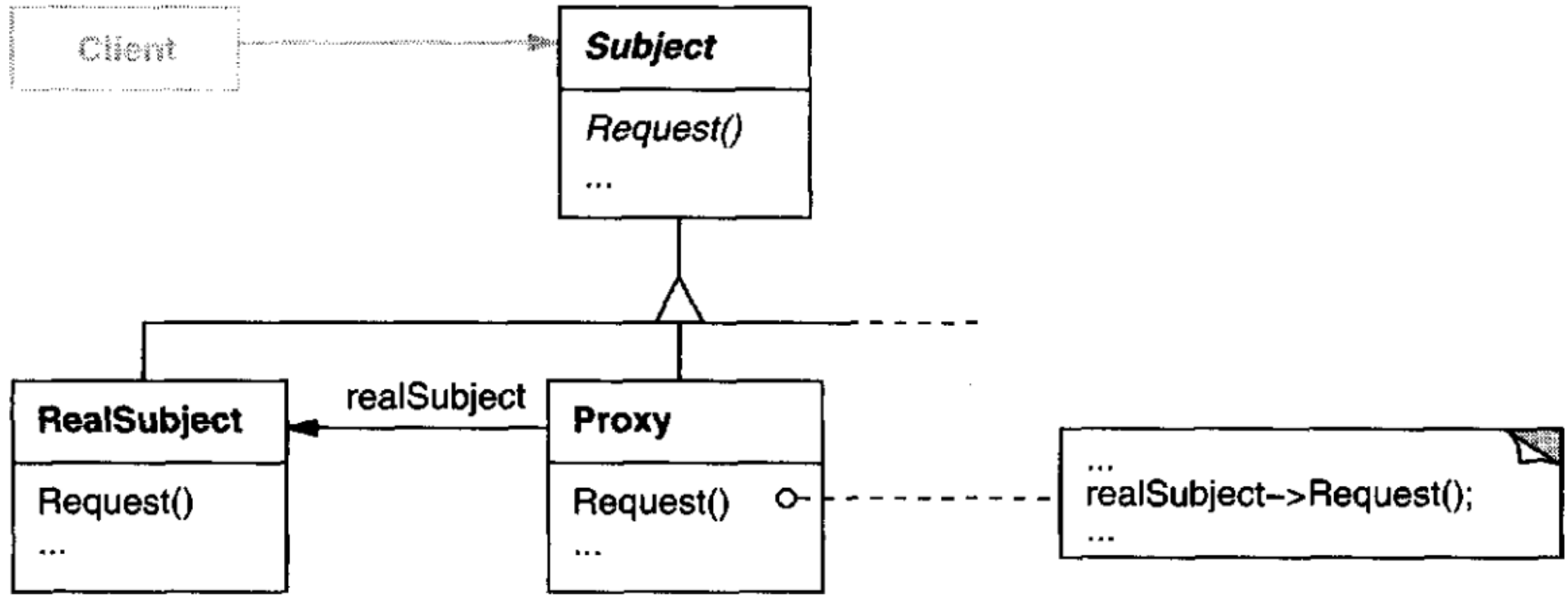
```
File Edit Source Refactor Navigate Search Project Run Window Help
[Icons]
UsuarioRepositorio.java X
28 @Override
29 public Usuario criarCarga(ResultSet resultado) throws Exception {
30     String email = resultado.getString("EMAIL");
31     String nome = resultado.getString("NOME");
32     List<Grupo> grupos = new ArrayList<>();
33     return new Usuario(email, nome, grupos);
34 }
35
36 @Override
37 public void adicionarCarga(List<Usuario> container, Usuario carga) {
38     container.add(carga);
39 }
40
41 @Override
42 public void fecharContainer(List<Usuario> container) throws Exception {
43 }
44 });
45
46 for (Usuario usuario : usuarios) {
47     int usuarioId = usuario.getId();
48     Collection<Grupo> grupos = new UsuarioGruposProxy(grupoRepositorio, usuarioId);
49     usuario.setGrupos(grupos);
50 }
51
52 return usuarios;
53 }
54 }
```

Writable Smart Insert 48 : 92 : 1476 281M of 590M

# Proxy

“Provide a surrogate or placeholder for another object to control access to it.”

# Proxy





# Proxy

- SOLID
  - Responsabilidade única (Single responsibility)
  - Aberto-fechado (Open-closed)
  - Substituição de Liskov (Liskob substitution)
  - Segregação de interfaces (Interface segregation)
  - Inversão de dependências (Dependency inversion)
- Prefira composição à herança
- Demeter

# Proxy

- Integridade conceitual
- (Alta) Coesão
- (Baixo) Acoplamento
- Ocultamento de informações