Padrões de projeto estruturais

Saulo Medeiros de Araujo

@sma

sma@cesar.school

Padrões estruturais

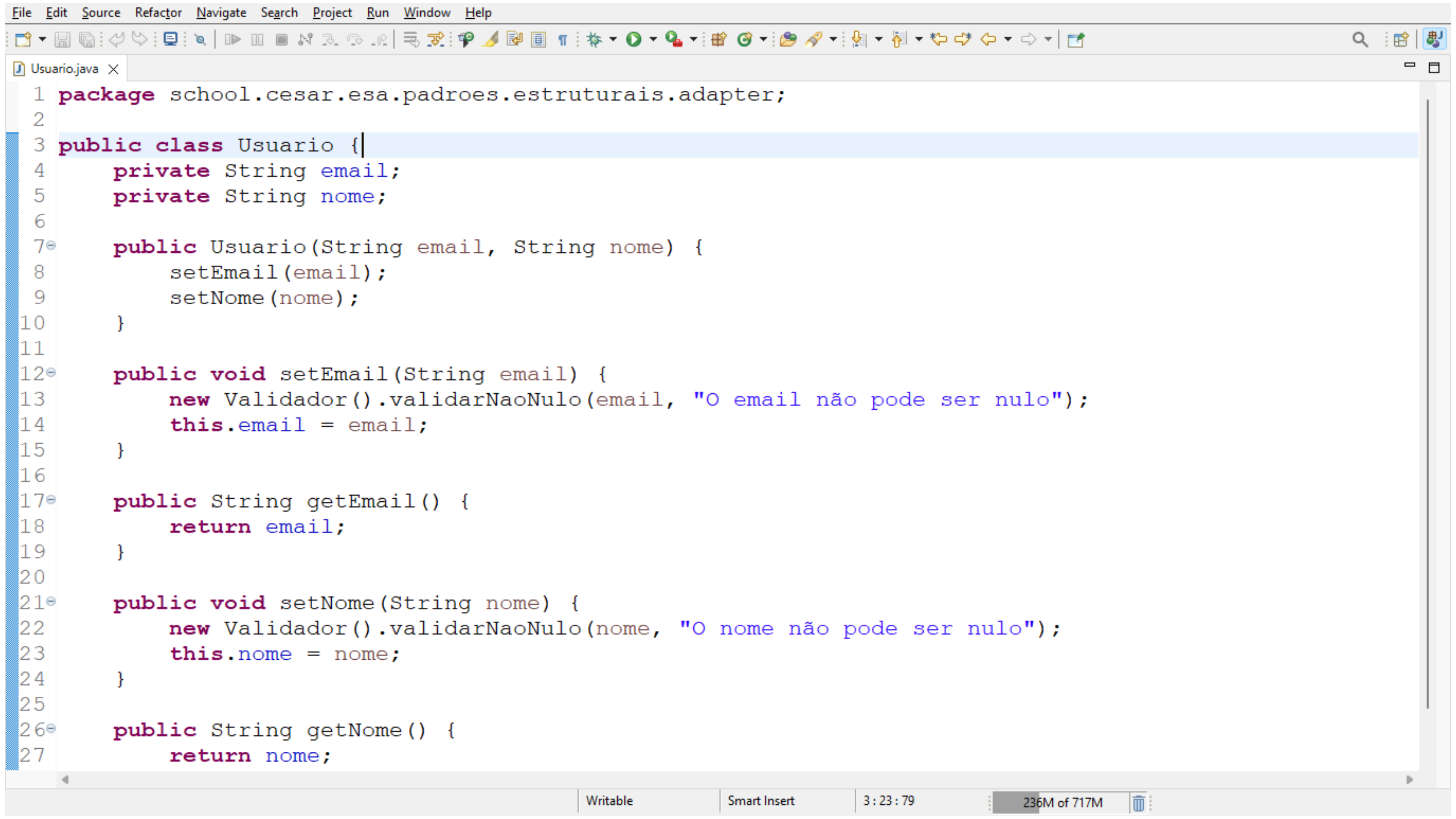
- Adapter
- Bridge
- Composite
- Decorator
- Facade
- Flyweight
- Proxy

Adapter

Adaptar-se é preciso

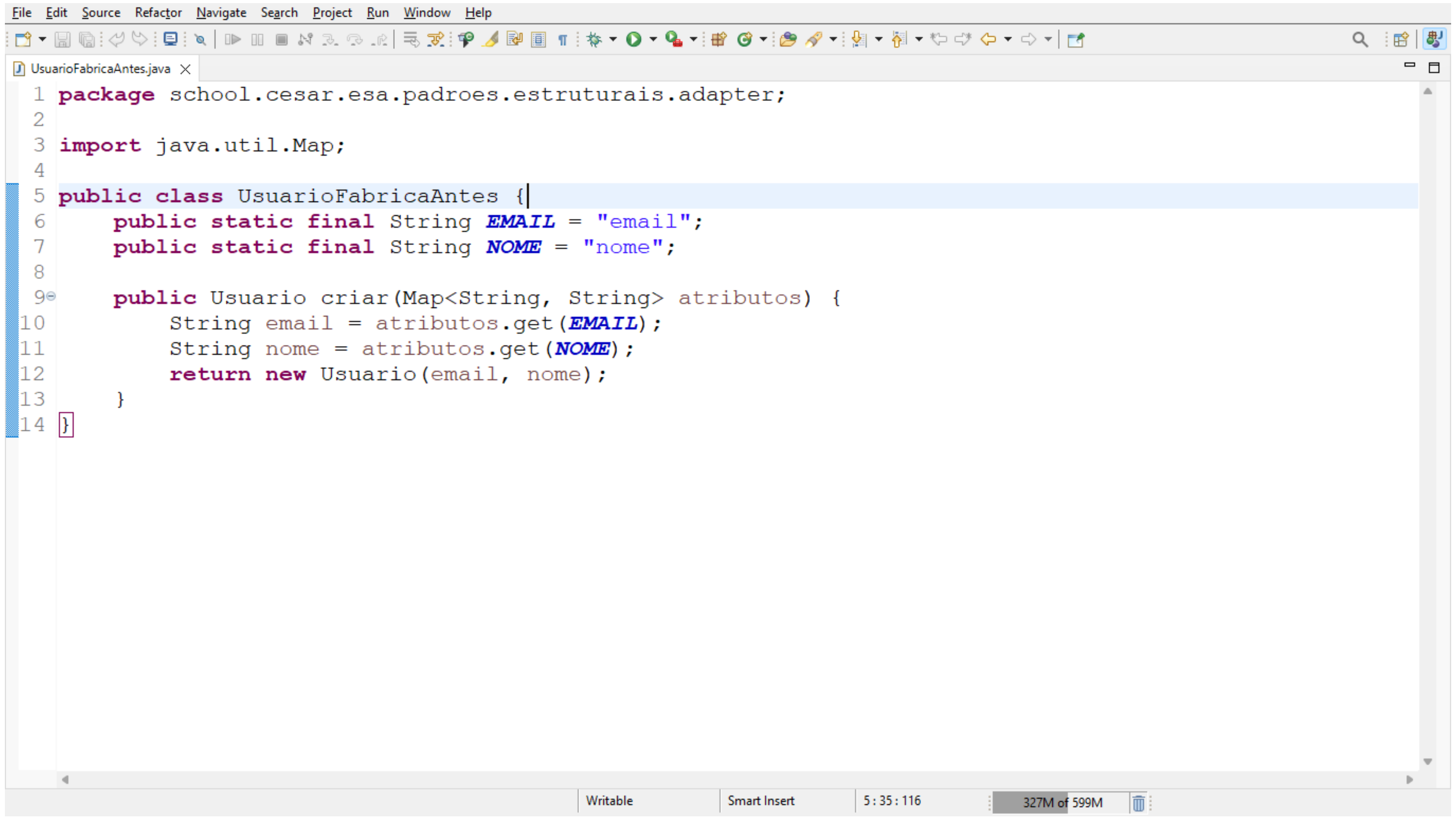
Aprenda a se Adaptar...
@wawawiwacomics
wawa wiwa

Adapter


File Edit Source Refactor Navigate Search Project Run Window Help
Usuario.java
Writable
Smart Insert
3 : 23 : 79
236M of 717M

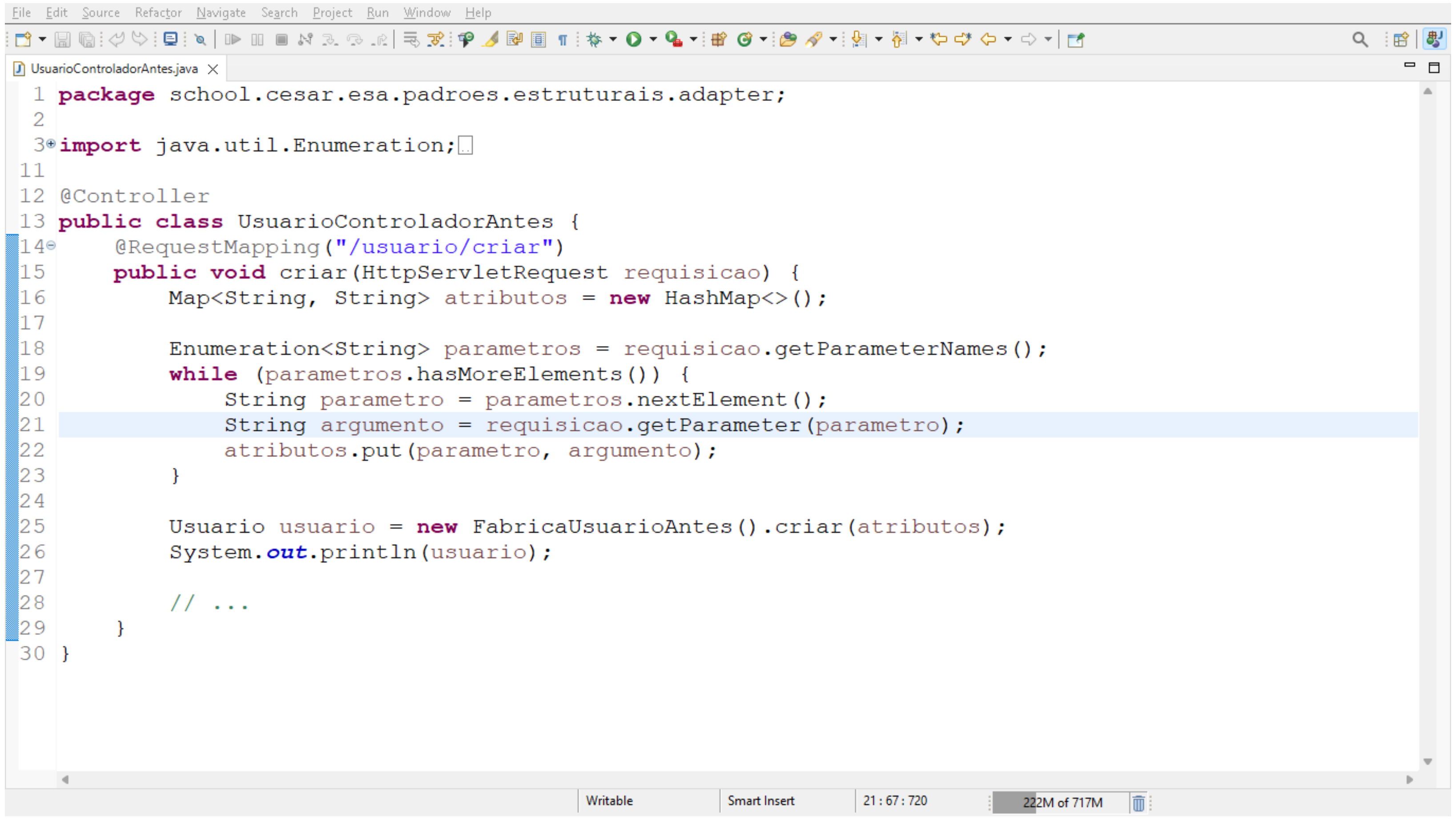
```java
package school.cesar.esa.padroes.estruturais.adapter;

public class Usuario {
    private String email;
    private String nome;

    public Usuario(String email, String nome) {
        setEmail(email);
        setNome(nome);
    }

    public void setEmail(String email) {
        new Validador().validarNaoNulo(email, "O email não pode ser nulo");
        this.email = email;
    }

    public String getEmail() {
        return email;
    }

    public void setNome(String nome) {
        new Validador().validarNaoNulo(nome, "O nome não pode ser nulo");
        this.nome = nome;
    }

    public String getNome() {
        return nome;
```

Adapter



```java
package school.cesar.esa.padroes.estruturais.adapter;

import java.util.Map;

public class UsuarioFabricaAntes {
    public static final String EMAIL = "email";
    public static final String NOME = "nome";

    public Usuario criar(Map<String, String> atributos) {
        String email = atributos.get(EMAIL);
        String nome = atributos.get(NOME);
        return new Usuario(email, nome);
    }
}
```

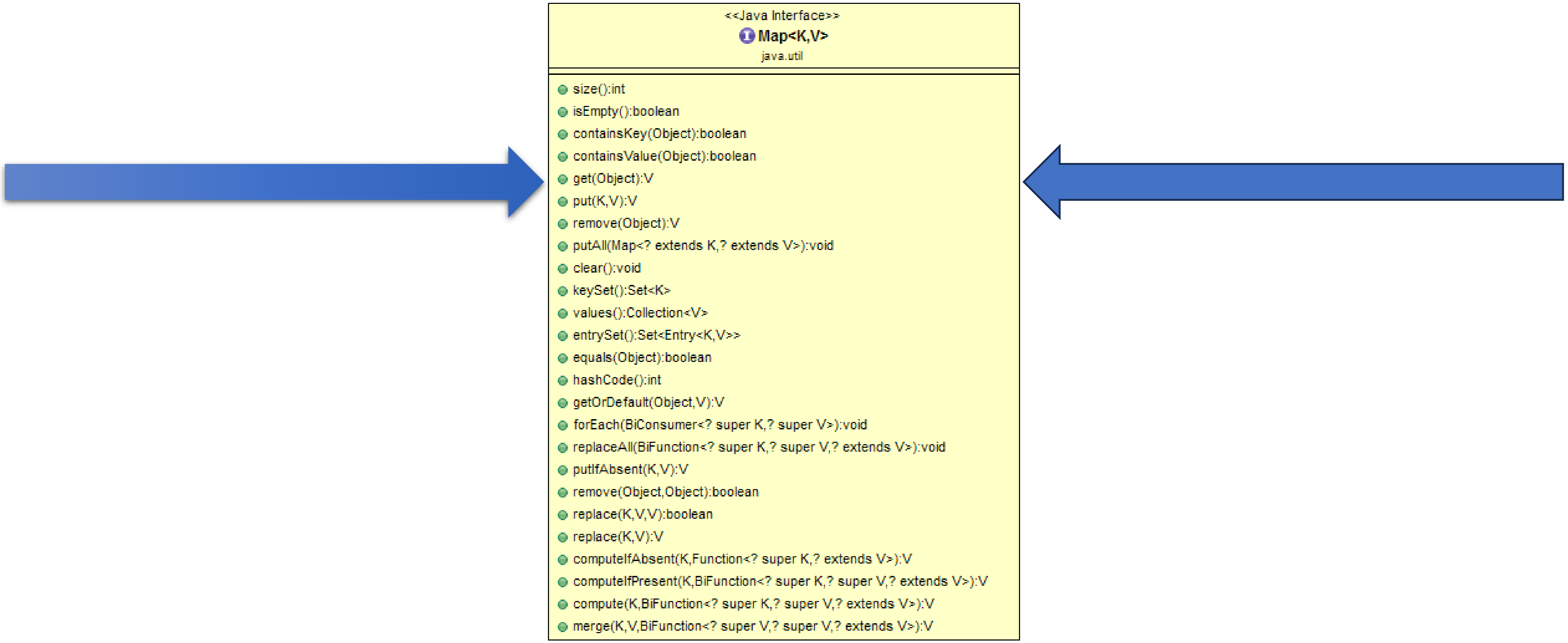
Adapter



File Edit Source Refactor Navigate Search Project Run Window Help
UsuarioControladorAntes.java

```java
package school.cesar.esa.padroes.estruturais.adapter;

import java.util.Enumeration;

@Controller
public class UsuarioControladorAntes {
    @RequestMapping("/usuario/criar")
    public void criar(HttpServletRequest requisicao) {
        Map<String, String> atributos = new HashMap<>();

        Enumeration<String> parametros = requisicao.getParameterNames();
        while (parametros.hasMoreElements()) {
            String parametro = parametros.nextElement();
            String argumento = requisicao.getParameter(parametro);
            atributos.put(parametro, argumento);
        }

        Usuario usuario = new FabricaUsuarioAntes().criar(atributos);
        System.out.println(usuario);

        // ...
    }
}
```

Adapter

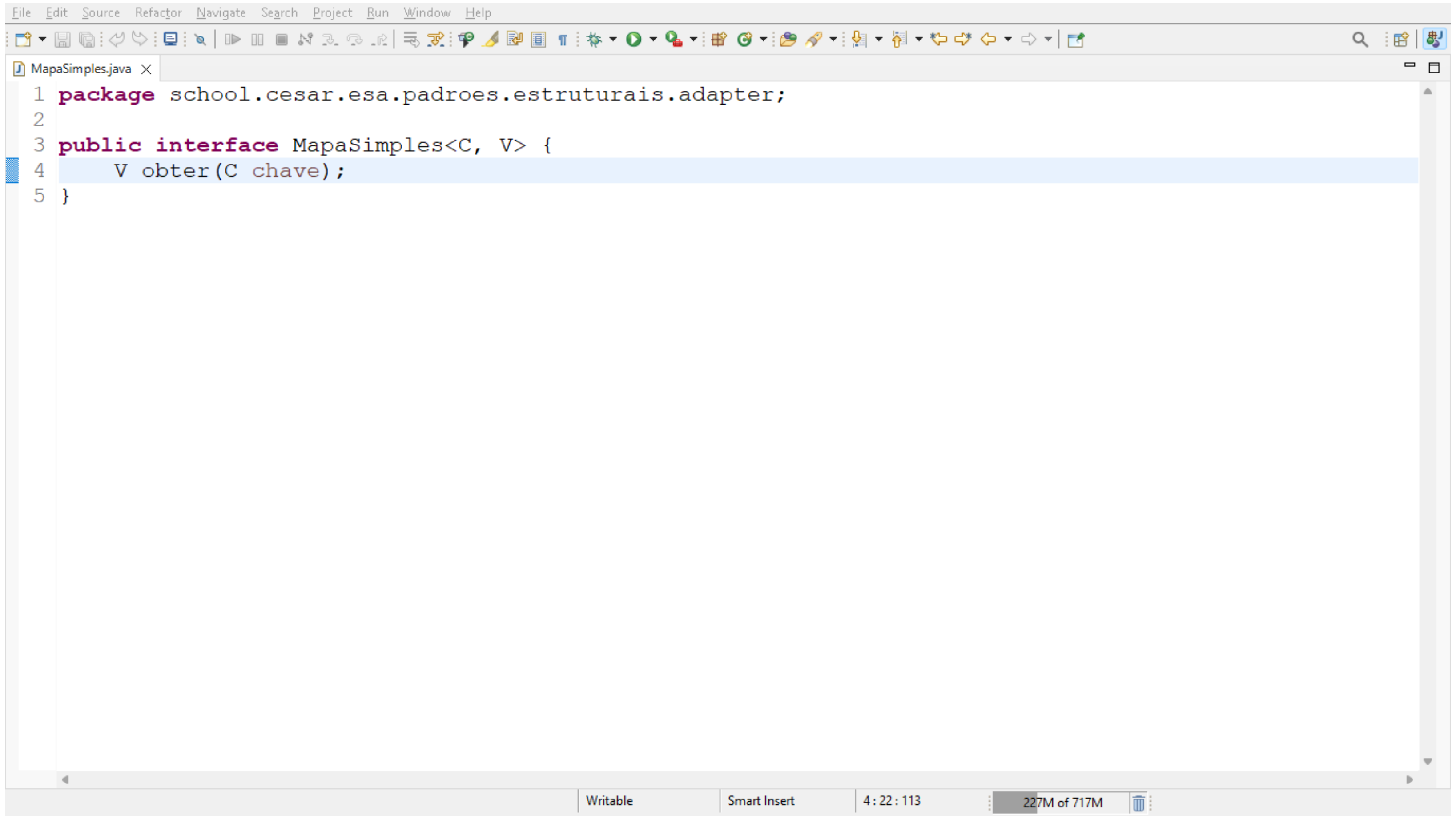


<<Java Interface>>
Map<K,V>
java.util

- size():int
- isEmpty():boolean
- containsKey(Object):boolean
- containsValue(Object):boolean
- get(Object):V
- put(K,V):V
- remove(Object):V
- putAll(Map<? extends K,? extends V>):void
- clear():void
- keySet():Set<K>
- values():Collection<V>
- entrySet():Set<Entry<K,V>>
- equals(Object):boolean
- hashCode():int
- getOrDefault(Object,V):V
- forEach(BiConsumer<? super K,? super V>):void
- replaceAll(BiFunction<? super K,? super V,? extends V>):void
- putIfAbsent(K,V):V
- remove(Object,Object):boolean
- replace(K,V,V):boolean
- replace(K,V):V
- computeIfAbsent(K,Function<? super K,? extends V>):V
- computeIfPresent(K,BiFunction<? super K,? super V,? extends V>):V
- compute(K,BiFunction<? super K,? super V,? extends V>):V
- merge(K,V,BiFunction<? super V,? super V,? extends V>):V

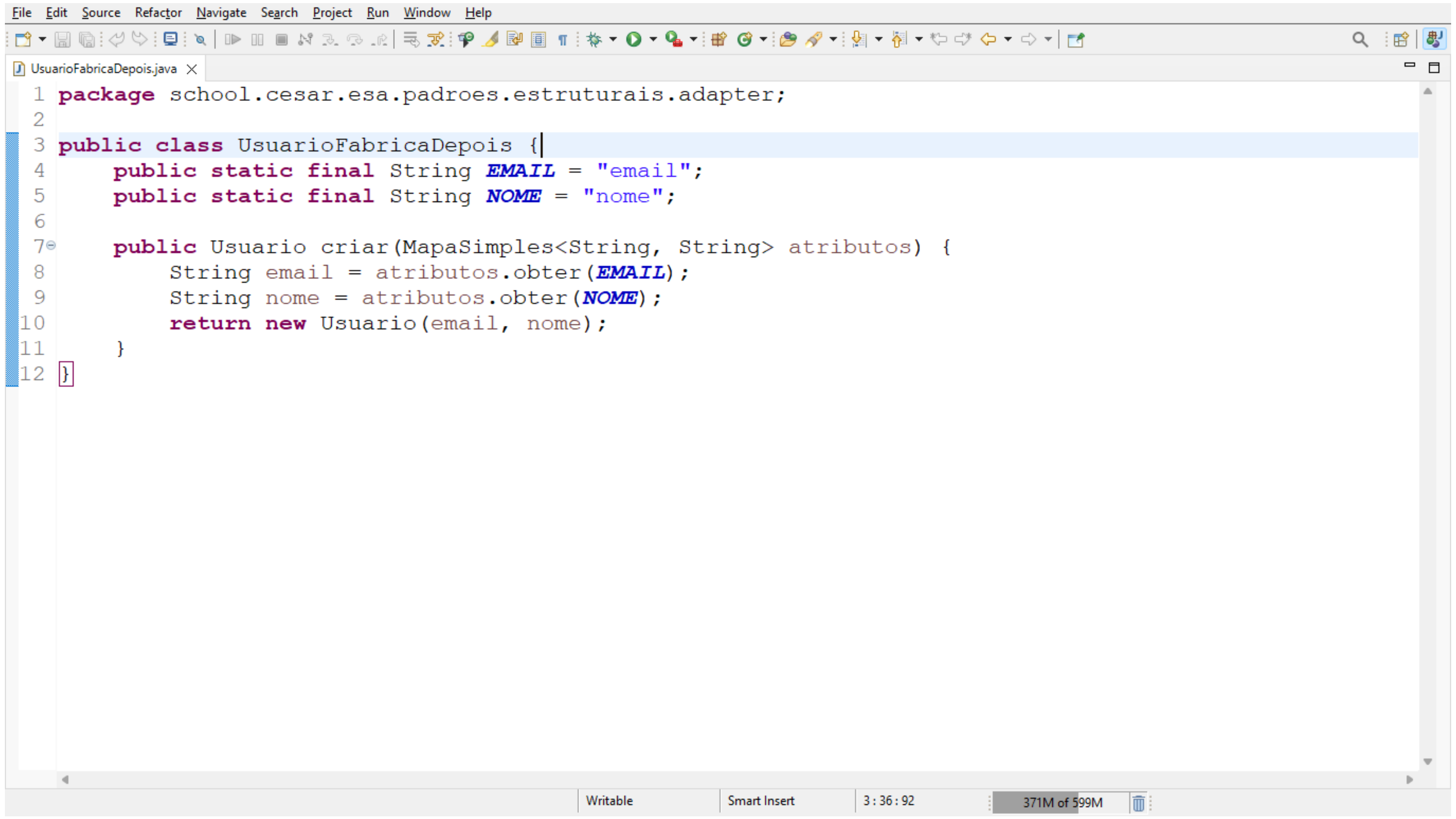
Adaptadores

Adapter


File Edit Source Refactor Navigate Search Project Run Window Help
MapaSimples.java
Writable
Smart Insert
4 : 22 : 113
227M of 717M

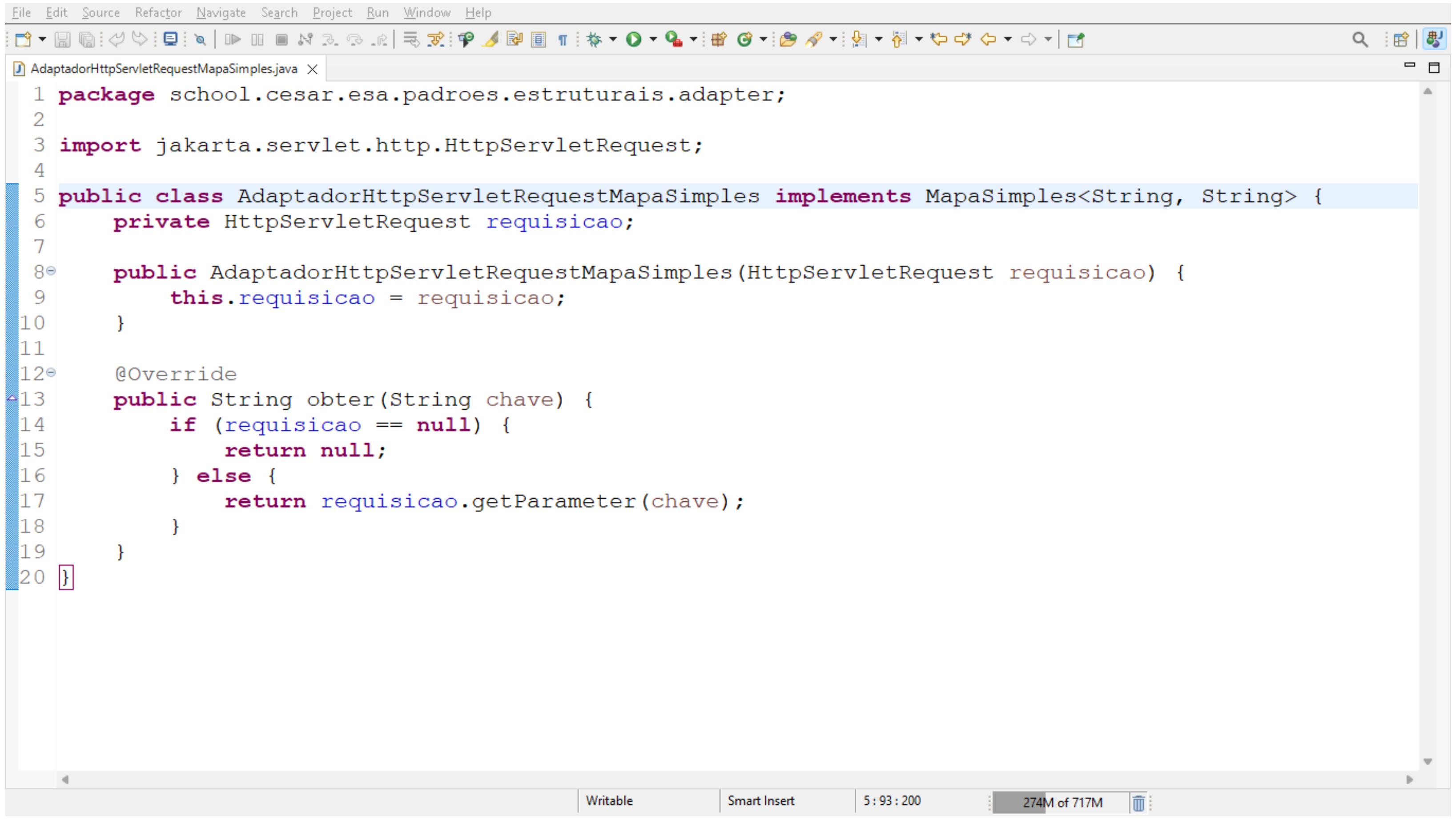
```java
package school.cesar.esa.padroes.estruturais.adapter;

public interface MapaSimples<C, V> {
    V obter(C chave);
}
```

Adapter


File Edit Source Refactor Navigate Search Project Run Window Help
UsuarioFabricaDepois.java
Writable
Smart Insert
3 : 36 : 92
371M of 599M

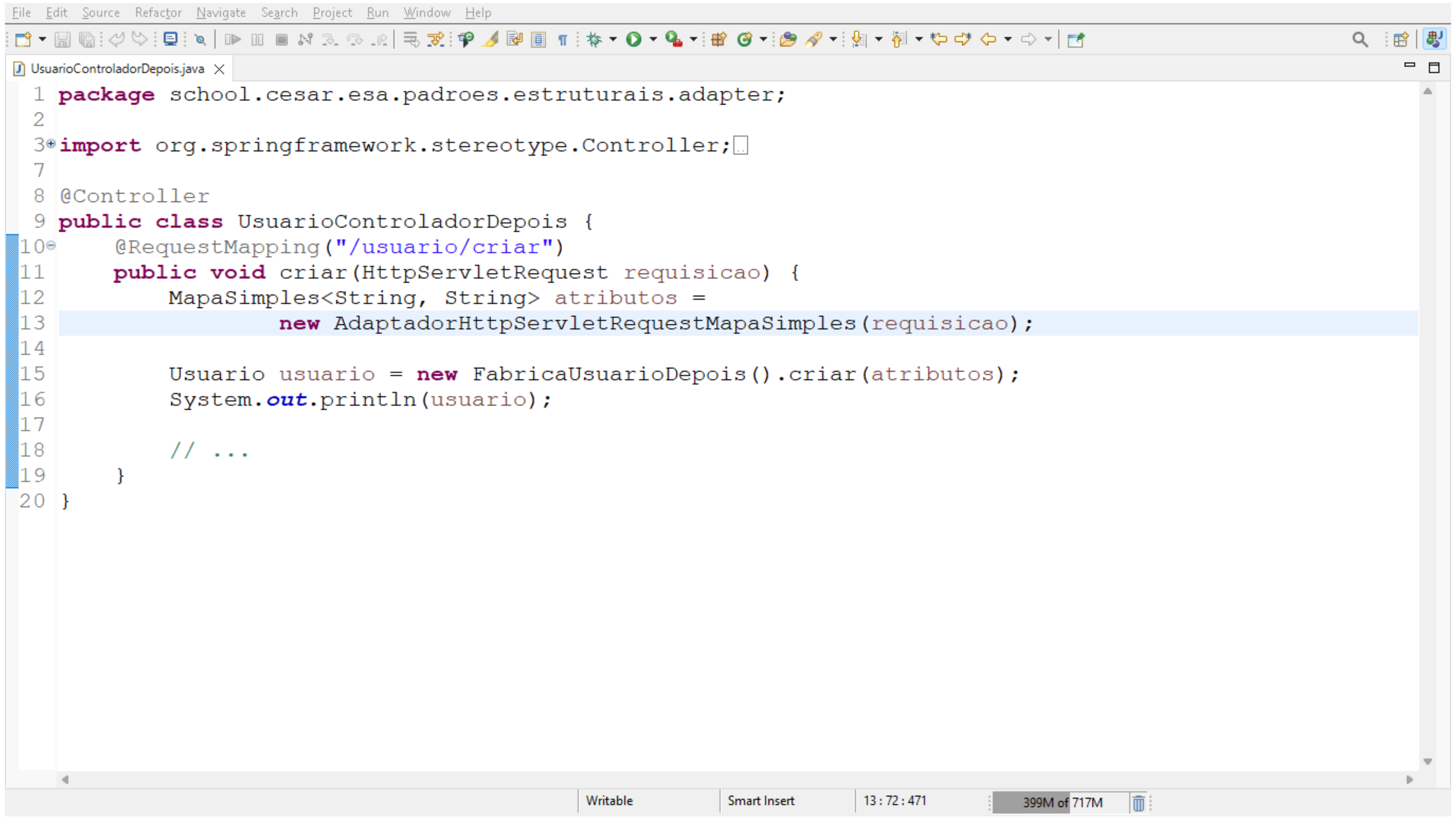
```java
package school.cesar.esa.padroes.estruturais.adapter;

public class UsuarioFabricaDepois {
    public static final String EMAIL = "email";
    public static final String NOME = "nome";

    public Usuario criar(MapaSimples<String, String> atributos) {
        String email = atributos.obter(EMAIL);
        String nome = atributos.obter(NOME);
        return new Usuario(email, nome);
    }
}
```

Adapter


File Edit Source Refactor Navigate Search Project Run Window Help
AdaptadorHttpServletRequestMapaSimples.java
Writable
Smart Insert
5 : 93 : 200
274M of 717M

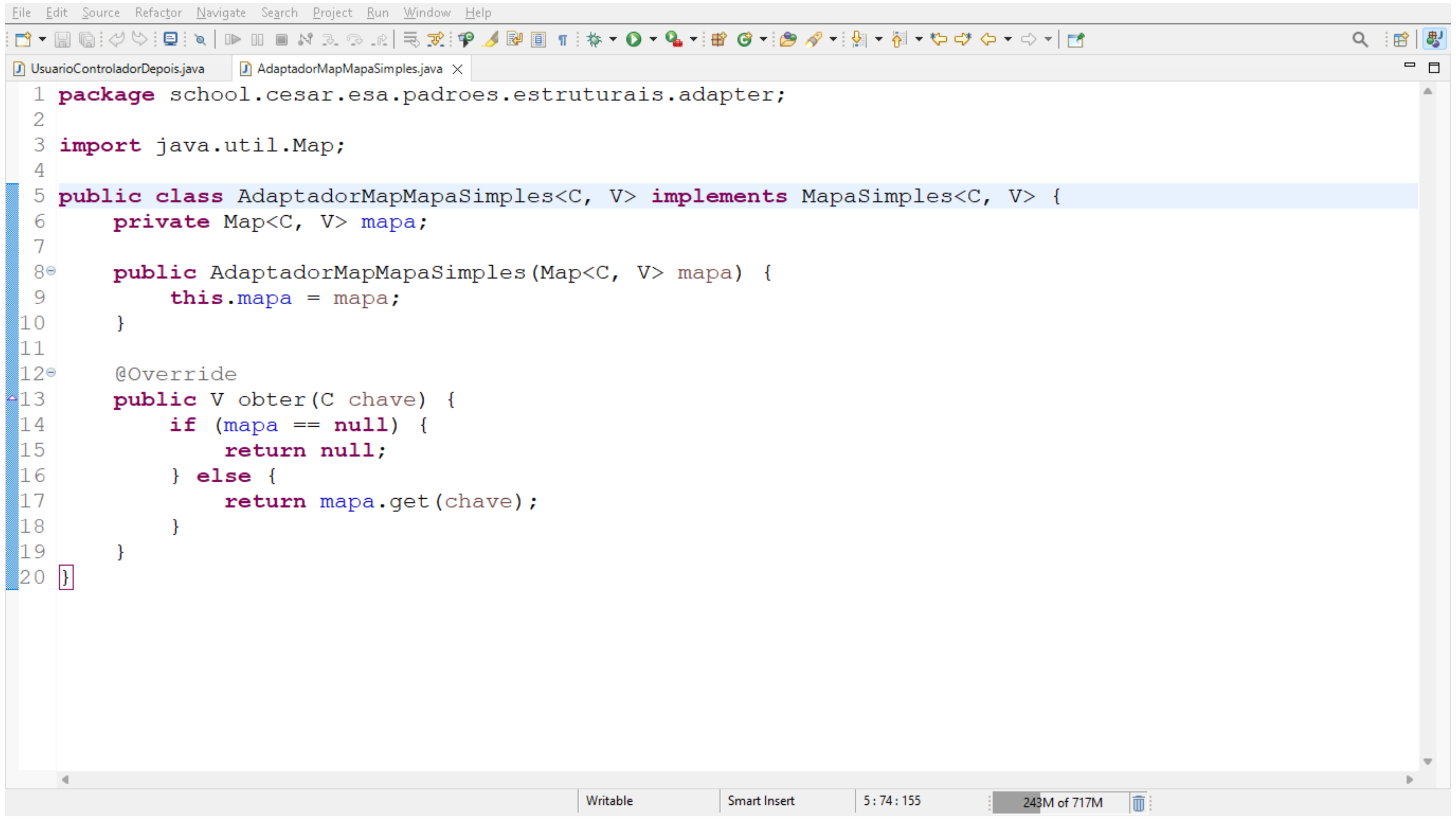
```java
package school.cesar.esa.padroes.estruturais.adapter;

import jakarta.servlet.http.HttpServletRequest;

public class AdaptadorHttpServletRequestMapaSimples implements MapaSimples<String, String> {
    private HttpServletRequest requisicao;

    public AdaptadorHttpServletRequestMapaSimples(HttpServletRequest requisicao) {
        this.requisicao = requisicao;
    }

    @Override
    public String obter(String chave) {
        if (requisicao == null) {
            return null;
        } else {
            return requisicao.getParameter(chave);
        }
    }
}
```

Adapter


File Edit Source Refactor Navigate Search Project Run Window Help
UsuarioControladorDepois.java
Writable
Smart Insert
13 : 72 : 471
399M of 717M

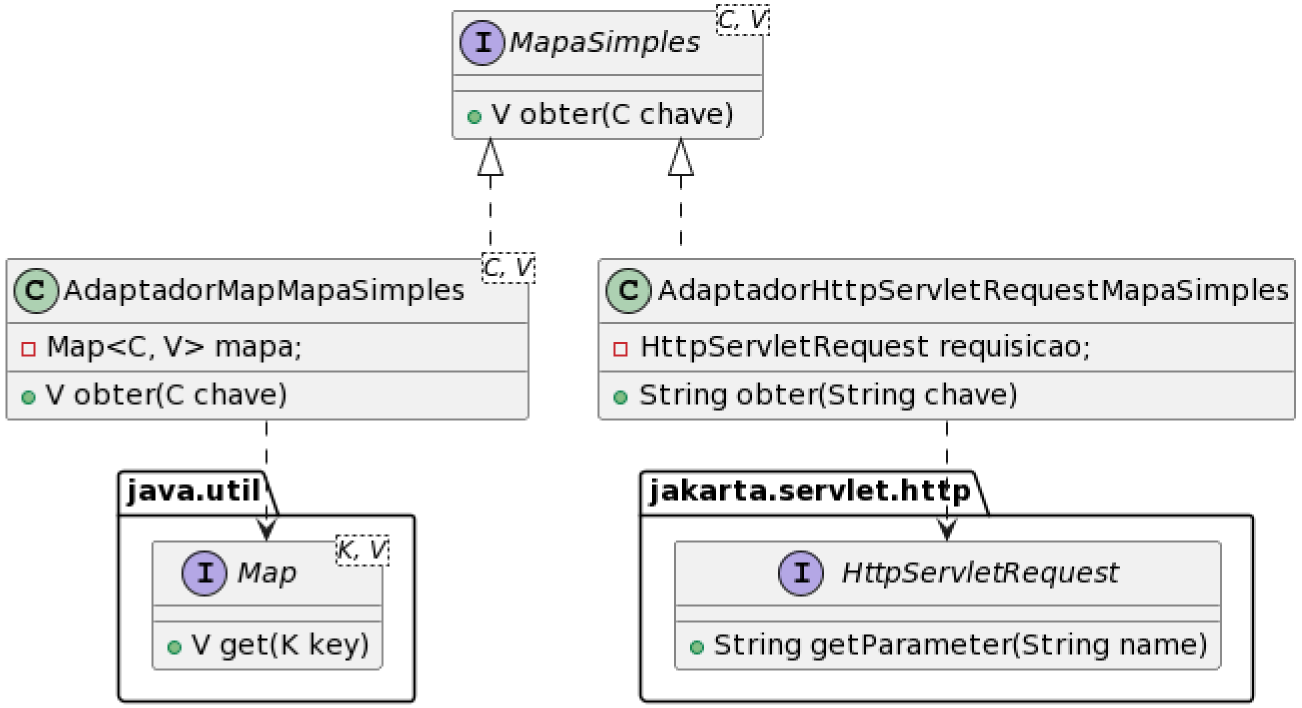
```java
package school.cesar.esa.padroes.estruturais.adapter;

import org.springframework.stereotype.Controller;

@Controller
public class UsuarioControladorDepois {
    @RequestMapping("/usuario/criar")
    public void criar(HttpServletRequest requisicao) {
        MapaSimples<String, String> atributos =
                new AdaptadorHttpServletRequestMapaSimples(requisicao);

        Usuario usuario = new FabricaUsuarioDepois().criar(atributos);
        System.out.println(usuario);

        // ...
    }
}
```

Adapter


File Edit Source Refactor Navigate Search Project Run Window Help
UsuarioControladorDepois.java
AdaptadorMapMapaSimples.java
Writable
Smart Insert
5 : 74 : 155
243M of 717M

```java
package school.cesar.esa.padroes.estruturais.adapter;

import java.util.Map;

public class AdaptadorMapMapaSimples<C, V> implements MapaSimples<C, V> {
    private Map<C, V> mapa;

    public AdaptadorMapMapaSimples(Map<C, V> mapa) {
        this.mapa = mapa;
    }

    @Override
    public V obter(C chave) {
        if (mapa == null) {
            return null;
        } else {
            return mapa.get(chave);
        }
    }
}
```

Adapter

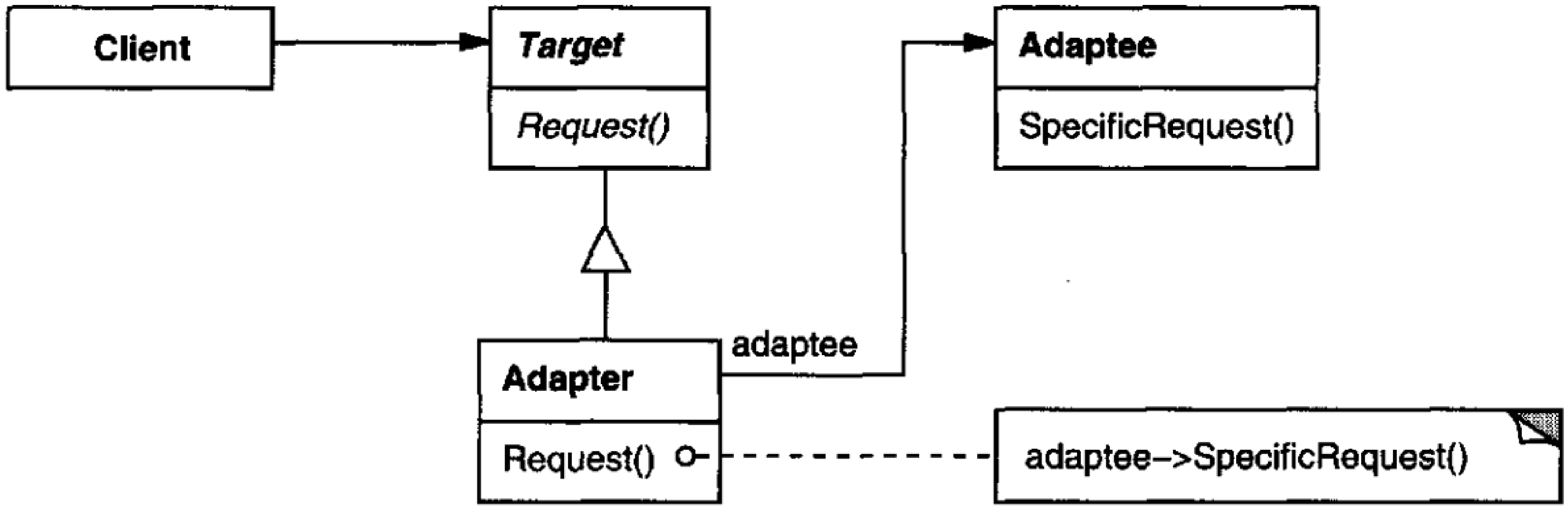


MapaSimples C, V
V obter(C chave)
AdaptadorMapMapaSimples C, V
Map<C, V> mapa;
V obter(C chave)
AdaptadorHttpServletRequestMapaSimples
HttpServletRequest requisicao;
String obter(String chave)
java.util
Map K, V
V get(K key)
jakarta.servlet.http
HttpServletRequest
String getParameter(String name)

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


Client
Target
Request()
Adaptee
SpecificRequest()
adaptee
Adapter
Request()
adaptee->SpecificRequest()

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)

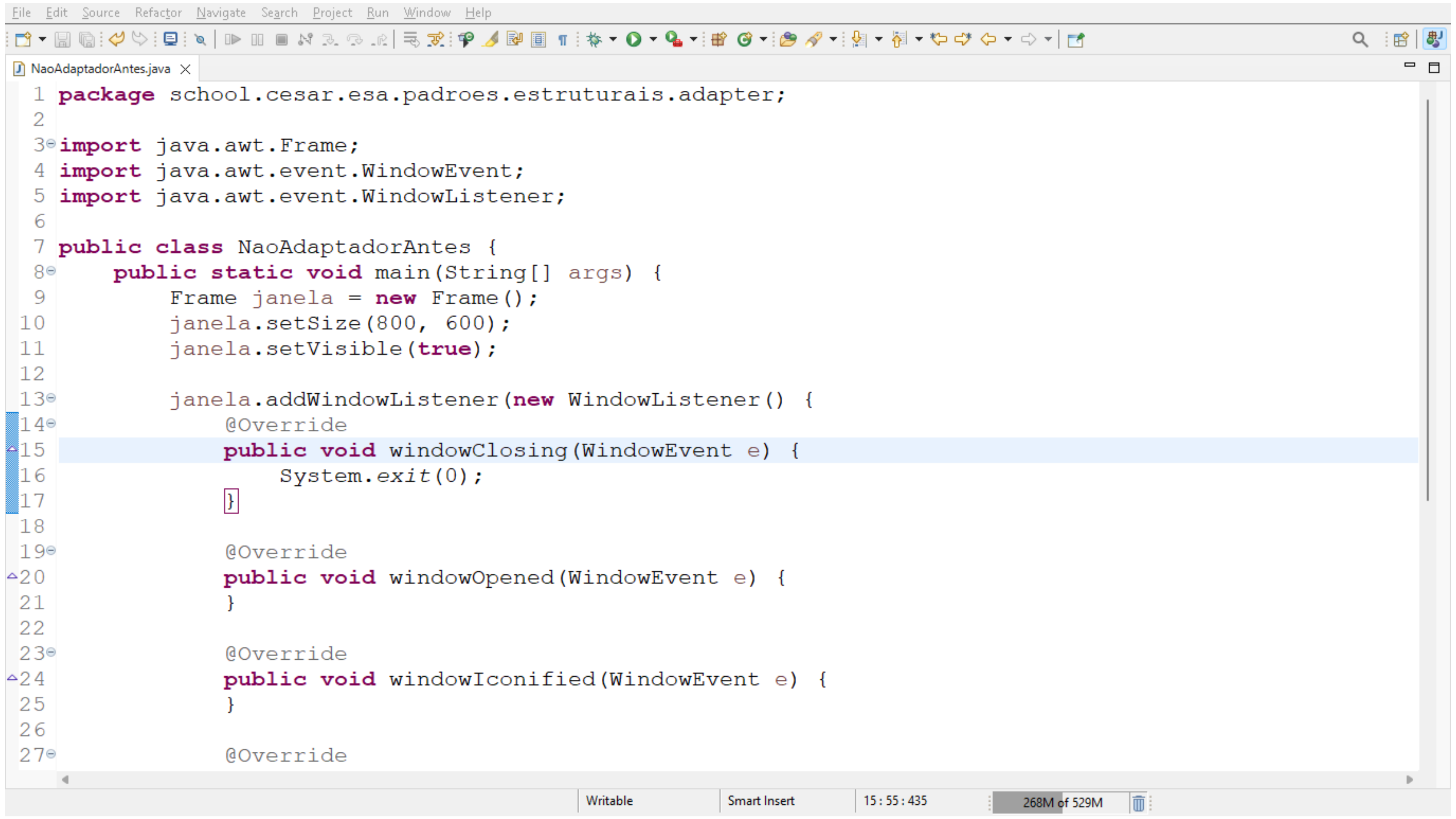
- Prefira composição à herança
- Demeter

Adapter

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

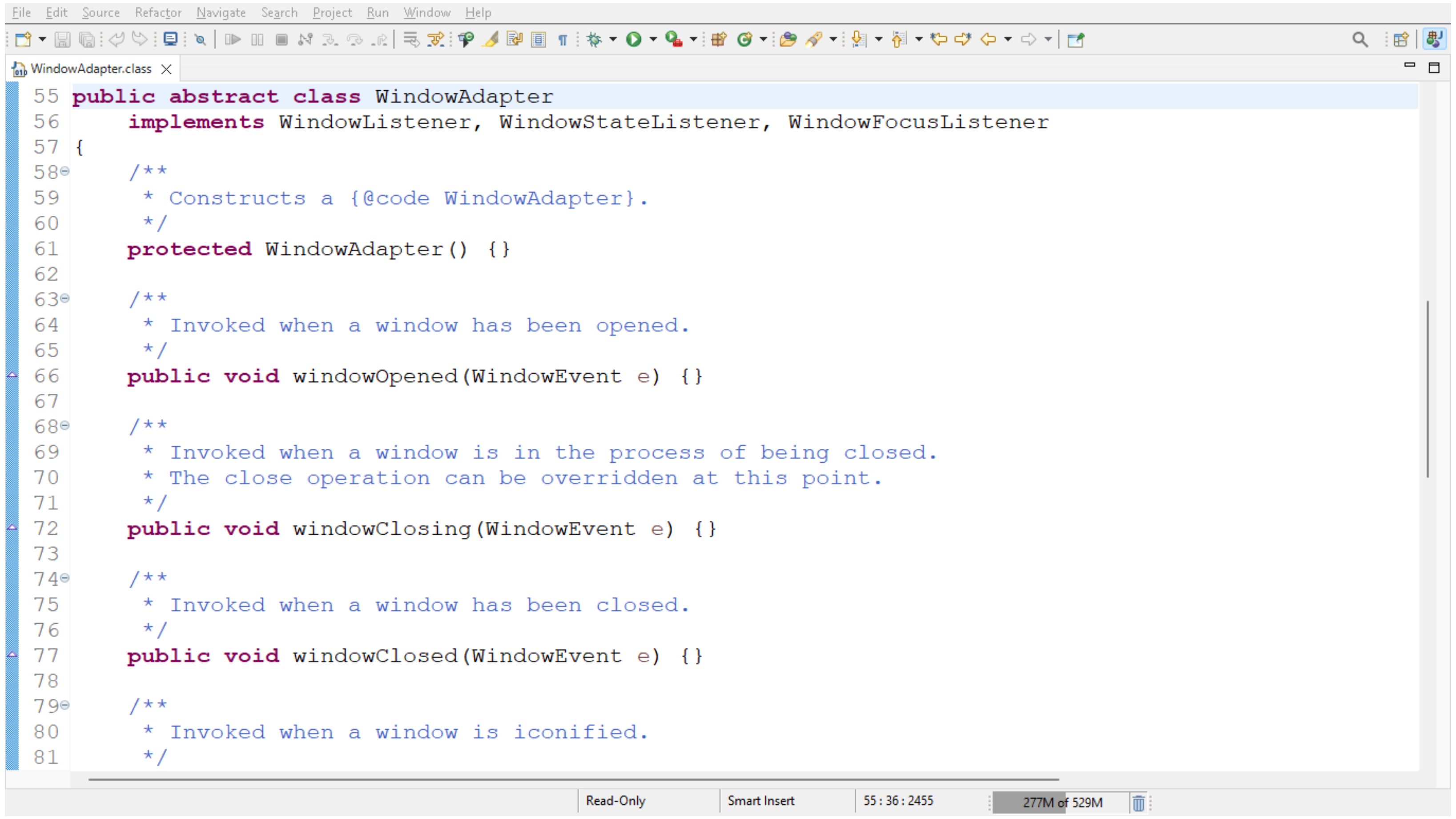
Adapter

Adapter



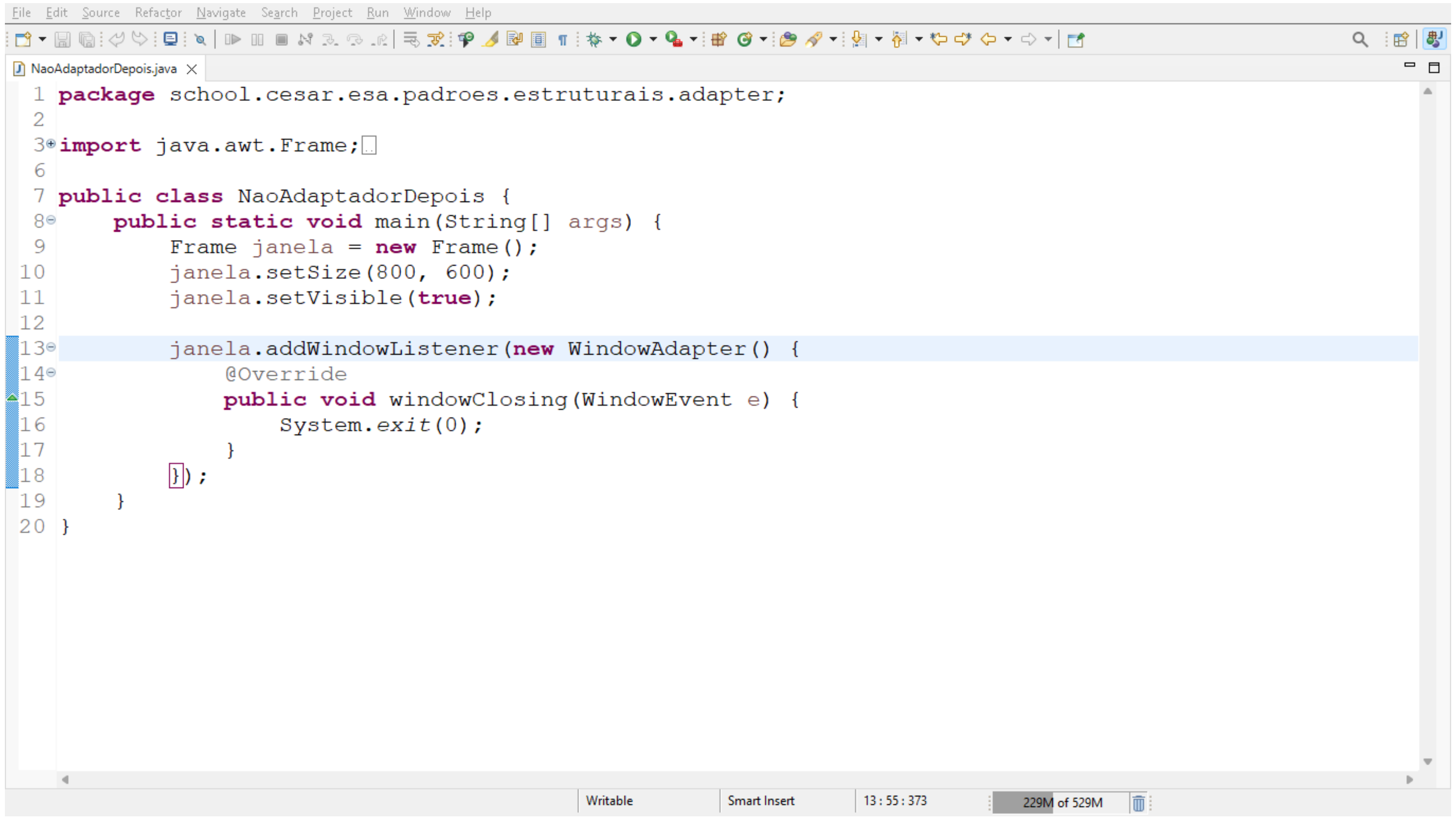
```java
package school.cesar.esa.padroes.estruturais.adapter;

import java.awt.Frame;
import java.awt.event.WindowEvent;
import java.awt.event.WindowListener;

public class NaoAdaptadorAntes {
    public static void main(String[] args) {
        Frame janela = new Frame();
        janela.setSize(800, 600);
        janela.setVisible(true);

        janela.addWindowListener(new WindowListener() {
            @Override
            public void windowClosing(WindowEvent e) {
                System.exit(0);
            }

            @Override
            public void windowOpened(WindowEvent e) {
            }

            @Override
            public void windowIconified(WindowEvent e) {
            }

            @Override
```

Adapter


File Edit Source Refactor Navigate Search Project Run Window Help
WindowAdapter.class

```
public abstract class WindowAdapter
    implements WindowListener, WindowStateListener, WindowFocusListener
{
    /**
     * Constructs a {@code WindowAdapter}.
     */
    protected WindowAdapter() {}

    /**
     * Invoked when a window has been opened.
     */
    public void windowOpened(WindowEvent e) {}

    /**
     * Invoked when a window is in the process of being closed.
     * The close operation can be overridden at this point.
     */
    public void windowClosing(WindowEvent e) {}

    /**
     * Invoked when a window has been closed.
     */
    public void windowClosed(WindowEvent e) {}

    /**
     * Invoked when a window is iconified.
     */
```

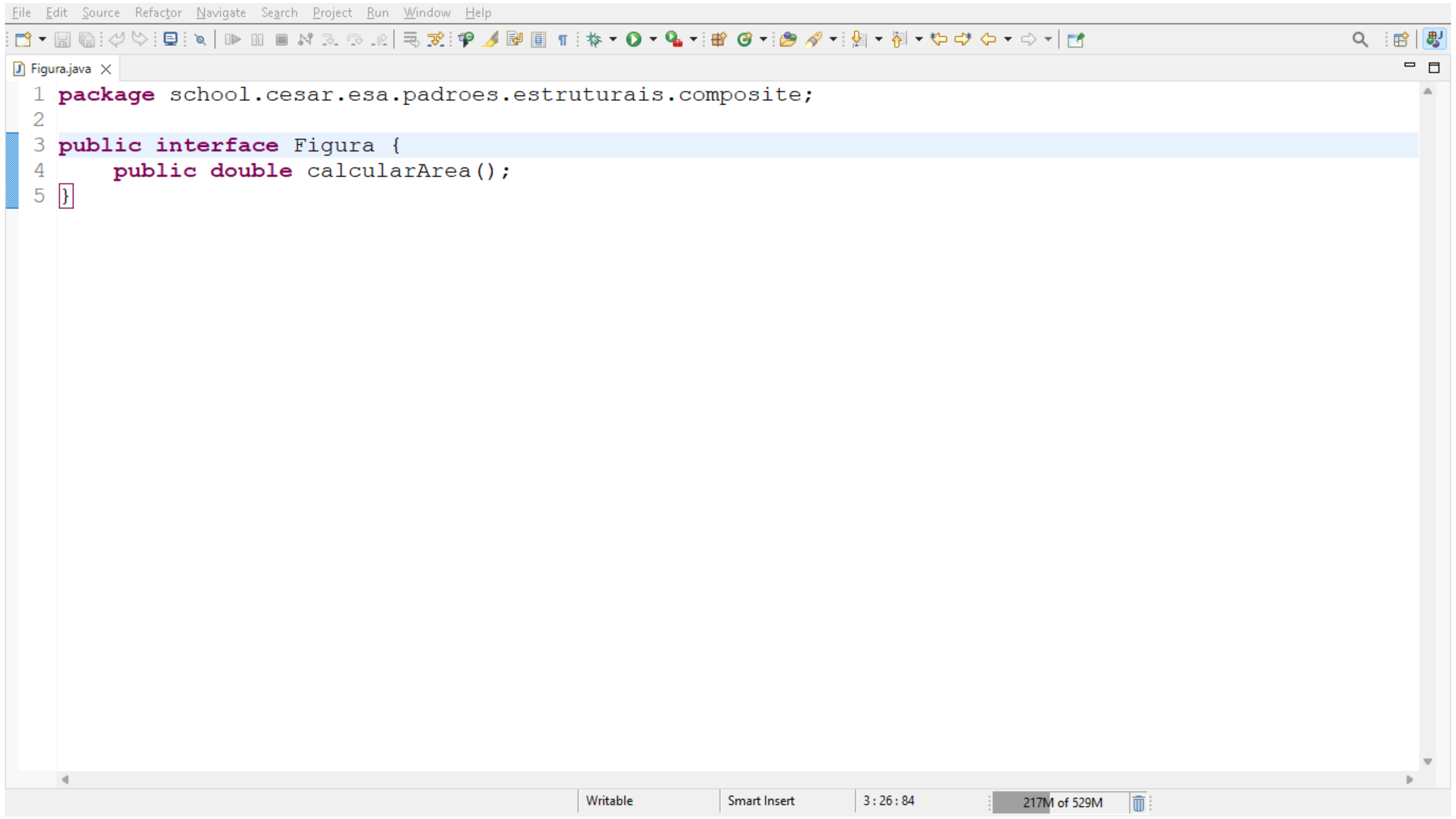
Adapter



```
package school.cesar.esa.padroes.estruturais.adapter;

import java.awt.Frame;

public class NaoAdaptadorDepois {
    public static void main(String[] args) {
        Frame janela = new Frame();
        janela.setSize(800, 600);
        janela.setVisible(true);

        janela.addWindowListener(new WindowAdapter() {
            @Override
            public void windowClosing(WindowEvent e) {
                System.exit(0);
            }
        });
    }
}
```

Composite

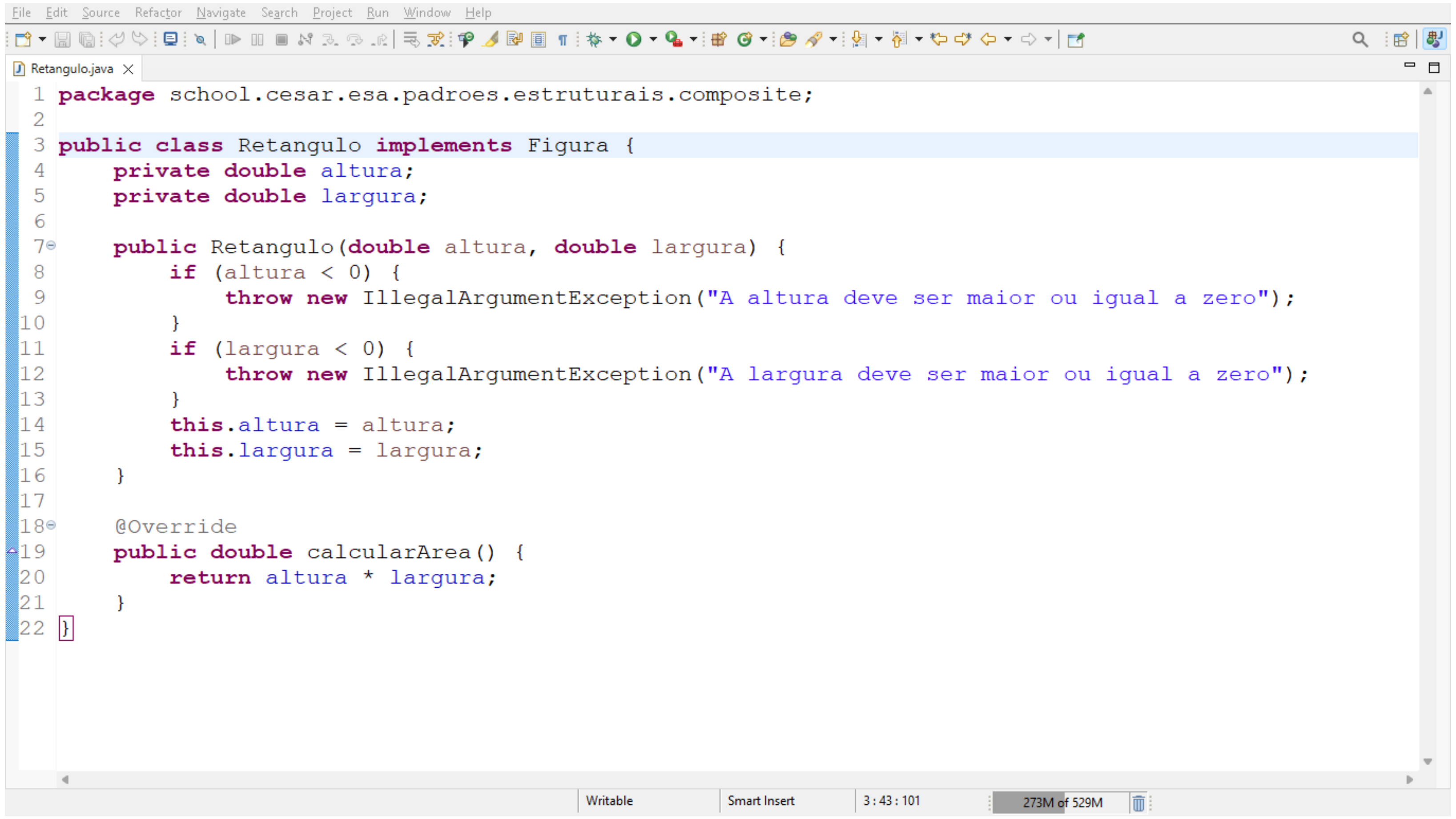
Composite


File Edit Source Refactor Navigate Search Project Run Window Help
Figura.java

```java
package school.cesar.esa.padroes.estruturais.composite;

public interface Figura {
    public double calcularArea();
}
```

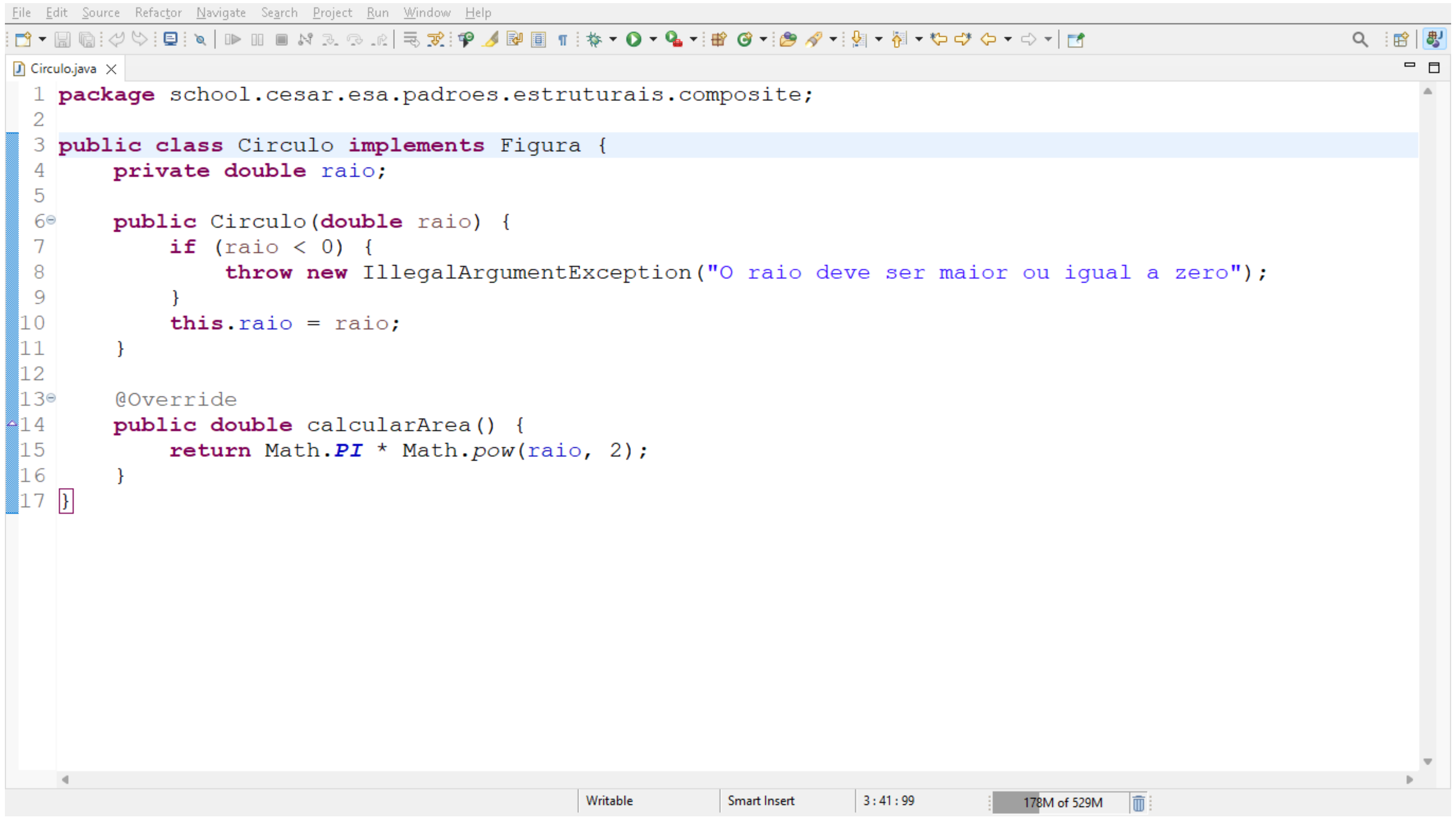
Writable
Smart Insert
3 : 26 : 84
217M of 529M

Composite


File Edit Source Refactor Navigate Search Project Run Window Help
Retangulo.java
Writable
Smart Insert
3 : 43 : 101
273M of 529M

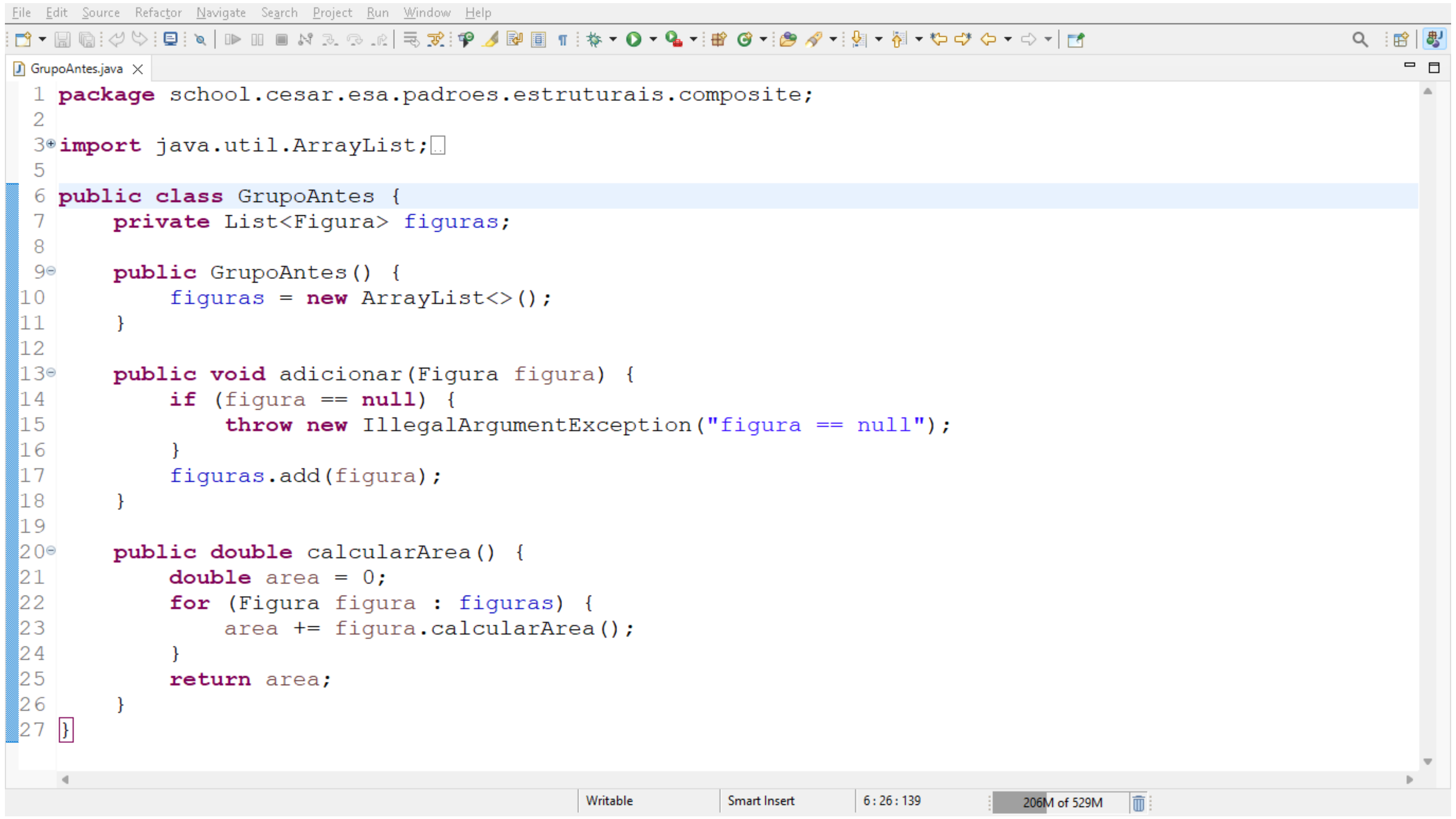
```java
package school.cesar.esa.padroes.estruturais.composite;

public class Retangulo implements Figura {
    private double altura;
    private double largura;

    public Retangulo(double altura, double largura) {
        if (altura < 0) {
            throw new IllegalArgumentException("A altura deve ser maior ou igual a zero");
        }
        if (largura < 0) {
            throw new IllegalArgumentException("A largura deve ser maior ou igual a zero");
        }
        this.altura = altura;
        this.largura = largura;
    }

    @Override
    public double calcularArea() {
        return altura * largura;
    }
}
```

Composite


File Edit Source Refactor Navigate Search Project Run Window Help
Circulo.java
Writable
Smart Insert
3 : 41 : 99
178M of 529M

```java
package school.cesar.esa.padroes.estruturais.composite;

public class Circulo implements Figura {
    private double raio;

    public Circulo(double raio) {
        if (raio < 0) {
            throw new IllegalArgumentException("O raio deve ser maior ou igual a zero");
        }
        this.raio = raio;
    }

    @Override
    public double calcularArea() {
        return Math.PI * Math.pow(raio, 2);
    }
}
```

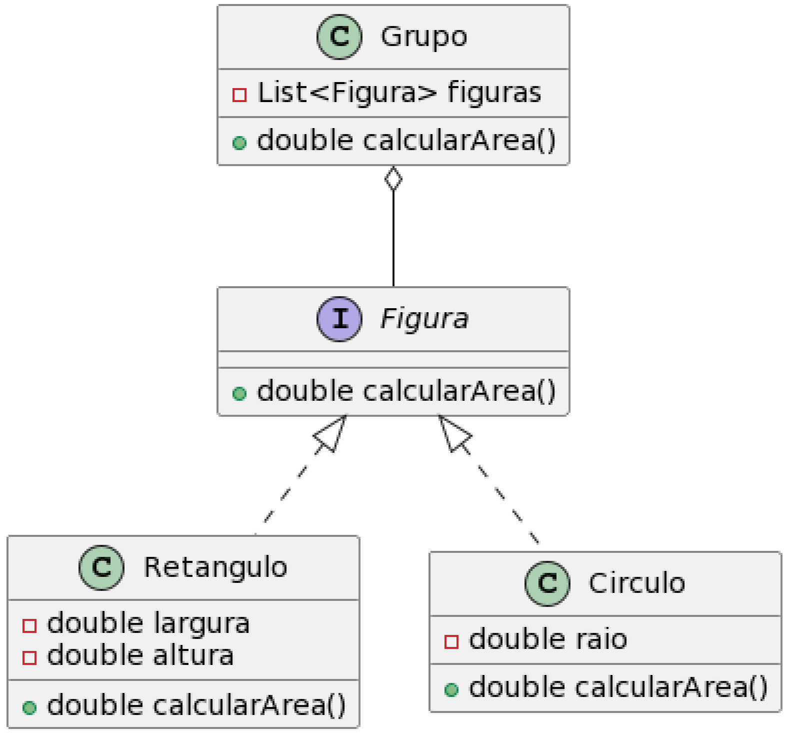
Composite



File Edit Source Refactor Navigate Search Project Run Window Help
GrupoAntes.java

```java
package school.cesar.esa.padroes.estruturais.composite;

import java.util.ArrayList;

public class GrupoAntes {
    private List<Figura> figuras;

    public GrupoAntes() {
        figuras = new ArrayList<>();
    }

    public void adicionar(Figura figura) {
        if (figura == null) {
            throw new IllegalArgumentException("figura == null");
        }
        figuras.add(figura);
    }

    public double calcularArea() {
        double area = 0;
        for (Figura figura : figuras) {
            area += figura.calcularArea();
        }
        return area;
    }
}
```

Composite

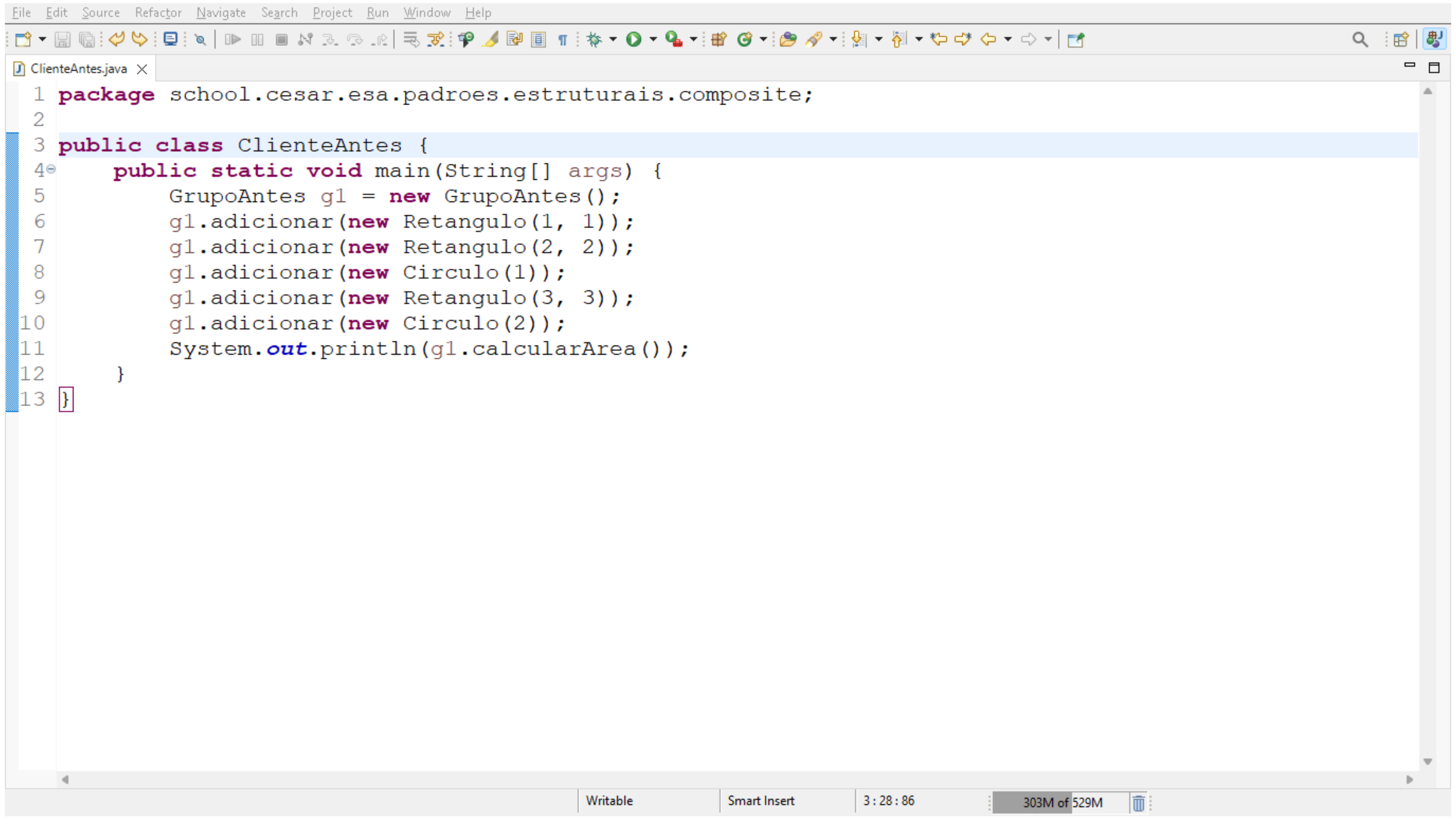


C Grupo
List<Figura> figuras
double calcularArea()
I Figura
double calcularArea()
C Retangulo
double largura
double altura
double calcularArea()
C Circulo
double raio
double calcularArea()

Composite

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

Composite



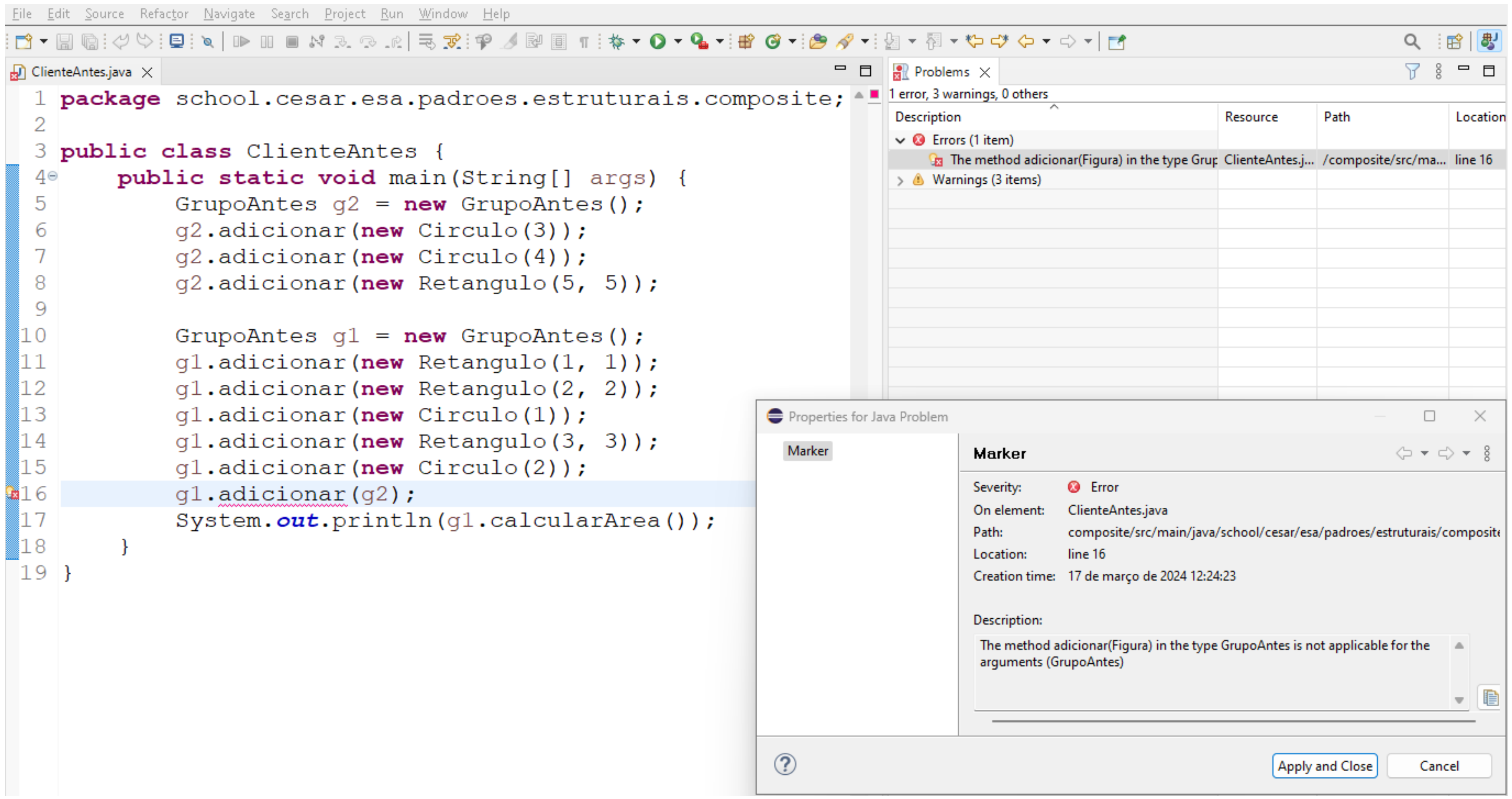
File Edit Source Refactor Navigate Search Project Run Window Help
ClienteAntes.java
Writable
Smart Insert
3 : 28 : 86
303M of 529M

```java
package school.cesar.esa.padroes.estruturais.composite;

public class ClienteAntes {
    public static void main(String[] args) {
        GrupoAntes g1 = new GrupoAntes();
        g1.adicionar(new Retangulo(1, 1));
        g1.adicionar(new Retangulo(2, 2));
        g1.adicionar(new Circulo(1));
        g1.adicionar(new Retangulo(3, 3));
        g1.adicionar(new Circulo(2));
        System.out.println(g1.calcularArea());
    }
}
```

Composite

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

Composite



```
package school.cesar.esa.padroes.estruturais.composite;

public class ClienteAntes {
    public static void main(String[] args) {
        GrupoAntes g2 = new GrupoAntes();
        g2.adicionar(new Circulo(3));
        g2.adicionar(new Circulo(4));
        g2.adicionar(new Retangulo(5, 5));

        GrupoAntes g1 = new GrupoAntes();
        g1.adicionar(new Retangulo(1, 1));
        g1.adicionar(new Retangulo(2, 2));
        g1.adicionar(new Circulo(1));
        g1.adicionar(new Retangulo(3, 3));
        g1.adicionar(new Circulo(2));
        g1.adicionar(g2);
        System.out.println(g1.calcularArea());
    }
}
```

Problems

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

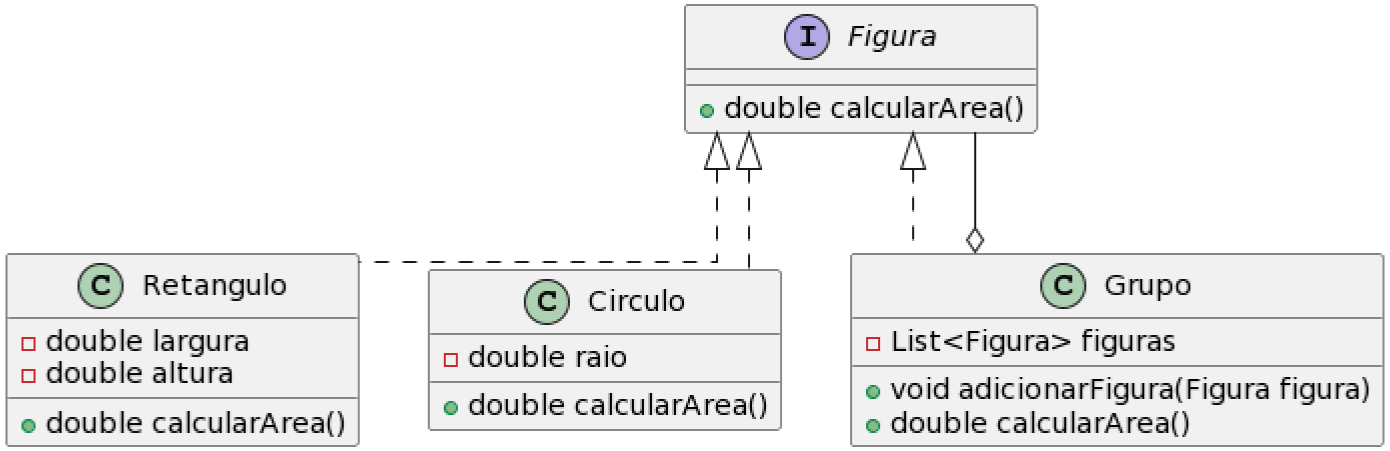
Marker

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:

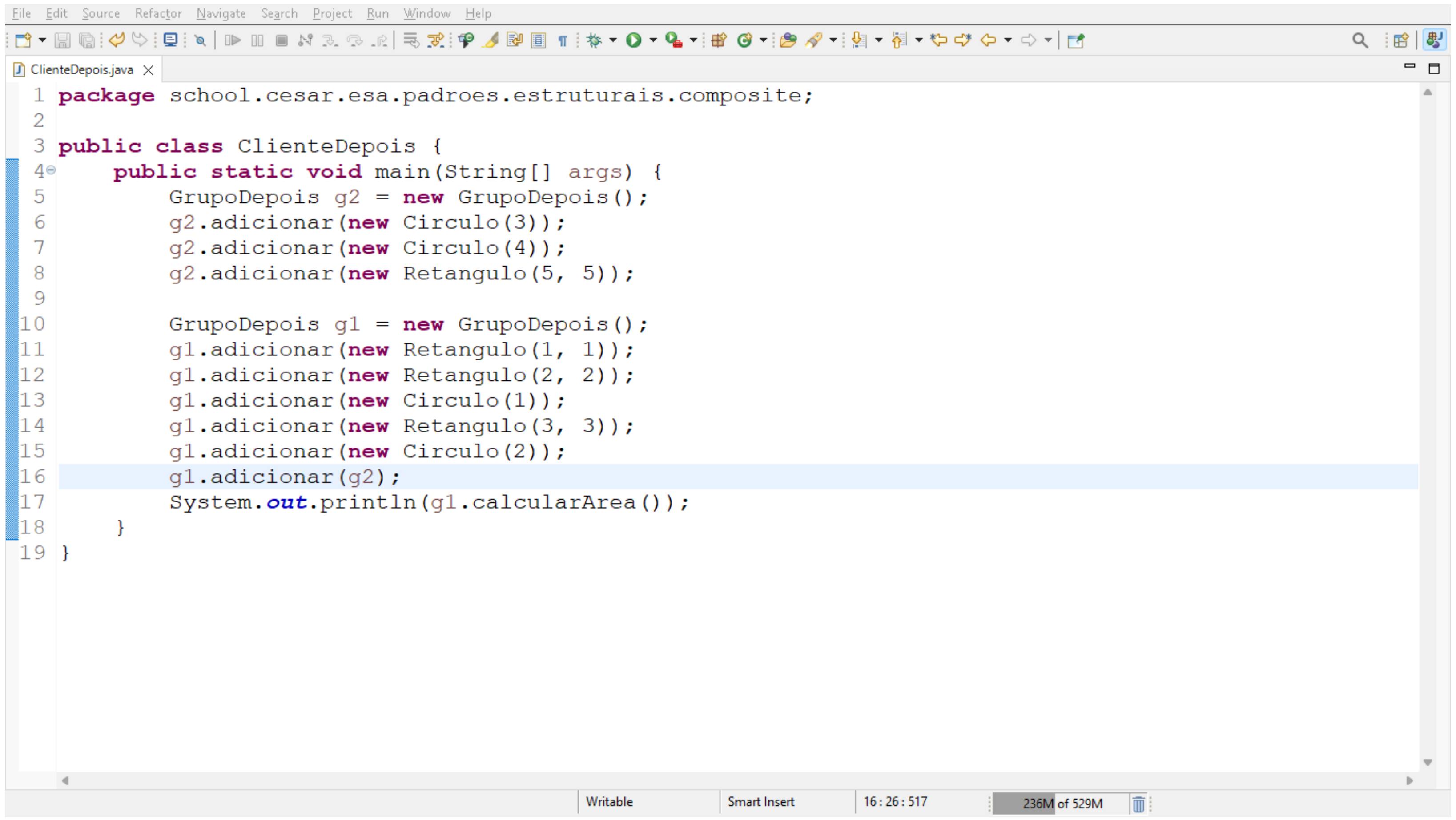
The method adicionar(Figura) in the type GrupoAntes is not applicable for the arguments (GrupoAntes)

Composite



I Figura
double calcularArea()
C Retangulo
double largura
double altura
double calcularArea()
C Circulo
double raio
double calcularArea()
C Grupo
List<Figura> figuras
void adicionarFigura(Figura figura)
double calcularArea()

Composite

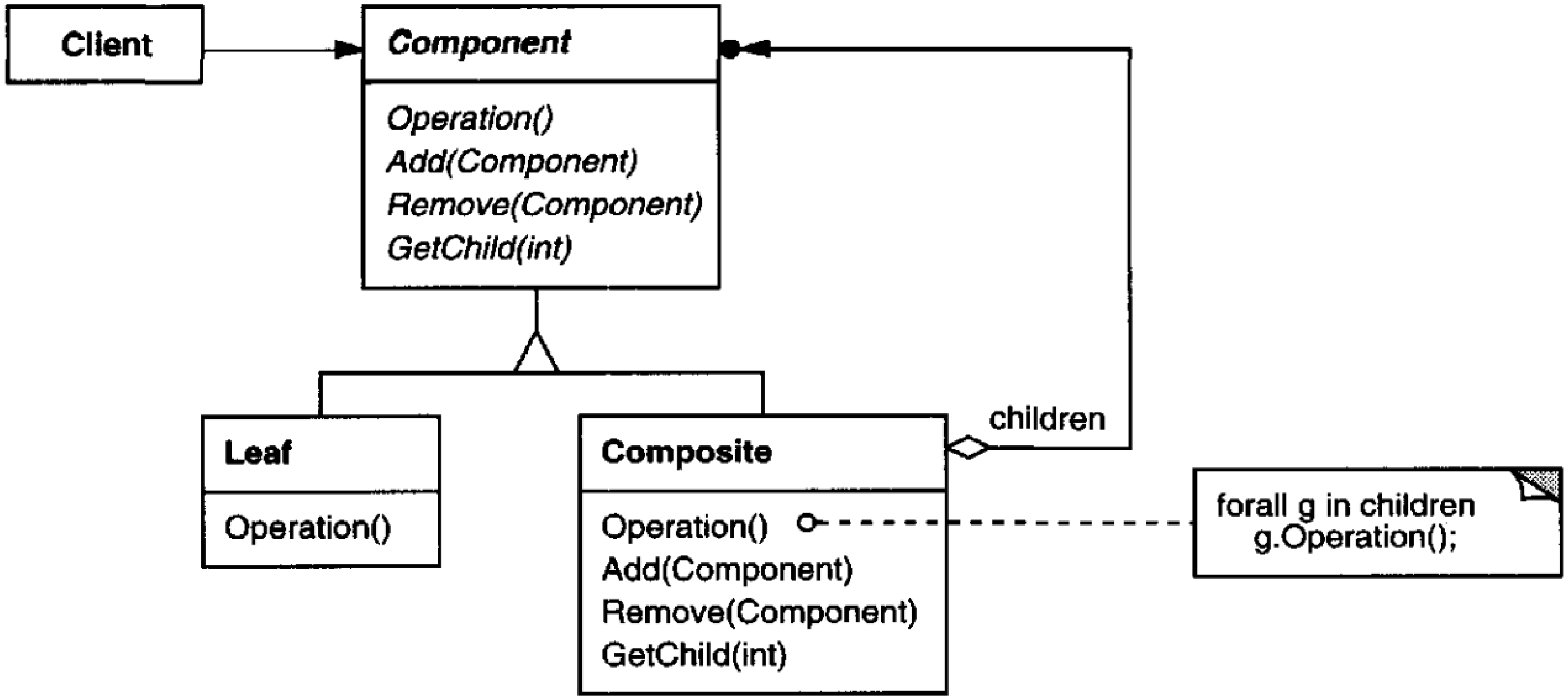

File Edit Source Refactor Navigate Search Project Run Window Help
ClienteDepois.java

```java
package school.cesar.esa.padroes.estruturais.composite;

public class ClienteDepois {
    public static void main(String[] args) {
        GrupoDepois g2 = new GrupoDepois();
        g2.adicionar(new Circulo(3));
        g2.adicionar(new Circulo(4));
        g2.adicionar(new Retangulo(5, 5));

        GrupoDepois g1 = new GrupoDepois();
        g1.adicionar(new Retangulo(1, 1));
        g1.adicionar(new Retangulo(2, 2));
        g1.adicionar(new Circulo(1));
        g1.adicionar(new Retangulo(3, 3));
        g1.adicionar(new Circulo(2));
        g1.adicionar(g2);
        System.out.println(g1.calcularArea());
    }
}
```

Composite

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

Composite


Client
Component
Operation()
Add(Component)
Remove(Component)
GetChild(int)
children
Leaf
Operation()
Composite
Operation()
Add(Component)
Remove(Component)
GetChild(int)
forall g in children
g.Operation();

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)

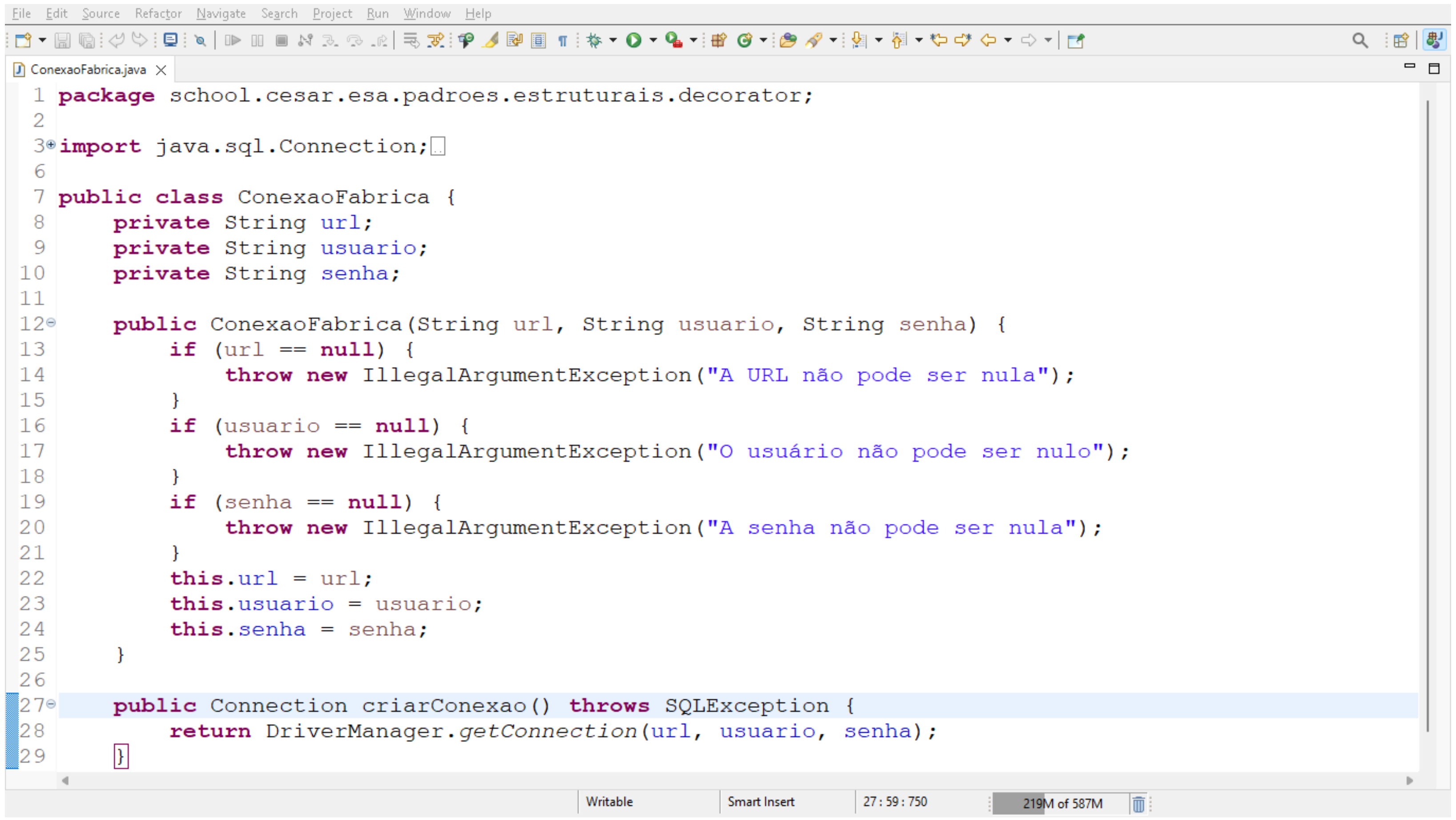
- Prefira composição à herança
- Demeter

Composite

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

Decorator

Decorator



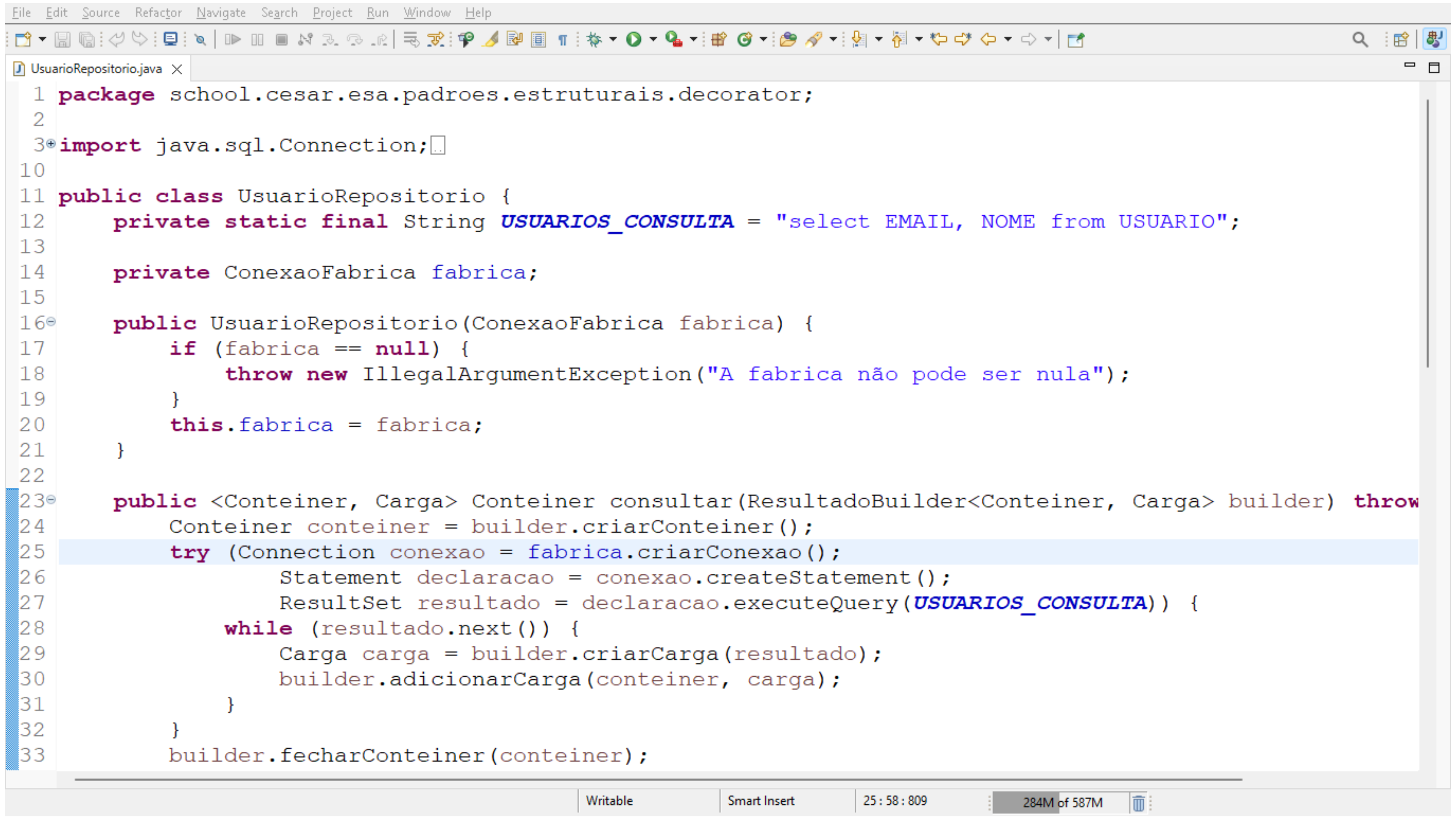
File Edit Source Refactor Navigate Search Project Run Window Help

ConexaoFabrica.java

```java
package school.cesar.esa.padroes.estruturais.decorator;

import java.sql.Connection;

public class ConexaoFabrica {
    private String url;
    private String usuario;
    private String senha;

    public ConexaoFabrica(String url, String usuario, String senha) {
        if (url == null) {
            throw new IllegalArgumentException("A URL não pode ser nula");
        }
        if (usuario == null) {
            throw new IllegalArgumentException("O usuário não pode ser nulo");
        }
        if (senha == null) {
            throw new IllegalArgumentException("A senha não pode ser nula");
        }
        this.url = url;
        this.usuario = usuario;
        this.senha = senha;
    }

    public Connection criarConexao() throws SQLException {
        return DriverManager.getConnection(url, usuario, senha);
    }
```

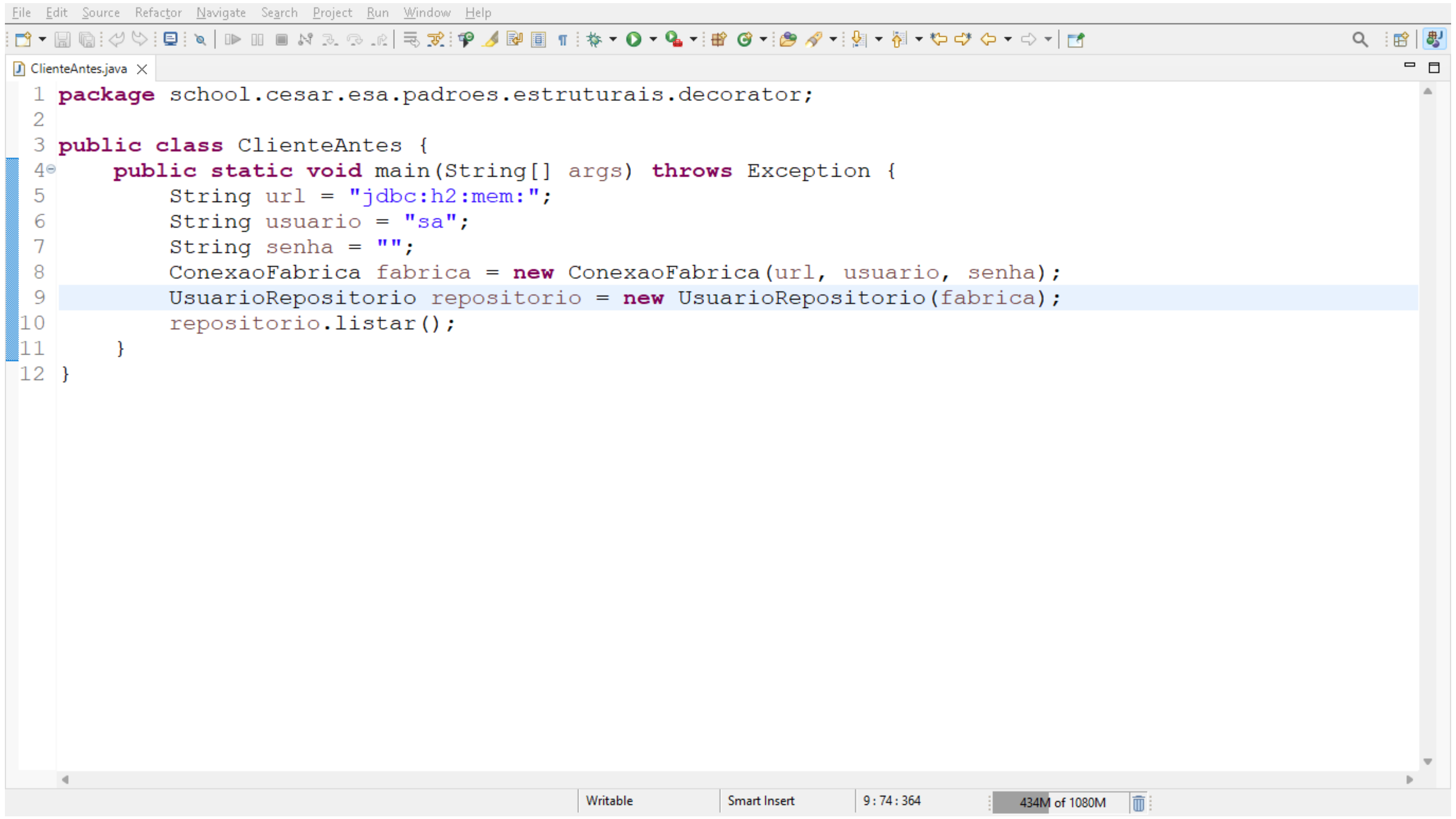
Writable Smart Insert 27 : 59 : 750 219M of 587M

Decorator



```java
package school.cesar.esa.padroes.estruturais.decorator;

import java.sql.Connection;

public class UsuarioRepositorio {
    private static final String USUARIOS_CONSULTA = "select EMAIL, NOME from USUARIO";

    private ConexaoFabrica fabrica;

    public UsuarioRepositorio(ConexaoFabrica fabrica) {
        if (fabrica == null) {
            throw new IllegalArgumentException("A fabrica não pode ser nula");
        }
        this.fabrica = fabrica;
    }

    public <Conteiner, Carga> Conteiner consultar(ResultadoBuilder<Conteiner, Carga> builder) throw
        Conteiner conteiner = builder.criarConteiner();
        try (Connection conexao = fabrica.criarConexao();
                Statement declaracao = conexao.createStatement();
                ResultSet resultado = declaracao.executeQuery(USUARIOS_CONSULTA)) {
            while (resultado.next()) {
                Carga carga = builder.criarCarga(resultado);
                builder.adicionarCarga(conteiner, carga);
            }
        }
        builder.fecharConteiner(conteiner);
```

Decorator


File Edit Source Refactor Navigate Search Project Run Window Help
ClienteAntes.java
Writable
Smart Insert
9 : 74 : 364
434M of 1080M

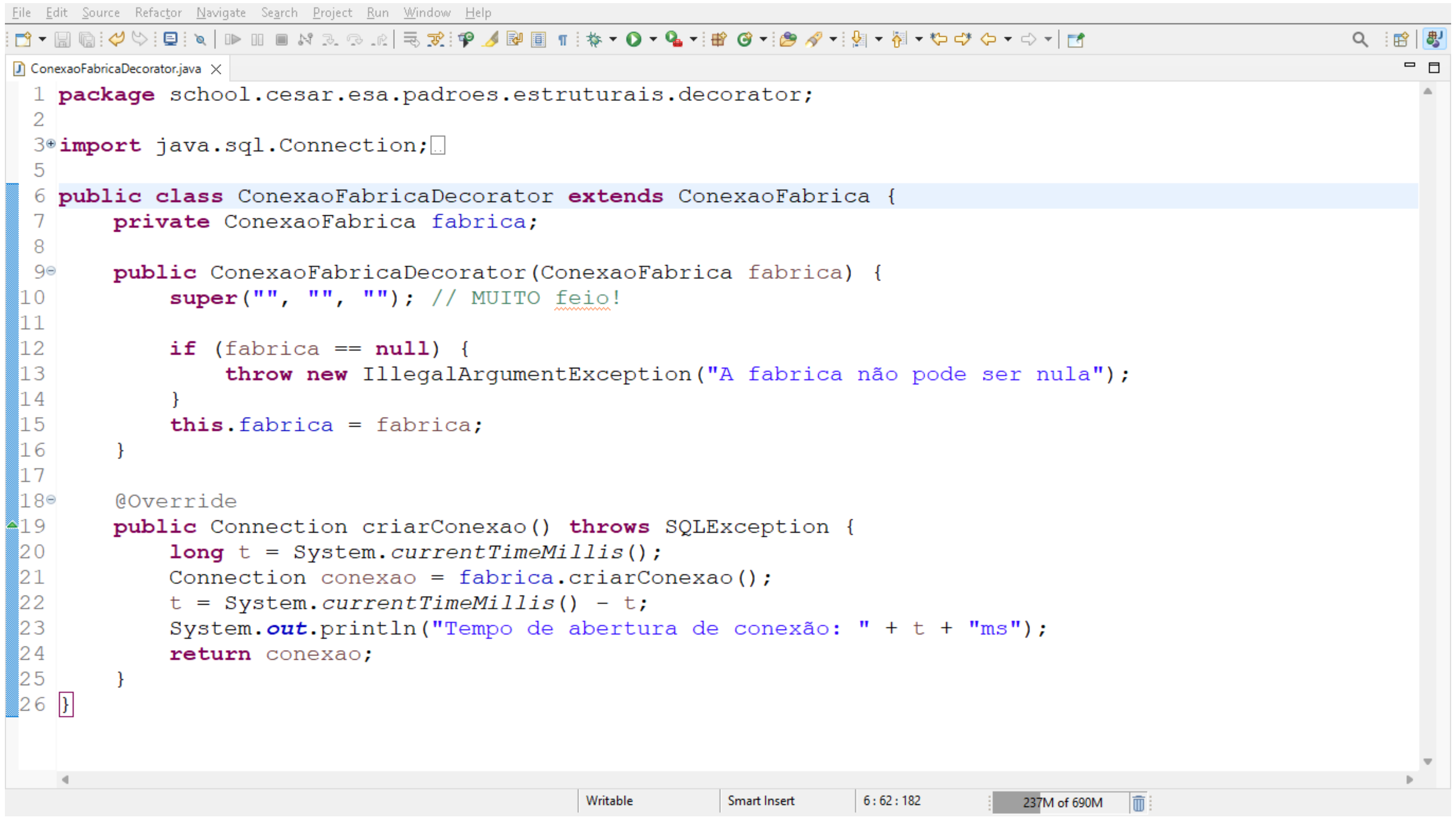
```java
package school.cesar.esa.padroes.estruturais.decorator;

public class ClienteAntes {
    public static void main(String[] args) throws Exception {
        String url = "jdbc:h2:mem:";
        String usuario = "sa";
        String senha = "";
        ConexaoFabrica fabrica = new ConexaoFabrica(url, usuario, senha);
        UsuarioRepositorio repositorio = new UsuarioRepositorio(fabrica);
        repositorio.listar();
    }
}
```

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`?

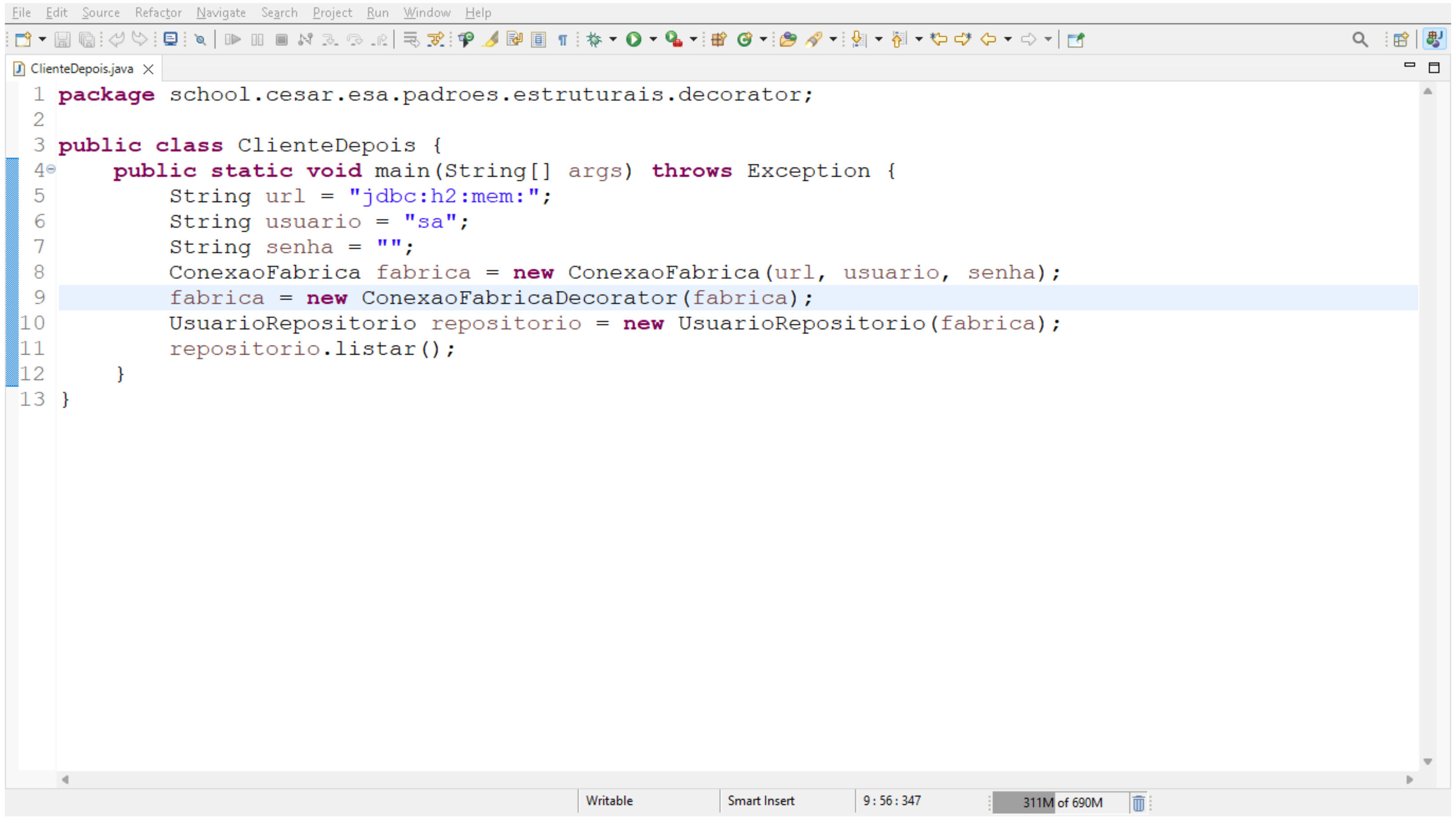
Decorator

Decorator


File Edit Source Refactor Navigate Search Project Run Window Help
ConexaoFabricaDecorator.java
Writable
Smart Insert
6 : 62 : 182
237M of 690M

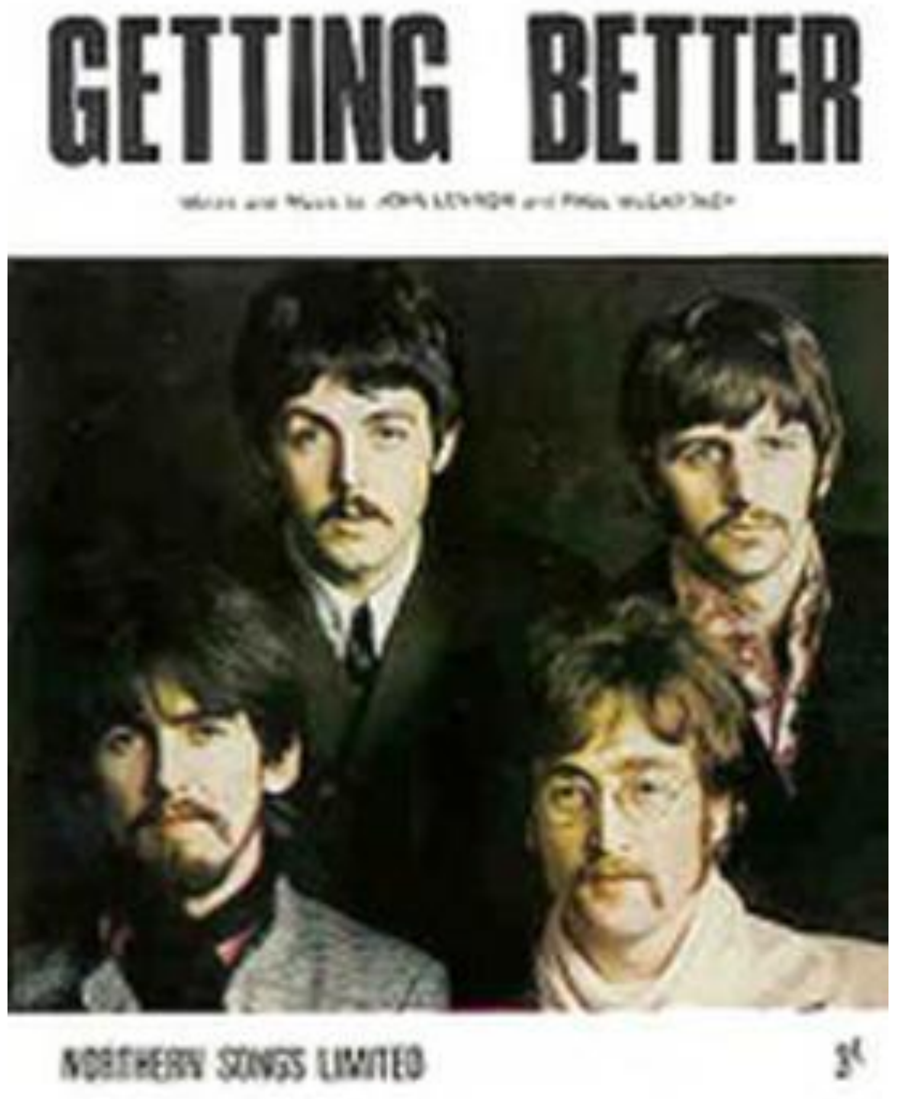
```java
package school.cesar.esa.padroes.estruturais.decorator;

import java.sql.Connection;

public class ConexaoFabricaDecorator extends ConexaoFabrica {
    private ConexaoFabrica fabrica;

    public ConexaoFabricaDecorator(ConexaoFabrica fabrica) {
        super("", "", ""); // MUITO feio!

        if (fabrica == null) {
            throw new IllegalArgumentException("A fabrica não pode ser nula");
        }
        this.fabrica = fabrica;
    }

    @Override
    public Connection criarConexao() throws SQLException {
        long t = System.currentTimeMillis();
        Connection conexao = fabrica.criarConexao();
        t = System.currentTimeMillis() - t;
        System.out.println("Tempo de abertura de conexão: " + t + "ms");
        return conexao;
    }
}
```

Decorator

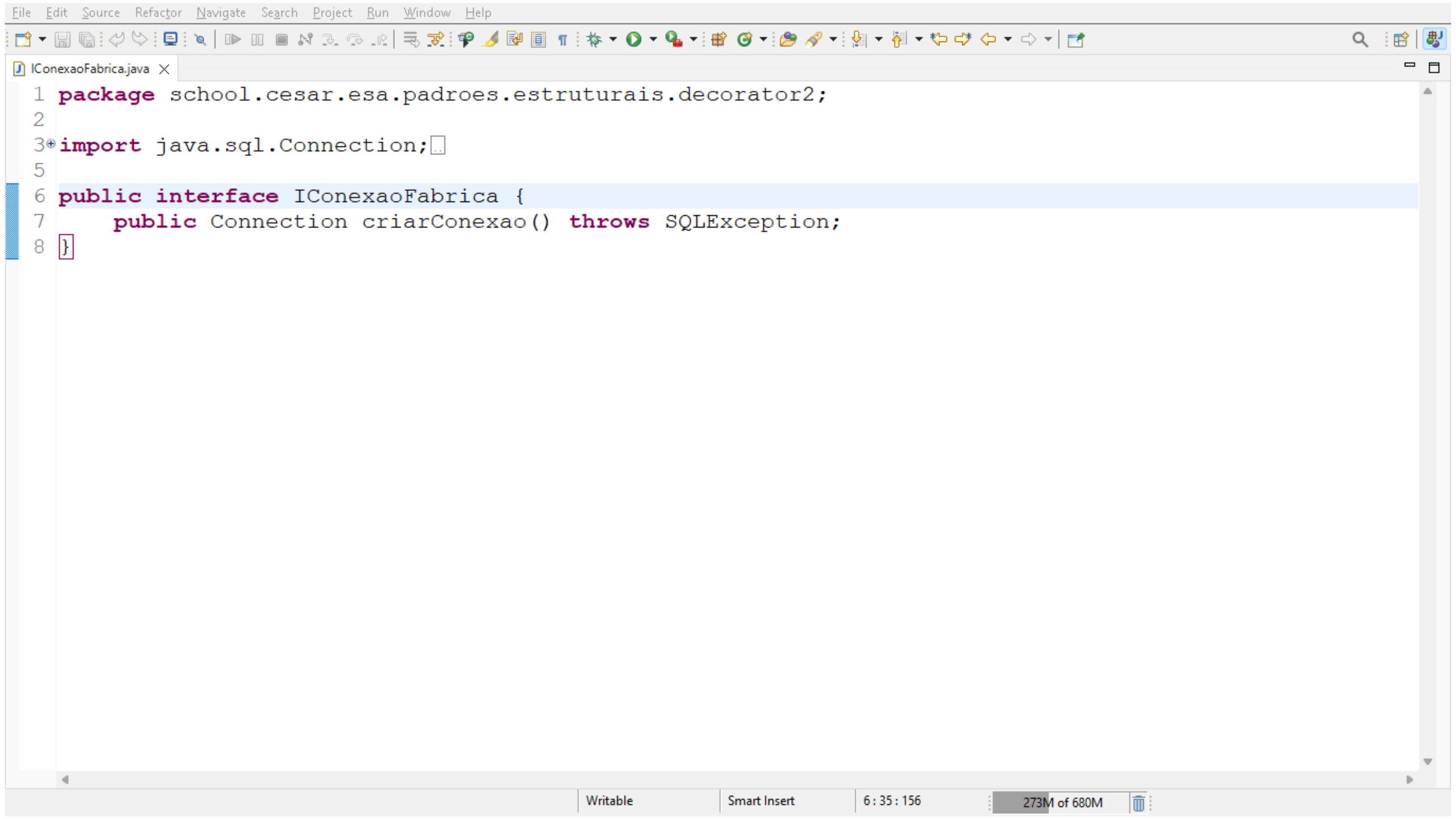

File Edit Source Refactor Navigate Search Project Run Window Help
ClienteDepois.java
Writable
Smart Insert
9 : 56 : 347
311M of 690M

```java
package school.cesar.esa.padroes.estruturais.decorator;

public class ClienteDepois {
    public static void main(String[] args) throws Exception {
        String url = "jdbc:h2:mem:";
        String usuario = "sa";
        String senha = "";
        ConexaoFabrica fabrica = new ConexaoFabrica(url, usuario, senha);
        fabrica = new ConexaoFabricaDecorator(fabrica);
        UsuarioRepositorio repositorio = new UsuarioRepositorio(fabrica);
        repositorio.listar();
    }
}
```

Decorator

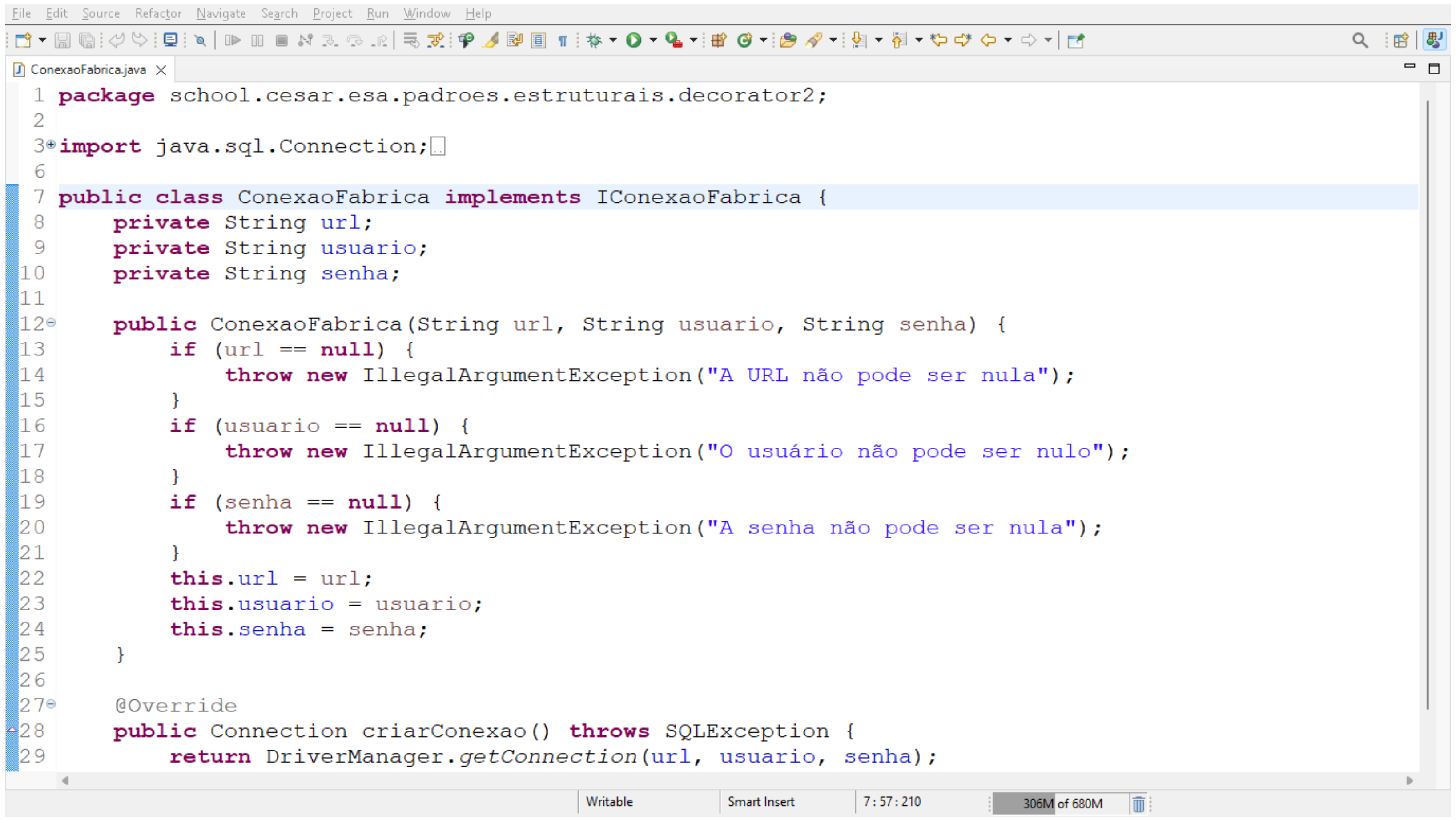


Decorator


File Edit Source Refactor Navigate Search Project Run Window Help
IConexaoFabrica.java
Writable
Smart Insert
6 : 35 : 156
273M of 680M

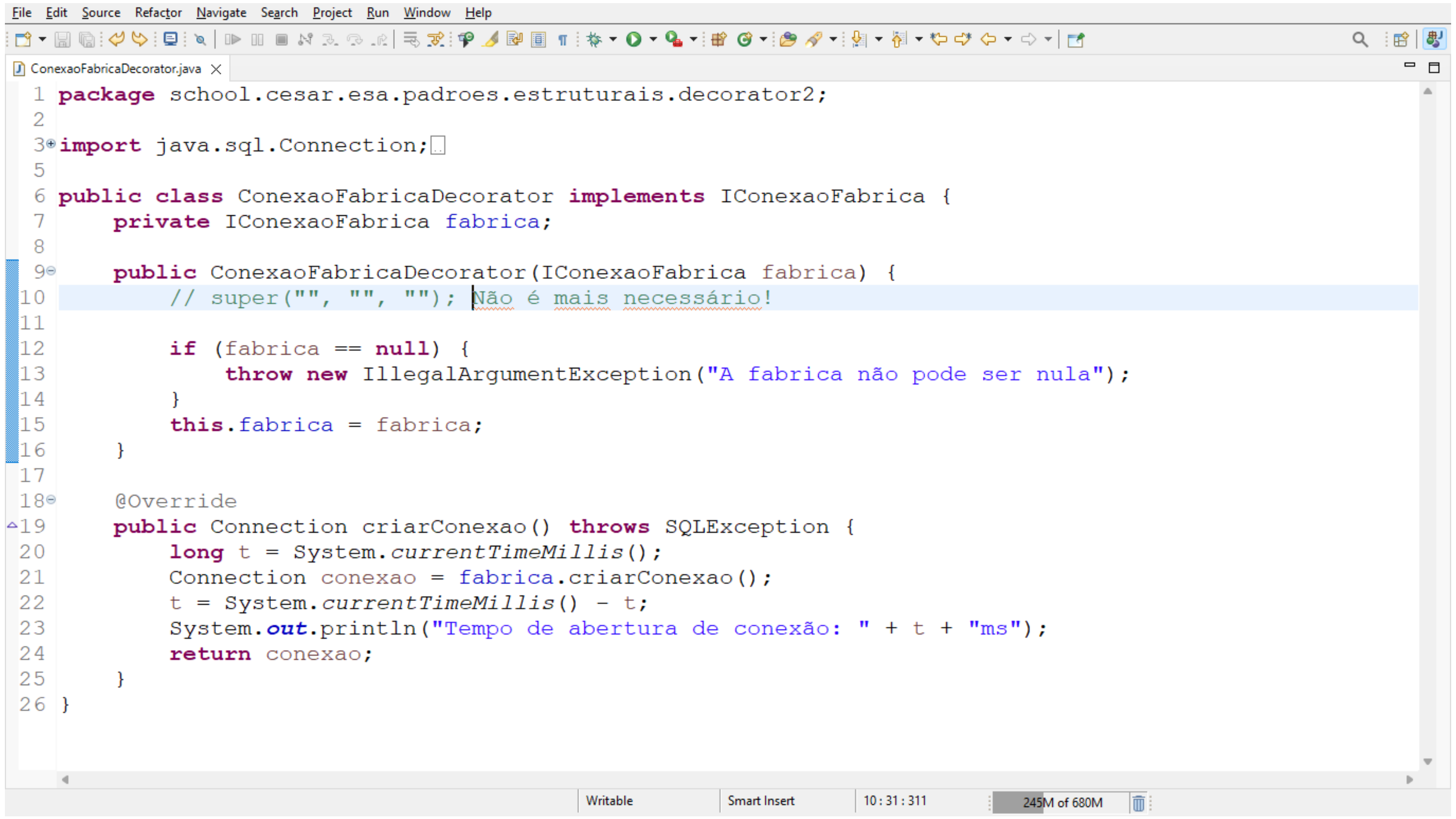
```java
package school.cesar.esa.padroes.estruturais.decorator2;

import java.sql.Connection;

public interface IConexaoFabrica {
    public Connection criarConexao() throws SQLException;
}
```

Decorator


File Edit Source Refactor Navigate Search Project Run Window Help
ConexaoFabrica.java
Writable
Smart Insert
7 : 57 : 210
306M of 680M

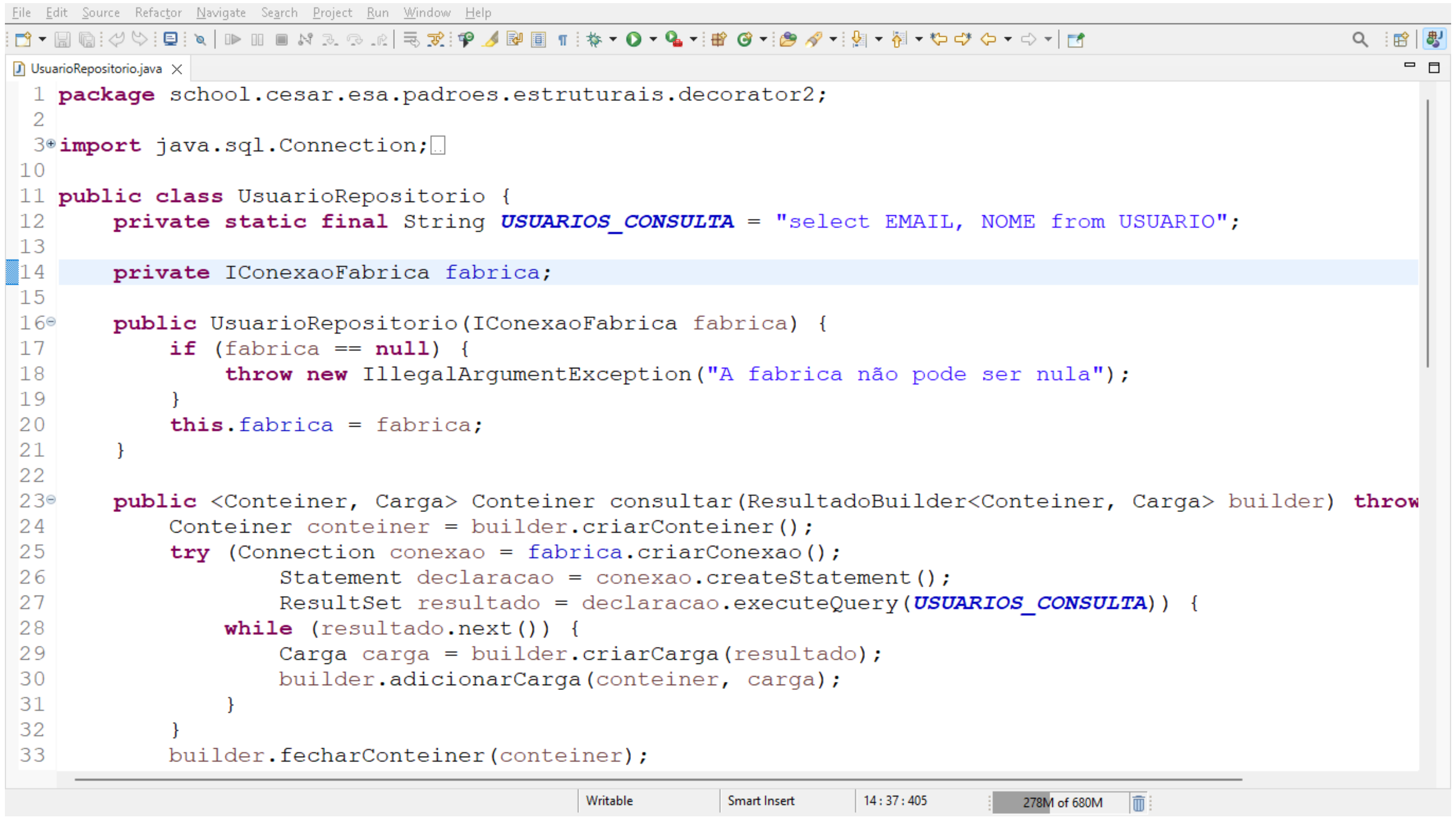
```java
package school.cesar.esa.padroes.estruturais.decorator2;

import java.sql.Connection;

public class ConexaoFabrica implements IConexaoFabrica {
    private String url;
    private String usuario;
    private String senha;

    public ConexaoFabrica(String url, String usuario, String senha) {
        if (url == null) {
            throw new IllegalArgumentException("A URL não pode ser nula");
        }
        if (usuario == null) {
            throw new IllegalArgumentException("O usuário não pode ser nulo");
        }
        if (senha == null) {
            throw new IllegalArgumentException("A senha não pode ser nula");
        }
        this.url = url;
        this.usuario = usuario;
        this.senha = senha;
    }

    @Override
    public Connection criarConexao() throws SQLException {
        return DriverManager.getConnection(url, usuario, senha);
```

Decorator


File Edit Source Refactor Navigate Search Project Run Window Help
ConexaoFabricaDecorator.java
Writable
Smart Insert
10 : 31 : 311
245M of 680M

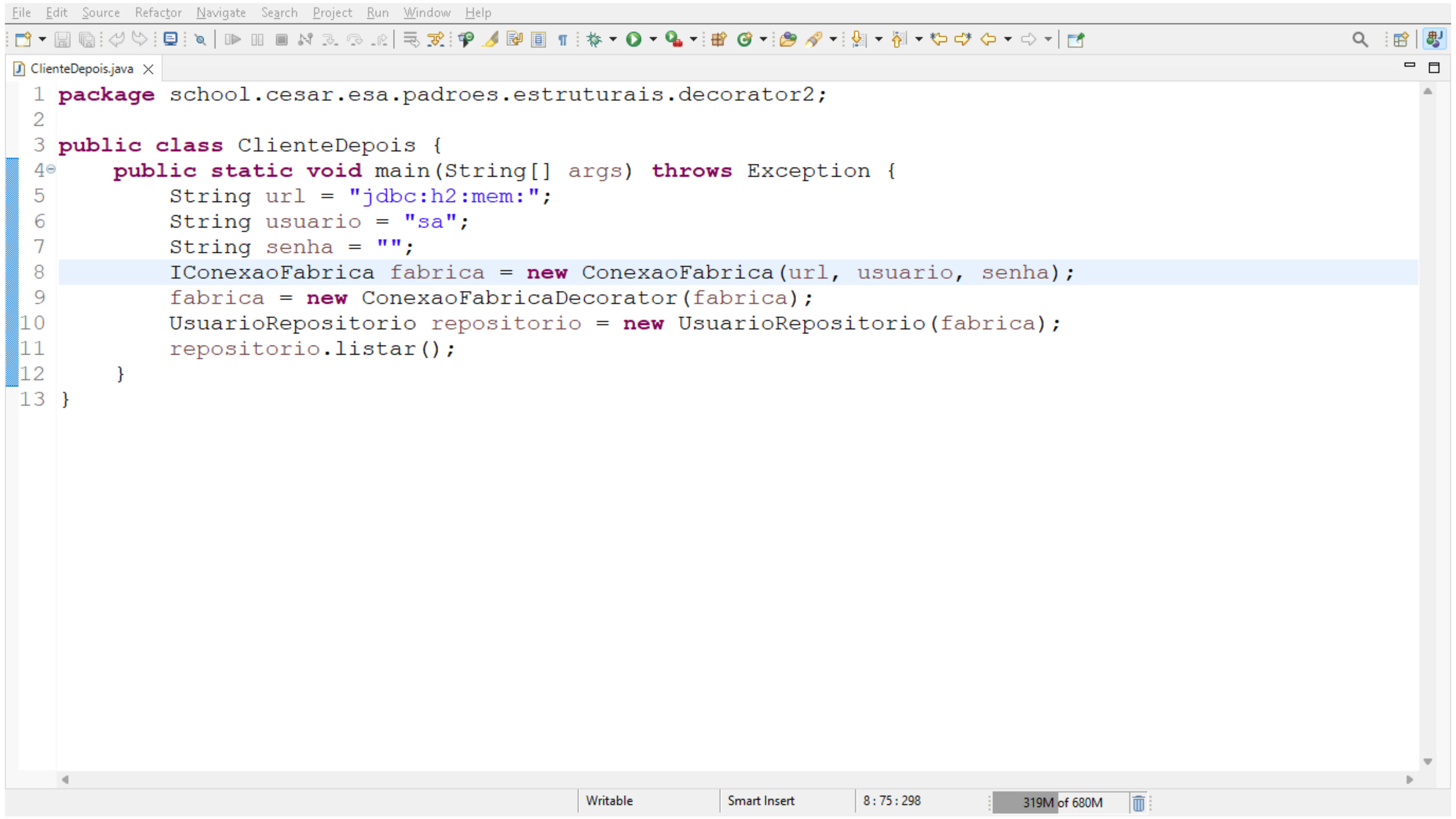
```java
package school.cesar.esa.padroes.estruturais.decorator2;

import java.sql.Connection;

public class ConexaoFabricaDecorator implements IConexaoFabrica {
    private IConexaoFabrica fabrica;

    public ConexaoFabricaDecorator(IConexaoFabrica fabrica) {
        // super("", "", ""); Não é mais necessário!

        if (fabrica == null) {
            throw new IllegalArgumentException("A fabrica não pode ser nula");
        }
        this.fabrica = fabrica;
    }

    @Override
    public Connection criarConexao() throws SQLException {
        long t = System.currentTimeMillis();
        Connection conexao = fabrica.criarConexao();
        t = System.currentTimeMillis() - t;
        System.out.println("Tempo de abertura de conexão: " + t + "ms");
        return conexao;
    }
}
```

Decorator



```java
package school.cesar.esa.padroes.estruturais.decorator2;

import java.sql.Connection;

public class UsuarioRepositorio {
    private static final String USUARIOS_CONSULTA = "select EMAIL, NOME from USUARIO";

    private IConexaoFabrica fabrica;

    public UsuarioRepositorio(IConexaoFabrica fabrica) {
        if (fabrica == null) {
            throw new IllegalArgumentException("A fabrica não pode ser nula");
        }
        this.fabrica = fabrica;
    }

    public <Conteiner, Carga> Conteiner consultar(ResultadoBuilder<Conteiner, Carga> builder) throw
        Conteiner conteiner = builder.criarConteiner();
        try (Connection conexao = fabrica.criarConexao();
                Statement declaracao = conexao.createStatement();
                ResultSet resultado = declaracao.executeQuery(USUARIOS_CONSULTA)) {
            while (resultado.next()) {
                Carga carga = builder.criarCarga(resultado);
                builder.adicionarCarga(conteiner, carga);
            }
        }
        builder.fecharConteiner(conteiner);
```

Decorator

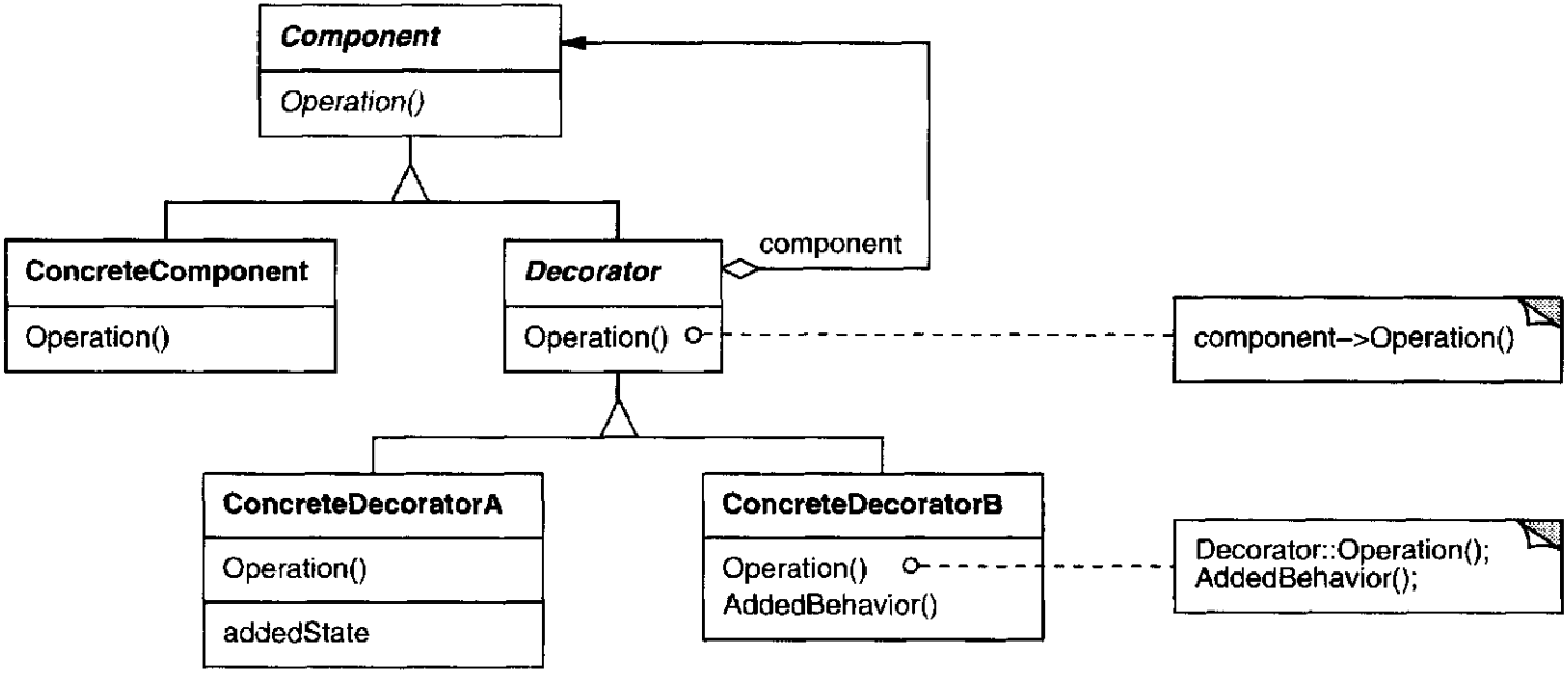

File Edit Source Refactor Navigate Search Project Run Window Help
ClienteDepois.java
Writable
Smart Insert
8 : 75 : 298
319M of 680M

```java
package school.cesar.esa.padroes.estruturais.decorator2;

public class ClienteDepois {
    public static void main(String[] args) throws Exception {
        String url = "jdbc:h2:mem:";
        String usuario = "sa";
        String senha = "";
        IConexaoFabrica fabrica = new ConexaoFabrica(url, usuario, senha);
        fabrica = new ConexaoFabricaDecorator(fabrica);
        UsuarioRepositorio repositorio = new UsuarioRepositorio(fabrica);
        repositorio.listar();
    }
}
```

Decorator

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

Decorator


Component
Operation()
component
ConcreteComponent
Operation()
Decorator
Operation()
component->Operation()
ConcreteDecoratorA
Operation()
addedState
ConcreteDecoratorB
Operation()
AddedBehavior()
Decorator::Operation();
AddedBehavior();

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)

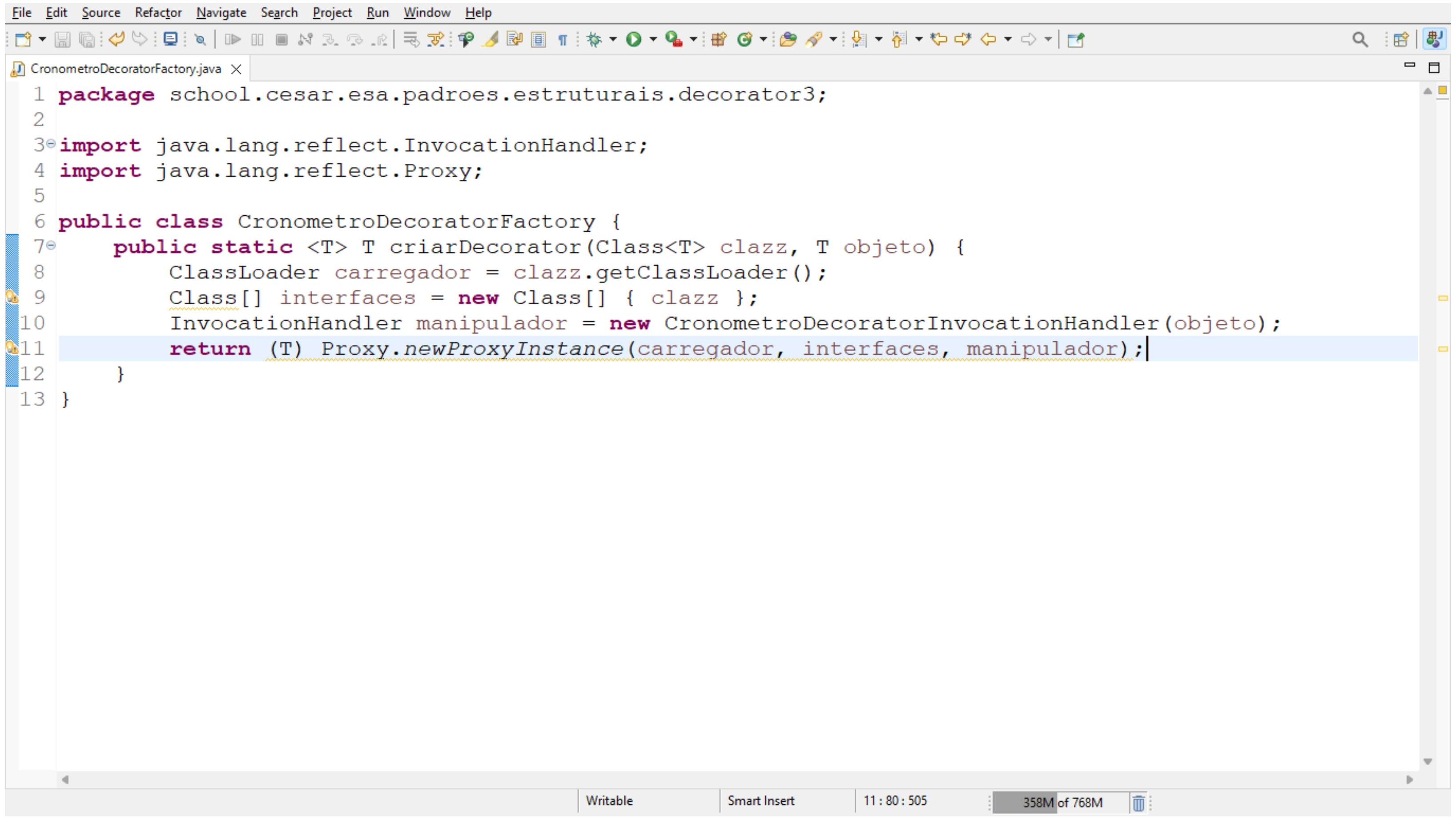
- Prefira composição à herança
- Demeter

Decorator

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

Decorator

Decorator



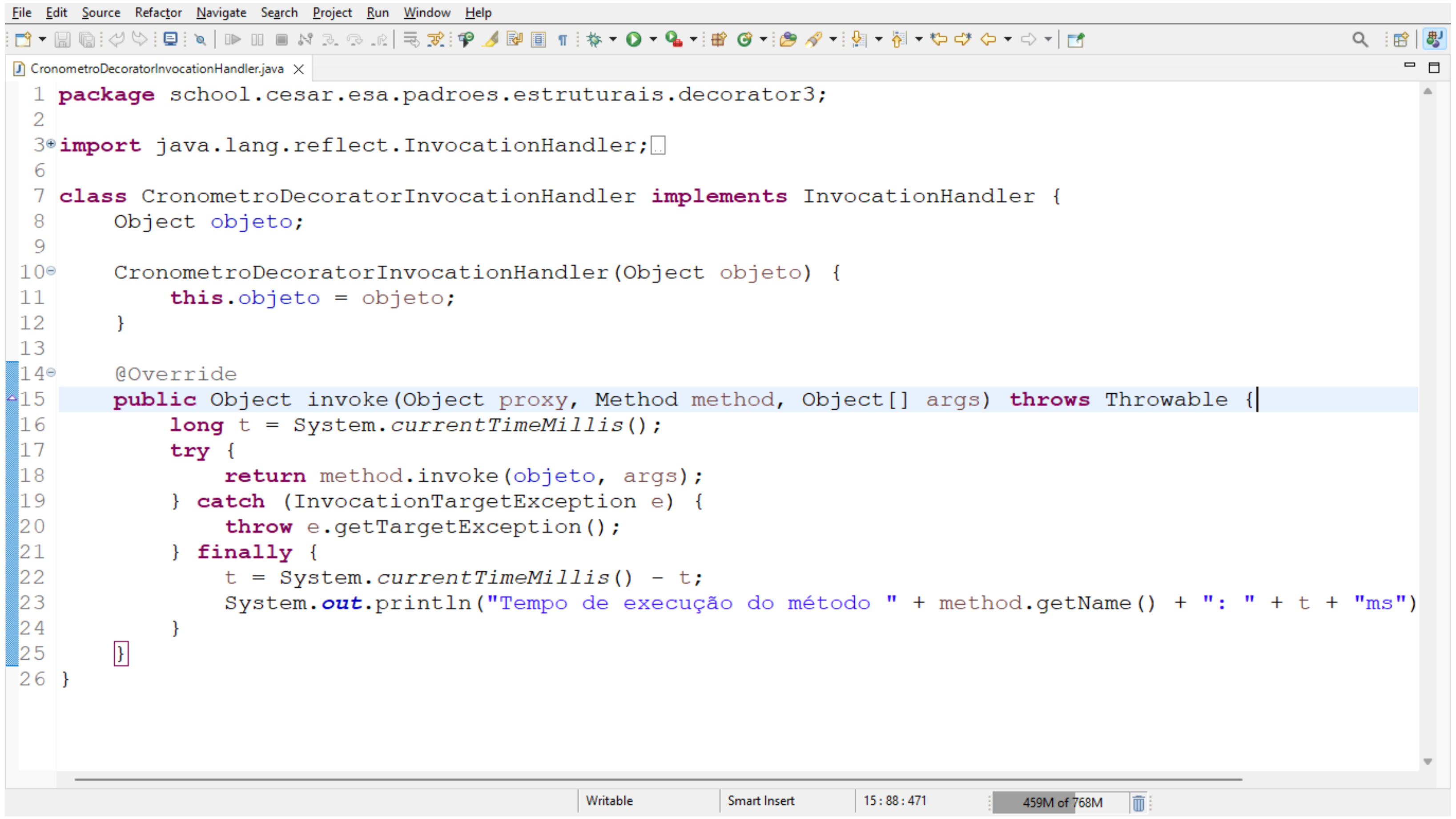
File Edit Source Refactor Navigate Search Project Run Window Help

CronometroDecoratorFactory.java

```java
package school.cesar.esa.padroes.estruturais.decorator3;

import java.lang.reflect.InvocationHandler;
import java.lang.reflect.Proxy;

public class CronometroDecoratorFactory {
    public static <T> T criarDecorator(Class<T> clazz, T objeto) {
        ClassLoader carregador = clazz.getClassLoader();
        Class[] interfaces = new Class[] { clazz };
        InvocationHandler manipulador = new CronometroDecoratorInvocationHandler(objeto);
        return (T) Proxy.newProxyInstance(carregador, interfaces, manipulador);
    }
}
```

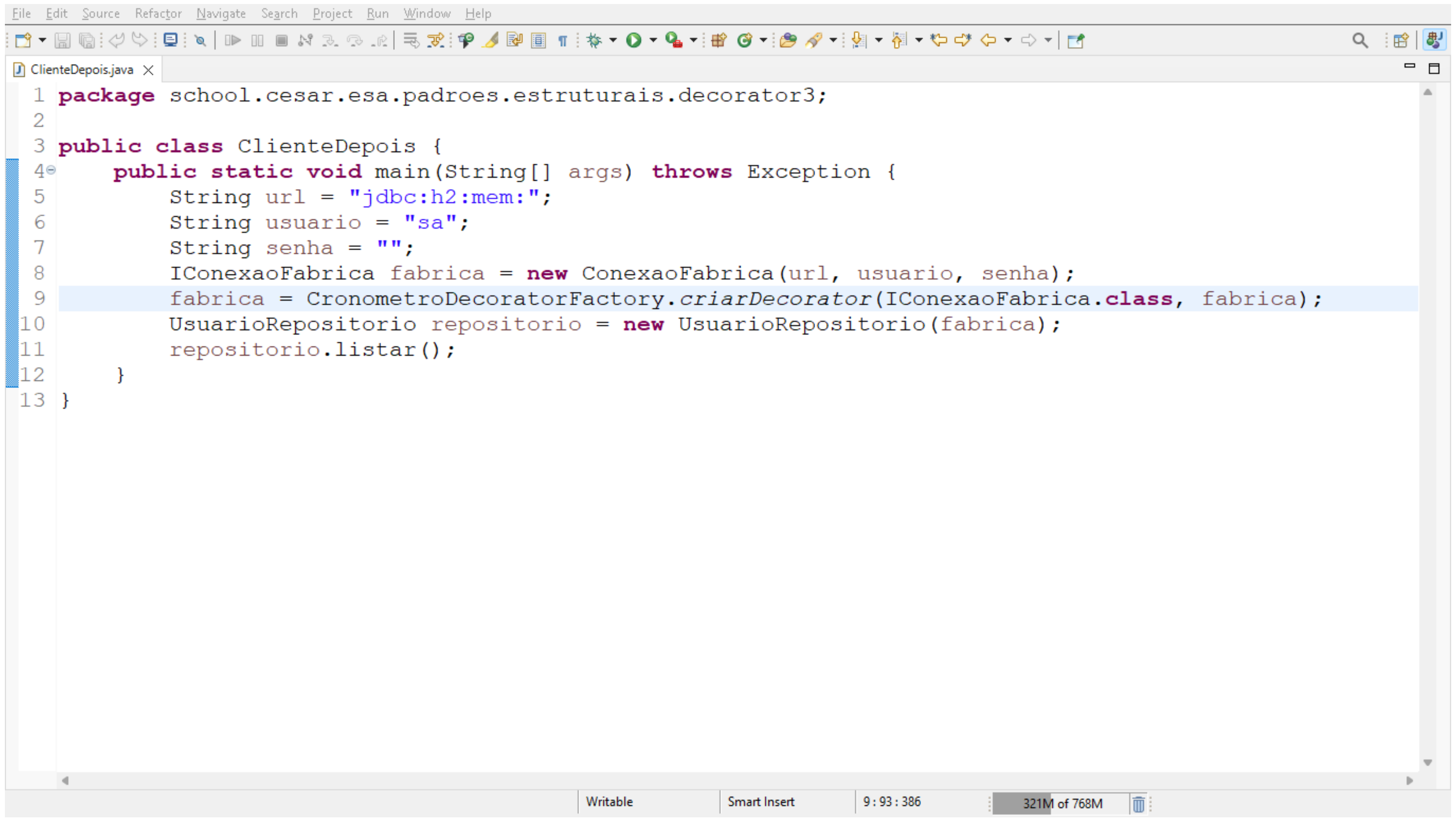
Writable | Smart Insert | 11 : 80 : 505 | 358M of 768M

Decorator


File Edit Source Refactor Navigate Search Project Run Window Help
CronometroDecoratorInvocationHandler.java
Writable
Smart Insert
15 : 88 : 471
459M of 768M

```java
package school.cesar.esa.padroes.estruturais.decorator3;

import java.lang.reflect.InvocationHandler;

class CronometroDecoratorInvocationHandler implements InvocationHandler {
    Object objeto;

    CronometroDecoratorInvocationHandler(Object objeto) {
        this.objeto = objeto;
    }

    @Override
    public Object invoke(Object proxy, Method method, Object[] args) throws Throwable {
        long t = System.currentTimeMillis();
        try {
            return method.invoke(objeto, args);
        } catch (InvocationTargetException e) {
            throw e.getTargetException();
        } finally {
            t = System.currentTimeMillis() - t;
            System.out.println("Tempo de execução do método " + method.getName() + ": " + t + "ms")
        }
    }
}
```

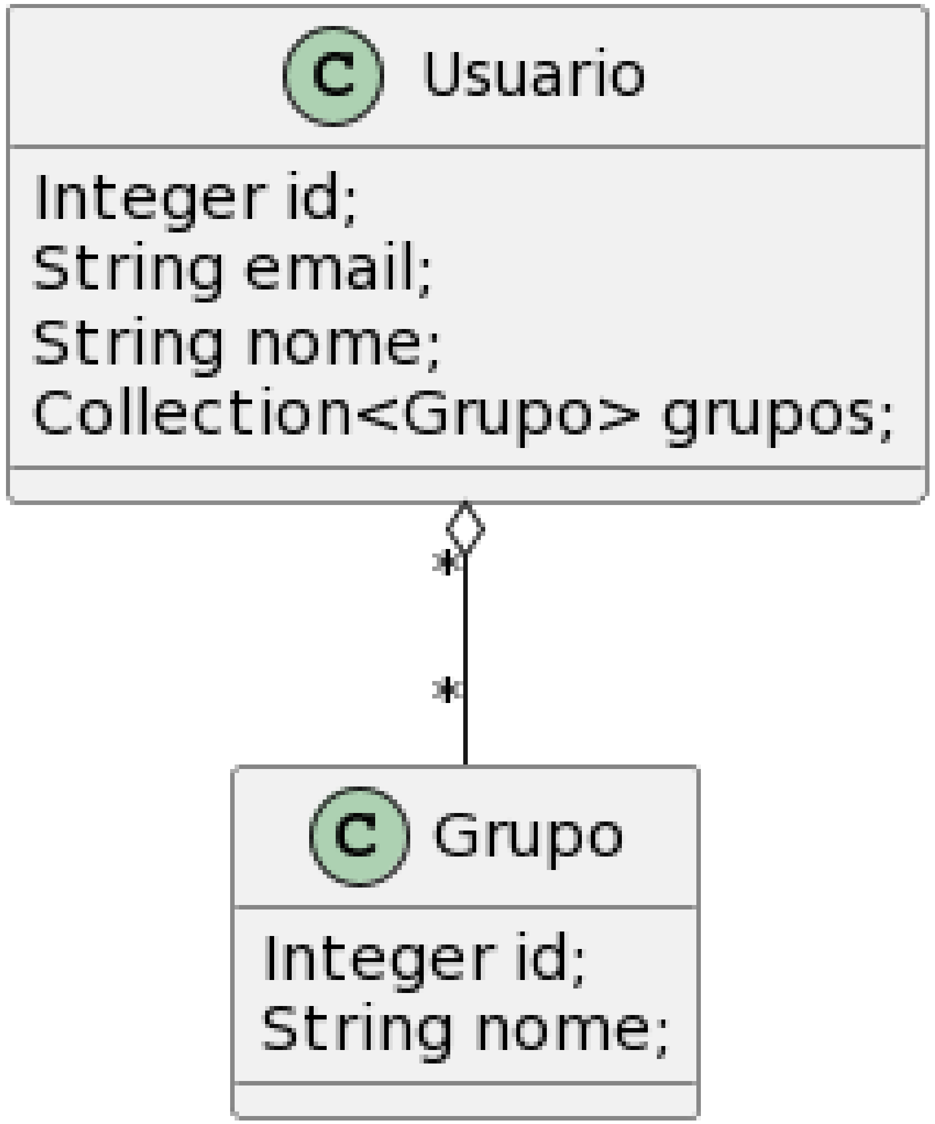
Decorator


File Edit Source Refactor Navigate Search Project Run Window Help
ClienteDepois.java
Writable
Smart Insert
9 : 93 : 386
321M of 768M

```java
package school.cesar.esa.padroes.estruturais.decorator3;

public class ClienteDepois {
    public static void main(String[] args) throws Exception {
        String url = "jdbc:h2:mem:";
        String usuario = "sa";
        String senha = "";
        IConexaoFabrica fabrica = new ConexaoFabrica(url, usuario, senha);
        fabrica = CronometroDecoratorFactory.criarDecorator(IConexaoFabrica.class, fabrica);
        UsuarioRepositorio repositorio = new UsuarioRepositorio(fabrica);
        repositorio.listar();
    }
}
```

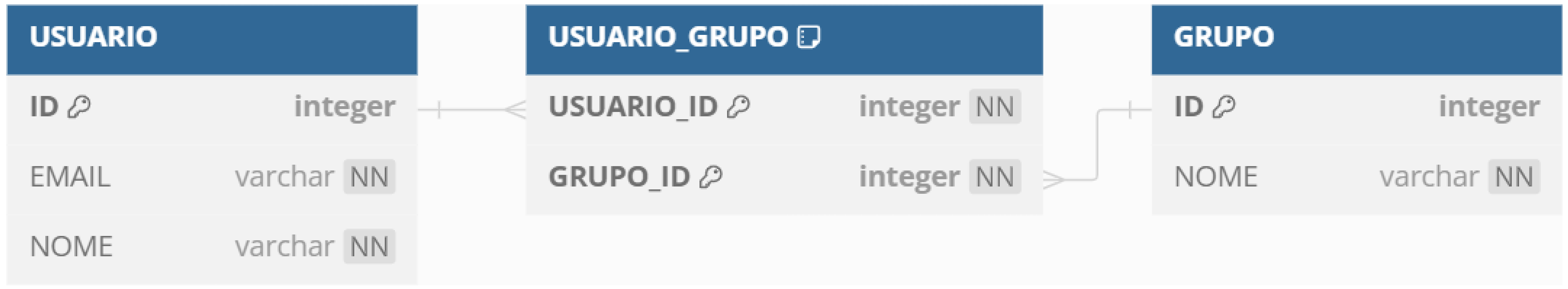
Proxy

Proxy



C Usuario
Integer id;
String email;
String nome;
Collection<Grupo> grupos;
*
*
C Grupo
Integer id;
String nome;

Proxy

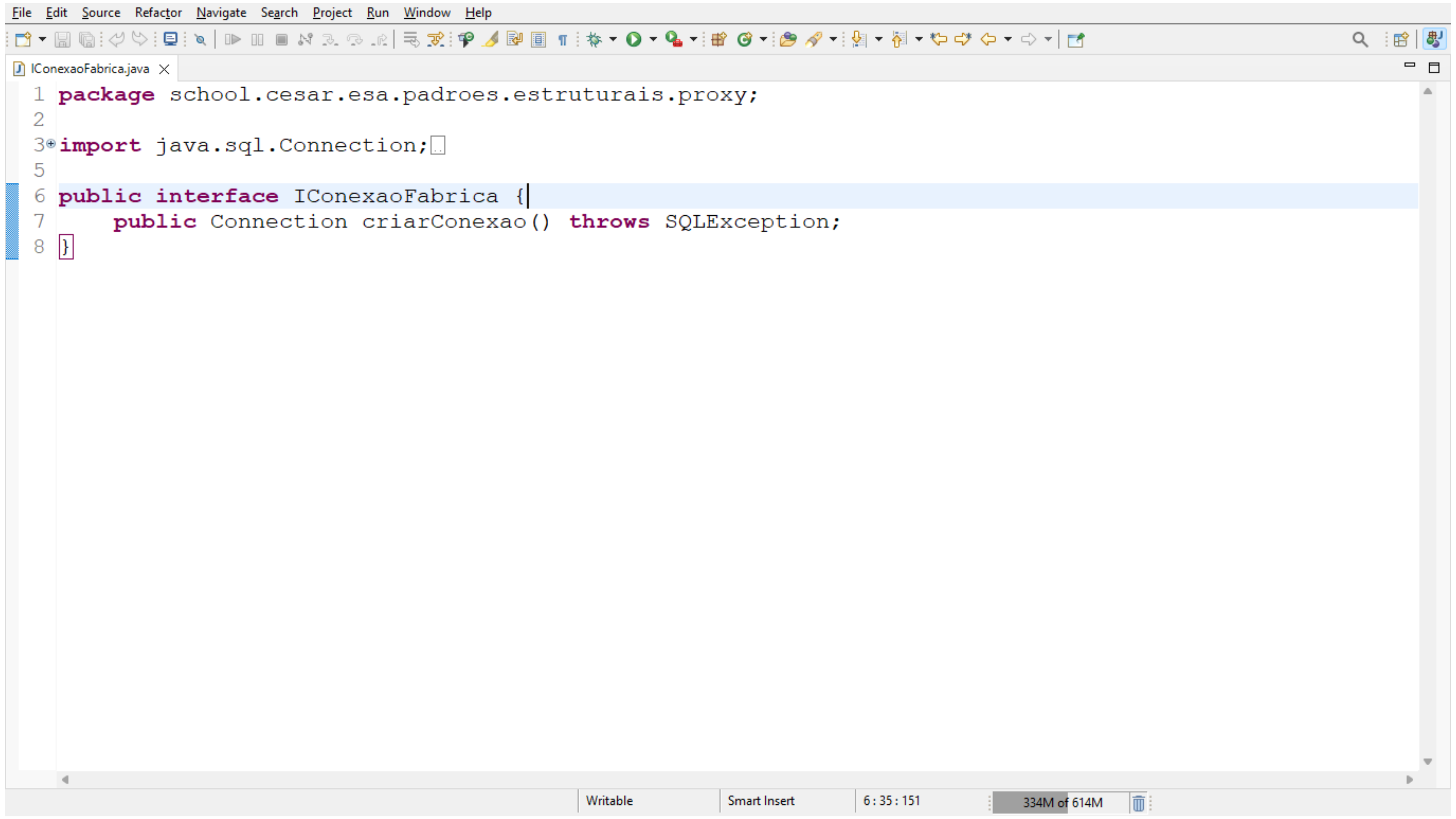


USUARIO		
ID	integer	
EMAIL	varchar	NN
NOME	varchar	NN

USUARIO_GRUPO		
USUARIO_ID	integer	NN
GRUPO_ID	integer	NN

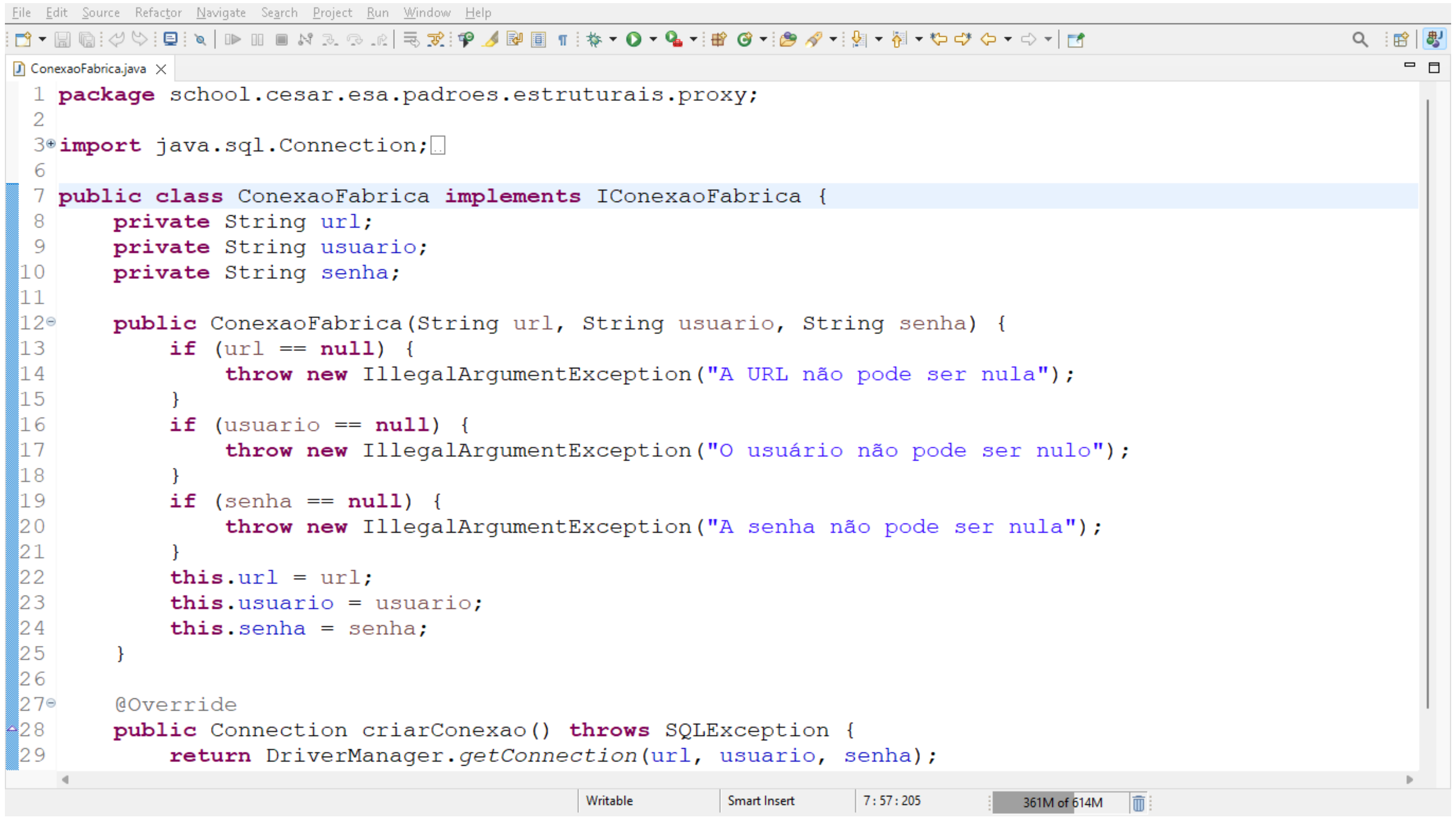
GRUPO		
ID	integer	
NOME	varchar	NN

Proxy


File Edit Source Refactor Navigate Search Project Run Window Help
IConexaoFabrica.java
Writable
Smart Insert
6 : 35 : 151
334M of 614M

```java
package school.cesar.esa.padroes.estruturais.proxy;

import java.sql.Connection;

public interface IConexaoFabrica {
    public Connection criarConexao() throws SQLException;
}
```

Proxy

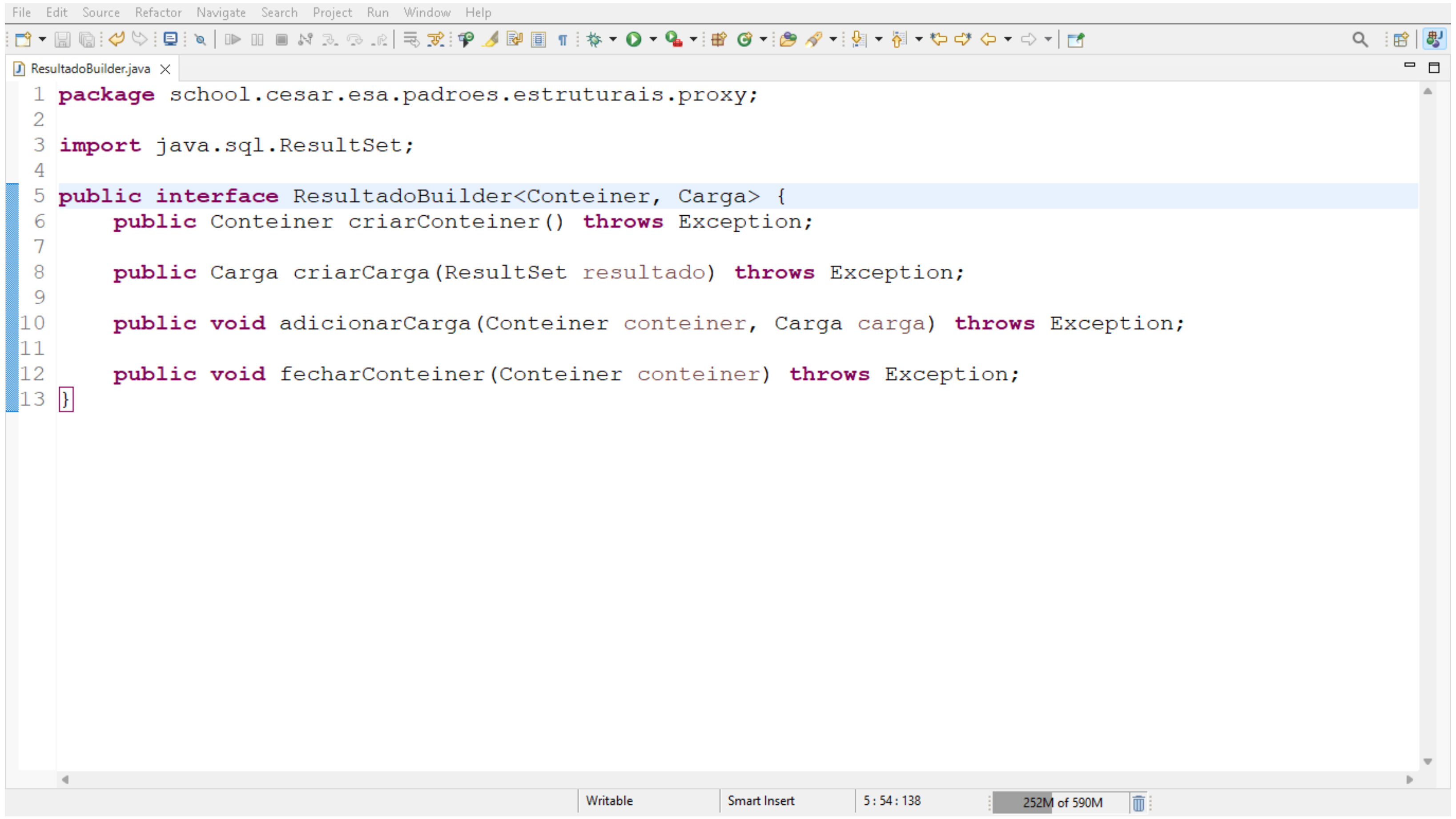


ConexaoFabrica.java

```
package school.cesar.esa.padroes.estruturais.proxy;

import java.sql.Connection;

public class ConexaoFabrica implements IConexaoFabrica {
    private String url;
    private String usuario;
    private String senha;

    public ConexaoFabrica(String url, String usuario, String senha) {
        if (url == null) {
            throw new IllegalArgumentException("A URL não pode ser nula");
        }
        if (usuario == null) {
            throw new IllegalArgumentException("O usuário não pode ser nulo");
        }
        if (senha == null) {
            throw new IllegalArgumentException("A senha não pode ser nula");
        }
        this.url = url;
        this.usuario = usuario;
        this.senha = senha;
    }

    @Override
    public Connection criarConexao() throws SQLException {
        return DriverManager.getConnection(url, usuario, senha);
```

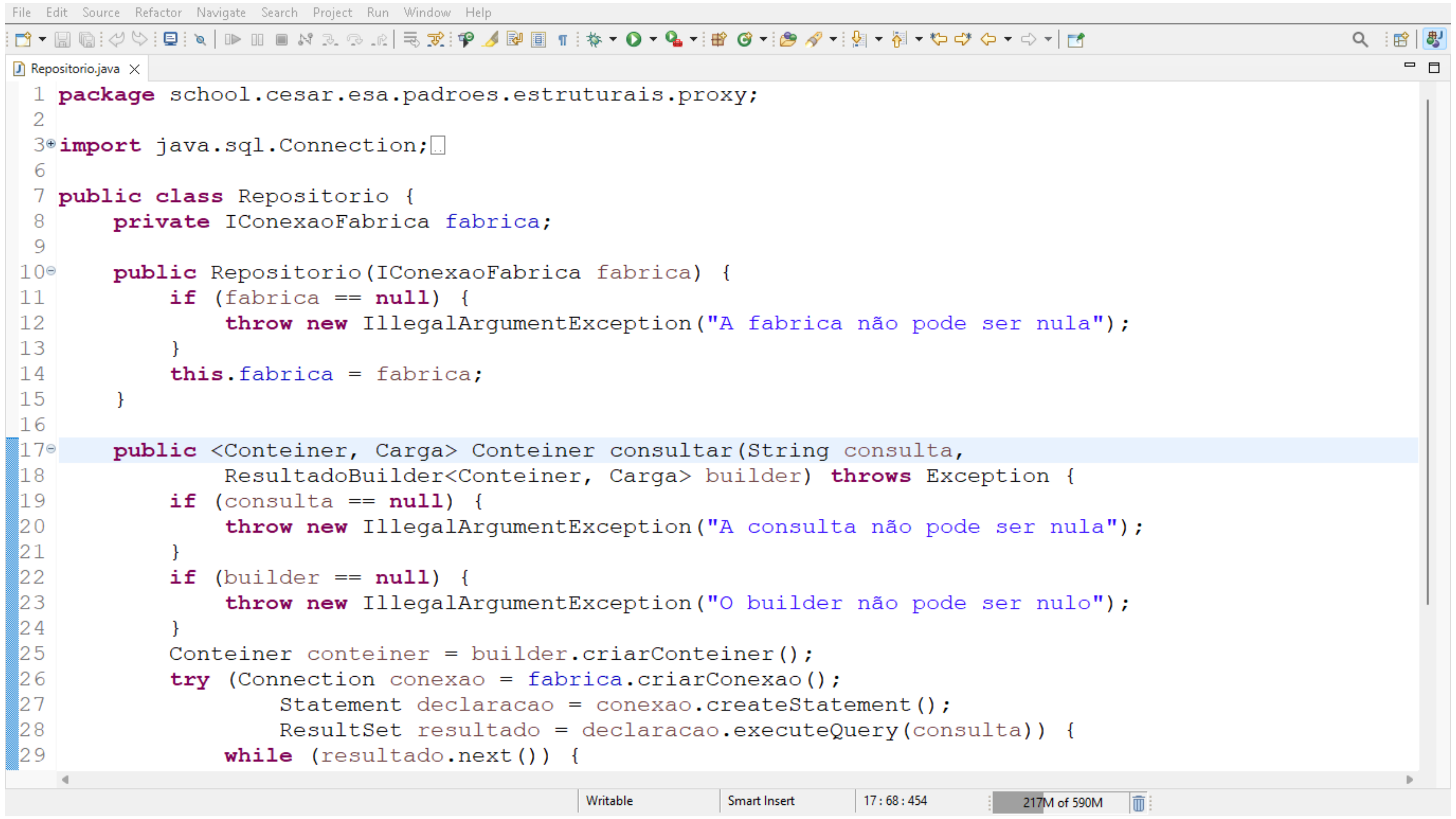
File Edit Source Refactor Navigate Search Project Run Window Help
Writable
Smart Insert
7 : 57 : 205
361M of 614M

Proxy


File Edit Source Refactor Navigate Search Project Run Window Help
ResultadoBuilder.java
Writable
Smart Insert
5 : 54 : 138
252M of 590M

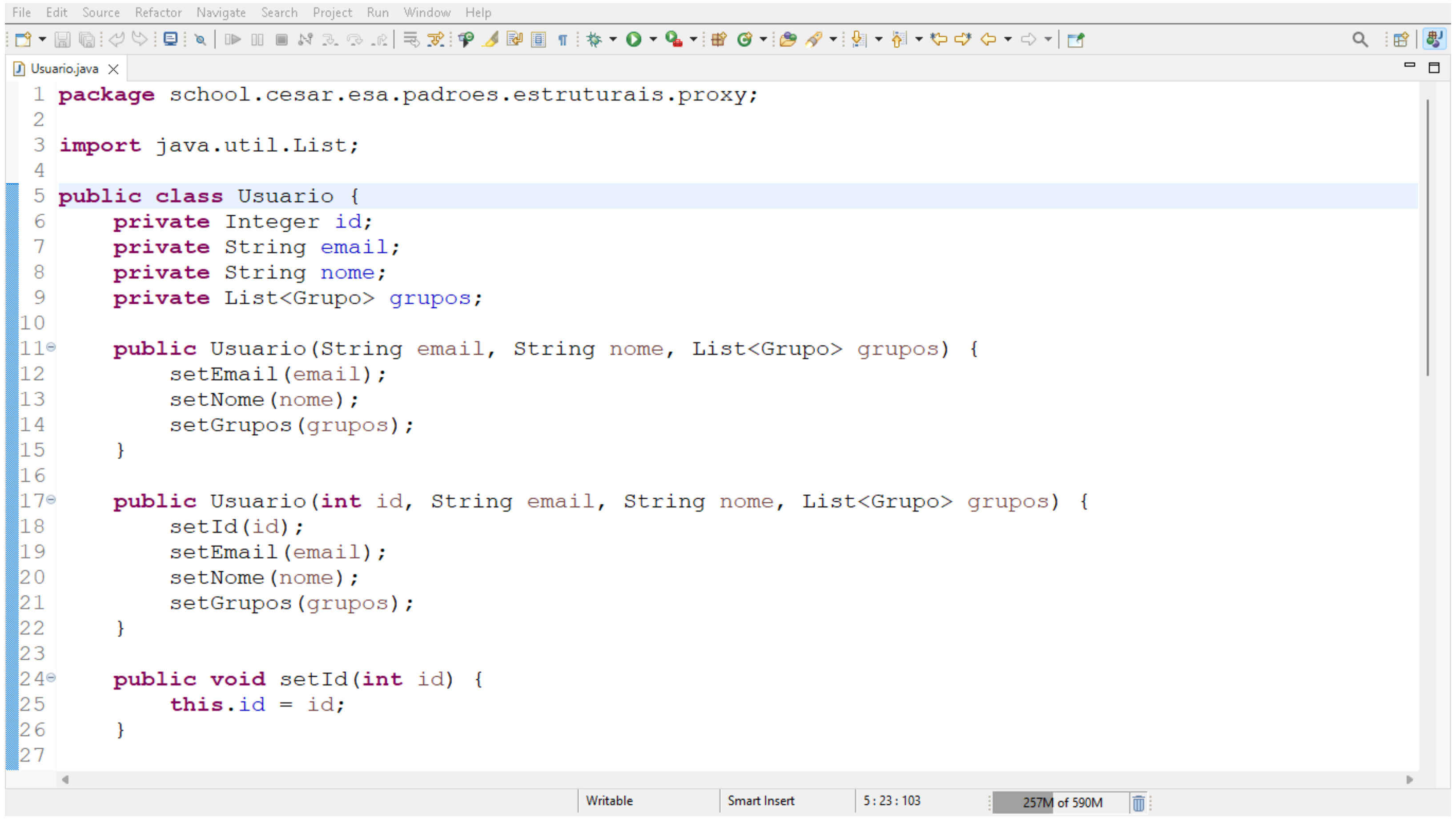
```java
package school.cesar.esa.padroes.estruturais.proxy;

import java.sql.ResultSet;

public interface ResultadoBuilder<Conteiner, Carga> {
    public Conteiner criarConteiner() throws Exception;

    public Carga criarCarga(ResultSet resultado) throws Exception;

    public void adicionarCarga(Conteiner conteiner, Carga carga) throws Exception;

    public void fecharConteiner(Conteiner conteiner) throws Exception;
}
```

Proxy



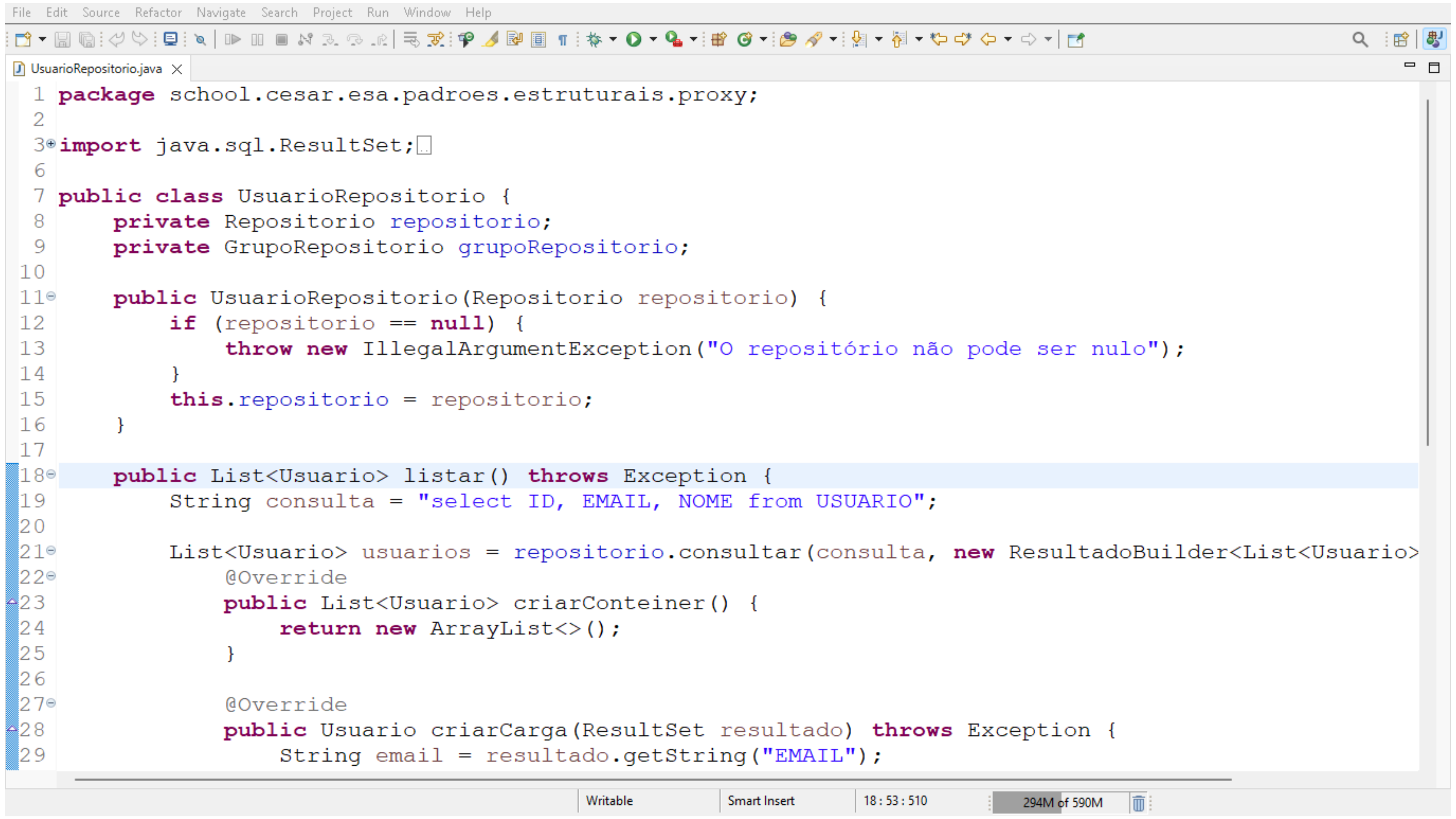
```java
package school.cesar.esa.padroes.estruturais.proxy;

import java.sql.Connection;

public class Repositorio {
    private IConexaoFabrica fabrica;

    public Repositorio(IConexaoFabrica fabrica) {
        if (fabrica == null) {
            throw new IllegalArgumentException("A fabrica não pode ser nula");
        }
        this.fabrica = fabrica;
    }

    public <Conteiner, Carga> Conteiner consultar(String consulta,
            ResultadoBuilder<Conteiner, Carga> builder) throws Exception {
        if (consulta == null) {
            throw new IllegalArgumentException("A consulta não pode ser nula");
        }
        if (builder == null) {
            throw new IllegalArgumentException("O builder não pode ser nulo");
        }
        Conteiner conteiner = builder.criarConteiner();
        try (Connection conexao = fabrica.criarConexao();
                Statement declaracao = conexao.createStatement();
                ResultSet resultado = declaracao.executeQuery(consulta)) {
            while (resultado.next()) {
```

Proxy


File Edit Source Refactor Navigate Search Project Run Window Help
Usuario.java
Writable
Smart Insert
5 : 23 : 103
257M of 590M

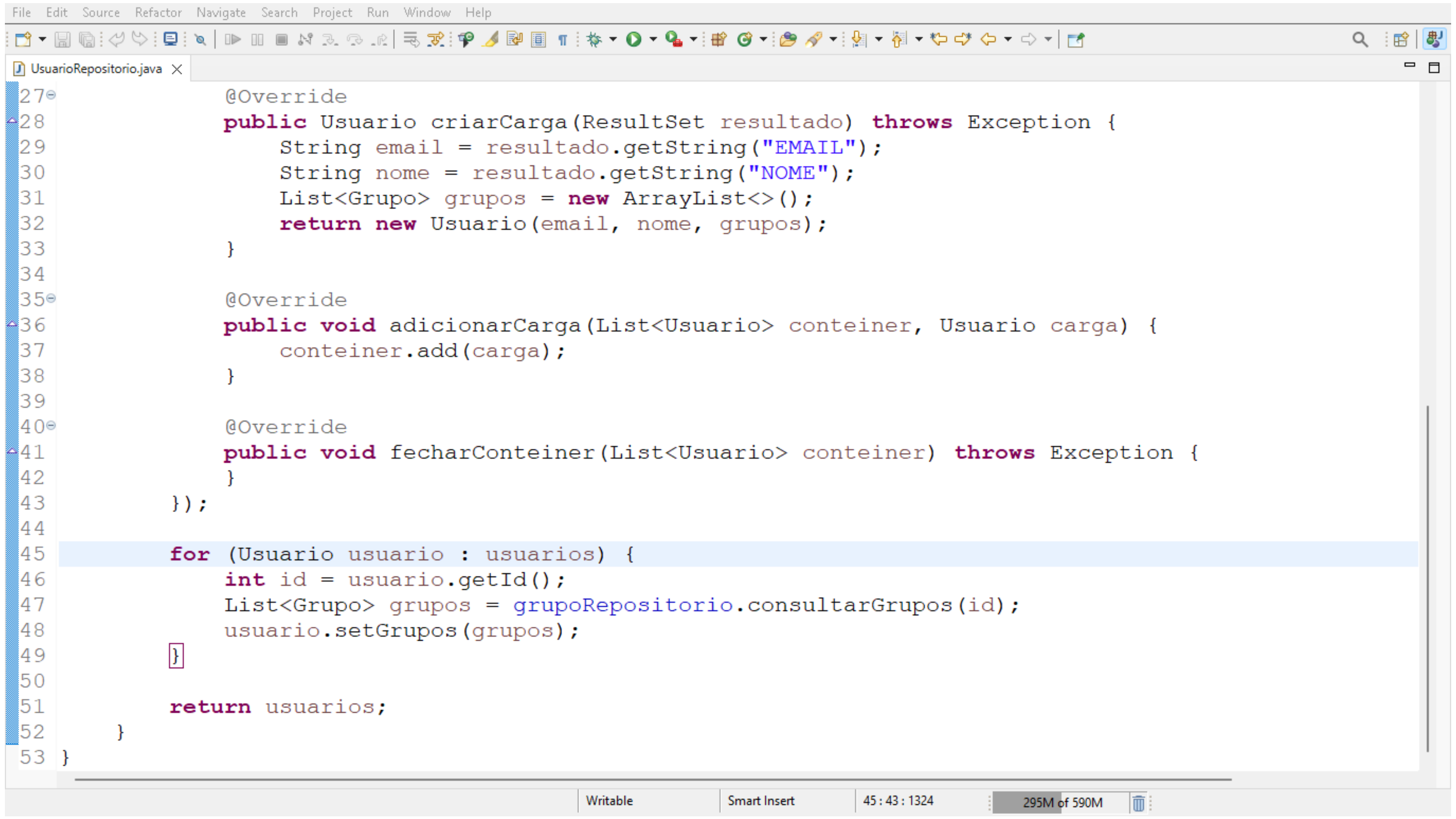
```java
package school.cesar.esa.padroes.estruturais.proxy;

import java.util.List;

public class Usuario {
    private Integer id;
    private String email;
    private String nome;
    private List<Grupo> grupos;

    public Usuario(String email, String nome, List<Grupo> grupos) {
        setEmail(email);
        setNome(nome);
        setGrupos(grupos);
    }

    public Usuario(int id, String email, String nome, List<Grupo> grupos) {
        setId(id);
        setEmail(email);
        setNome(nome);
        setGrupos(grupos);
    }

    public void setId(int id) {
        this.id = id;
    }

```

Proxy



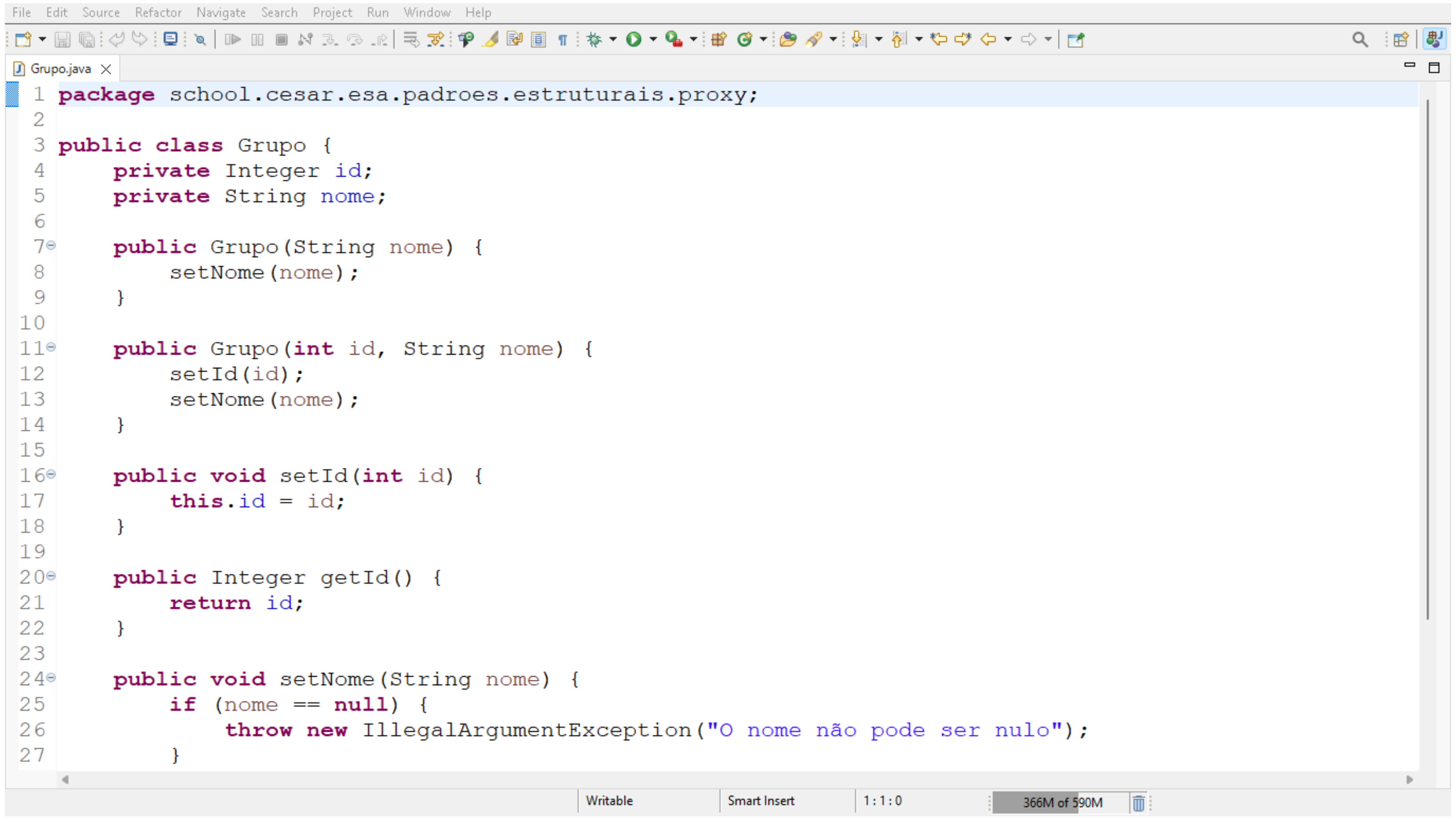
```java
package school.cesar.esa.padroes.estruturais.proxy;

import java.sql.ResultSet;

public class UsuarioRepositorio {
    private Repositorio repositorio;
    private GrupoRepositorio grupoRepositorio;

    public UsuarioRepositorio(Repositorio repositorio) {
        if (repositorio == null) {
            throw new IllegalArgumentException("O repositório não pode ser nulo");
        }
        this.repositorio = repositorio;
    }

    public List<Usuario> listar() throws Exception {
        String consulta = "select ID, EMAIL, NOME from USUARIO";

        List<Usuario> usuarios = repositorio.consultar(consulta, new ResultadoBuilder<List<Usuario>
            @Override
            public List<Usuario> criarConteiner() {
                return new ArrayList<>();
            }

            @Override
            public Usuario criarCarga(ResultSet resultado) throws Exception {
                String email = resultado.getString("EMAIL");
```

Proxy



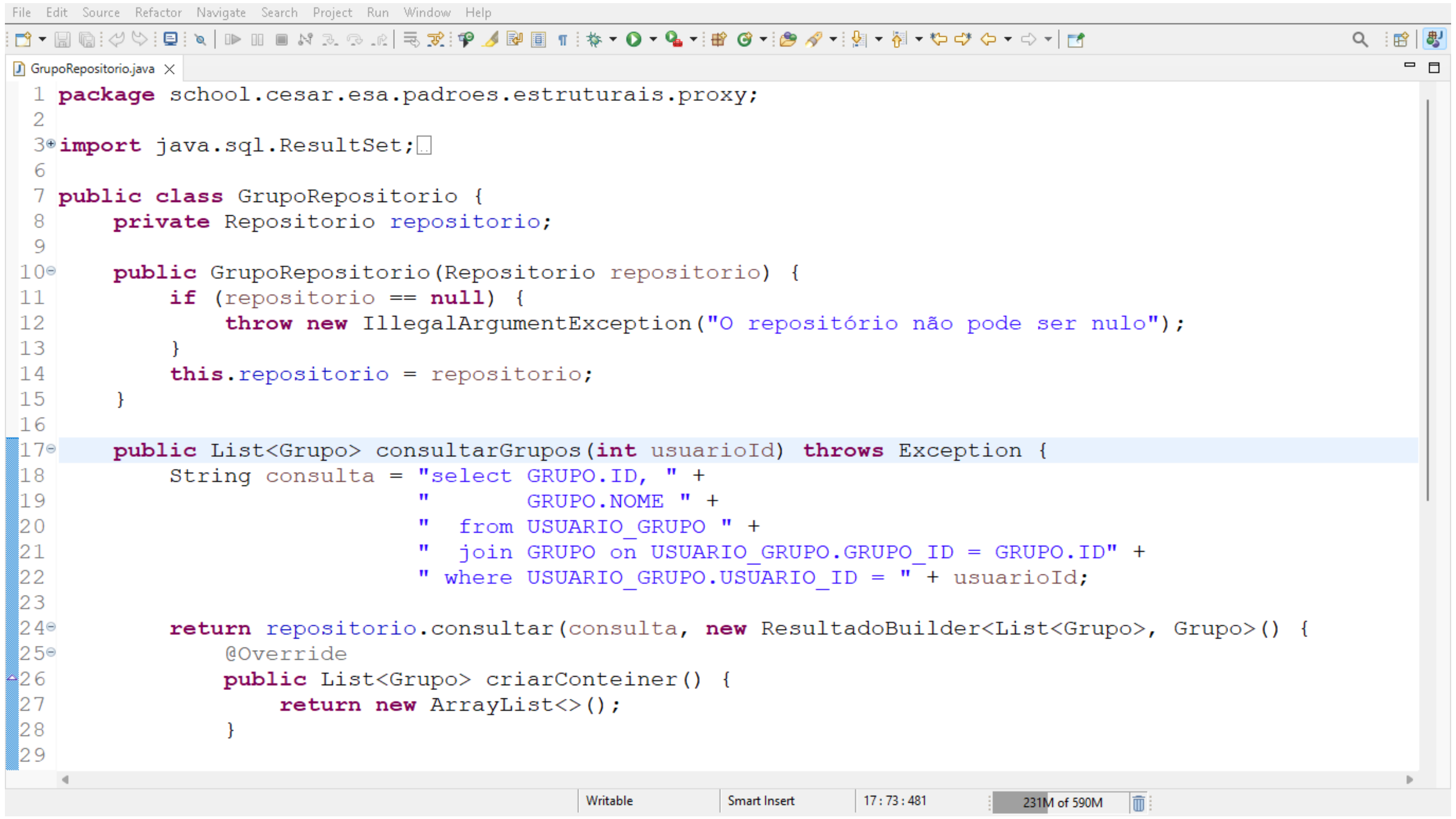
```java
            @Override
            public Usuario criarCarga(ResultSet resultado) throws Exception {
                String email = resultado.getString("EMAIL");
                String nome = resultado.getString("NOME");
                List<Grupo> grupos = new ArrayList<>();
                return new Usuario(email, nome, grupos);
            }

            @Override
            public void adicionarCarga(List<Usuario> conteiner, Usuario carga) {
                conteiner.add(carga);
            }

            @Override
            public void fecharConteiner(List<Usuario> conteiner) throws Exception {
            }
        });

        for (Usuario usuario : usuarios) {
            int id = usuario.getId();
            List<Grupo> grupos = grupoRepositorio.consultarGrupos(id);
            usuario.setGrupos(grupos);
        }

        return usuarios;
    }
}
```

Proxy


File Edit Source Refactor Navigate Search Project Run Window Help
Grupo.java
Writable
Smart Insert
1 : 1 : 0
366M of 590M

```java
package school.cesar.esa.padroes.estruturais.proxy;

public class Grupo {
    private Integer id;
    private String nome;

    public Grupo(String nome) {
        setNome(nome);
    }

    public Grupo(int id, String nome) {
        setId(id);
        setNome(nome);
    }

    public void setId(int id) {
        this.id = id;
    }

    public Integer getId() {
        return id;
    }

    public void setNome(String nome) {
        if (nome == null) {
            throw new IllegalArgumentException("O nome não pode ser nulo");
        }
```

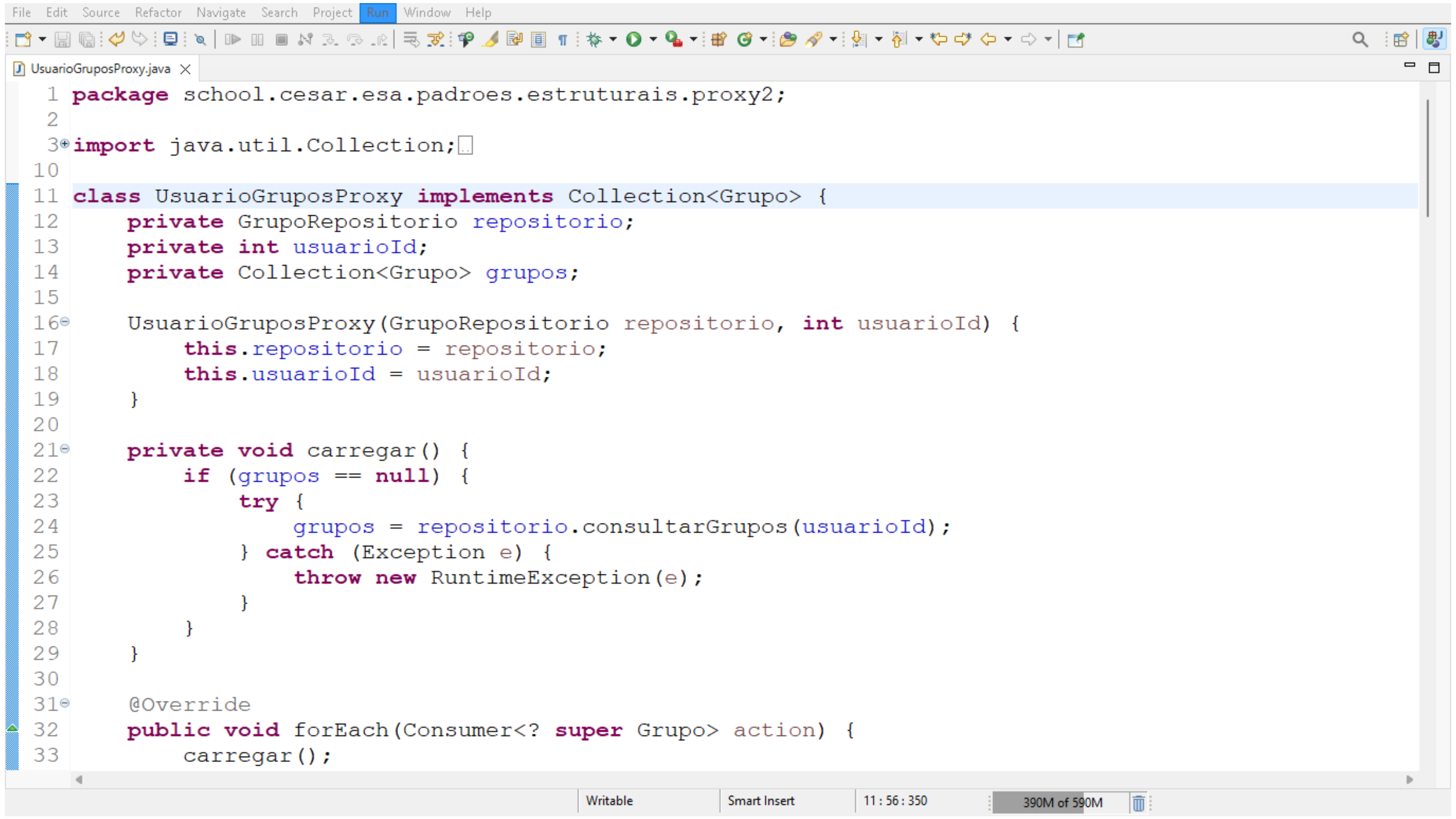
Proxy



```java
package school.cesar.esa.padroes.estruturais.proxy;

import java.sql.ResultSet;

public class GrupoRepositorio {
    private Repositorio repositorio;

    public GrupoRepositorio(Repositorio repositorio) {
        if (repositorio == null) {
            throw new IllegalArgumentException("O repositório não pode ser nulo");
        }
        this.repositorio = repositorio;
    }

    public List<Grupo> consultarGrupos(int usuarioId) throws Exception {
        String consulta = "select GRUPO.ID, " +
                          "       GRUPO.NOME " +
                          "   from USUARIO_GRUPO " +
                          "   join GRUPO on USUARIO_GRUPO.GRUPO_ID = GRUPO.ID" +
                          " where USUARIO_GRUPO.USUARIO_ID = " + usuarioId;

        return repositorio.consultar(consulta, new ResultadoBuilder<List<Grupo>, Grupo>() {
            @Override
            public List<Grupo> criarConteiner() {
                return new ArrayList<>();
            }

```

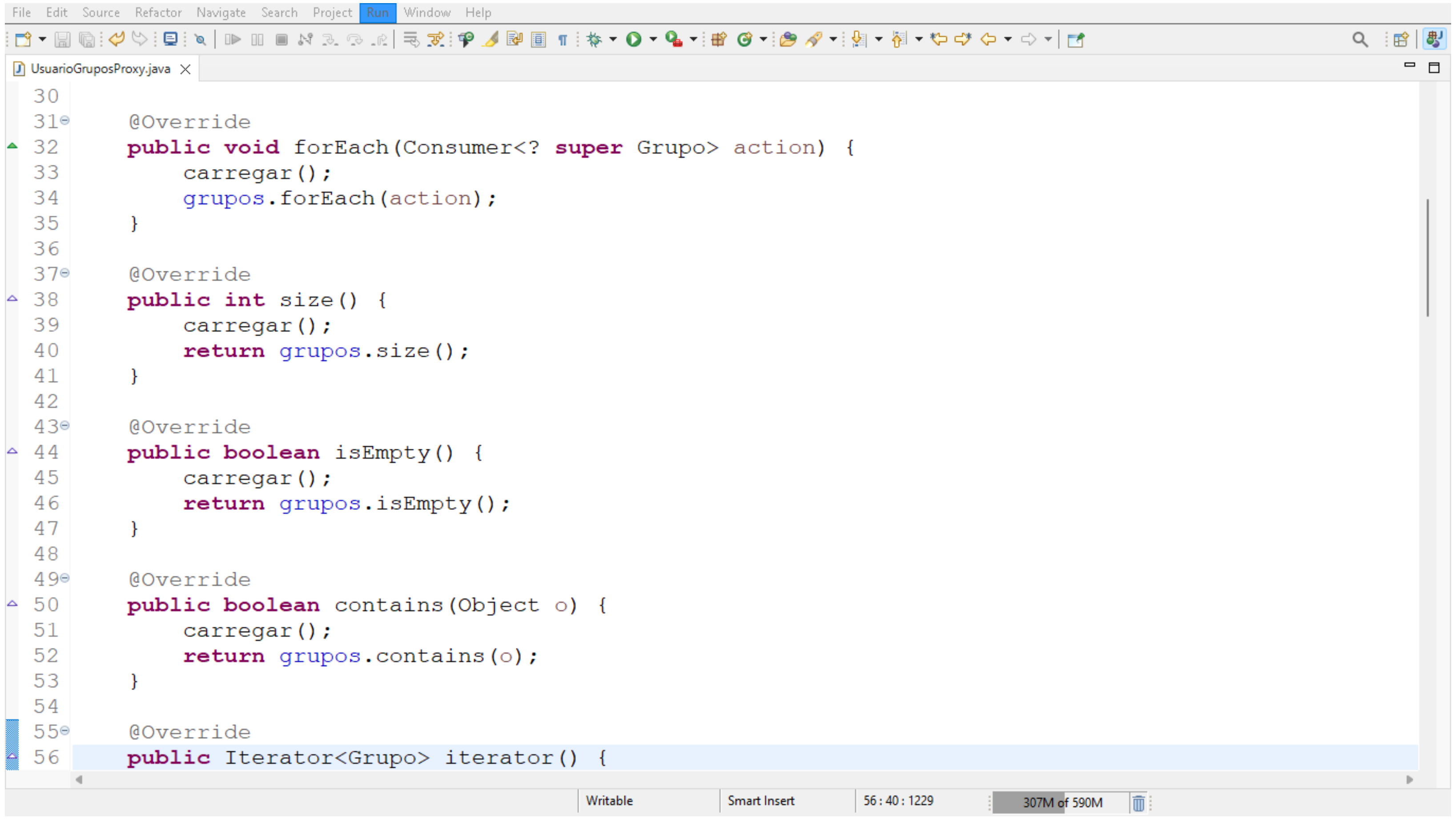
Proxy

Proxy



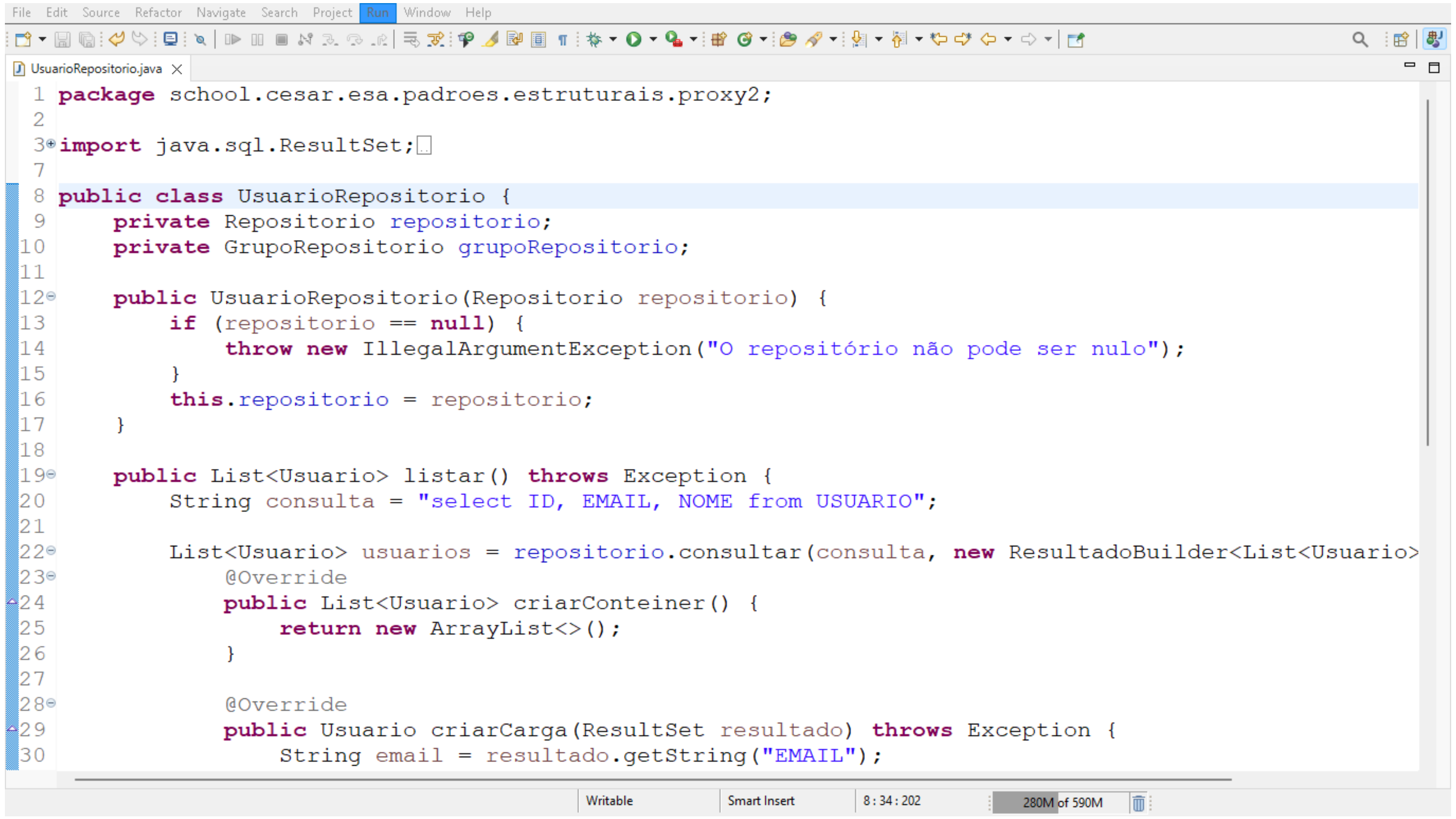
```java
package school.cesar.esa.padroes.estruturais.proxy2;

import java.util.Collection;

class UsuarioGruposProxy implements Collection<Grupo> {
    private GrupoRepositorio repositorio;
    private int usuarioId;
    private Collection<Grupo> grupos;

    UsuarioGruposProxy(GrupoRepositorio repositorio, int usuarioId) {
        this.repositorio = repositorio;
        this.usuarioId = usuarioId;
    }

    private void carregar() {
        if (grupos == null) {
            try {
                grupos = repositorio.consultarGrupos(usuarioId);
            } catch (Exception e) {
                throw new RuntimeException(e);
            }
        }
    }

    @Override
    public void forEach(Consumer<? super Grupo> action) {
        carregar();
```

Proxy



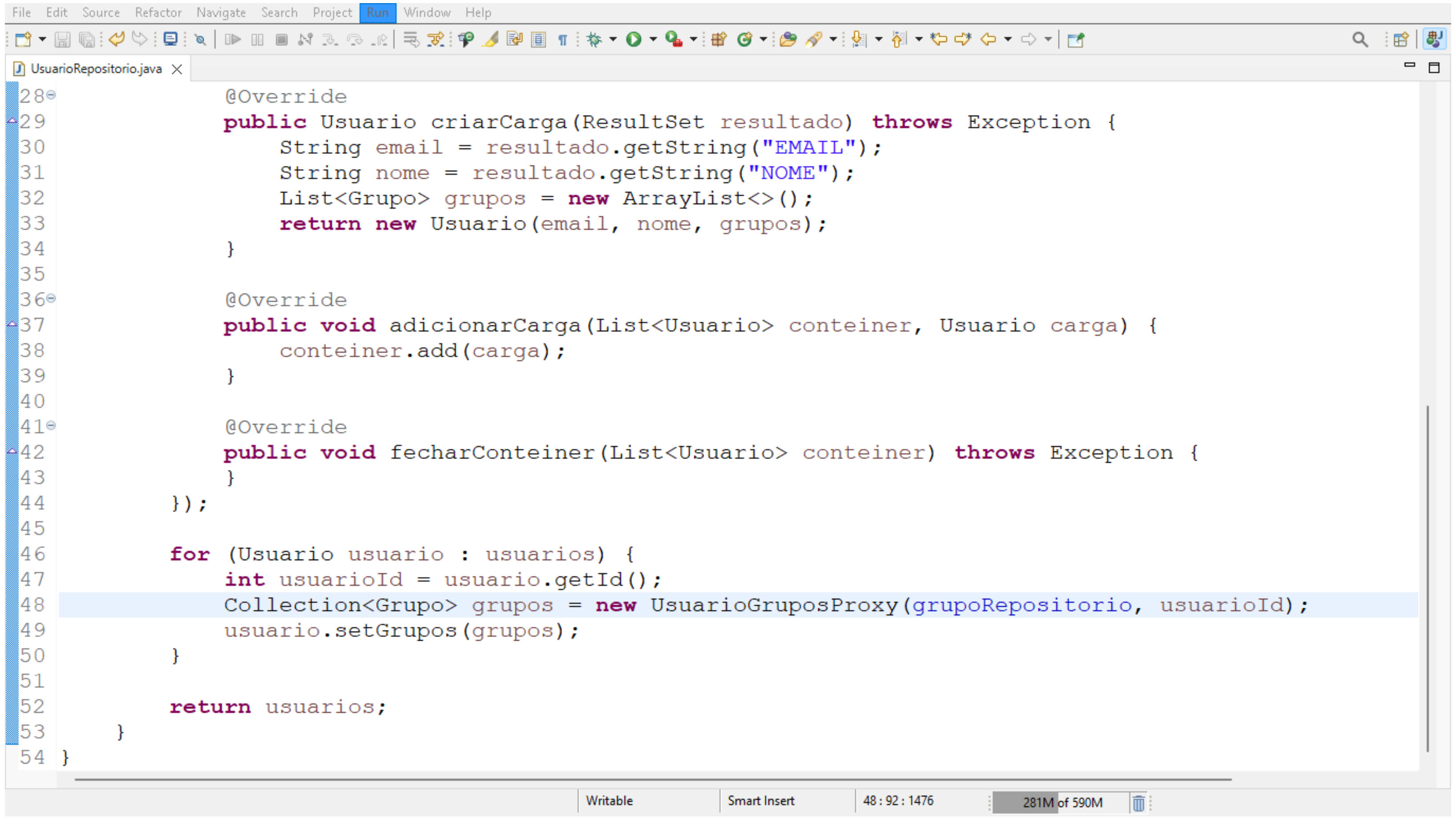
```java

    @Override
    public void forEach(Consumer<? super Grupo> action) {
        carregar();
        grupos.forEach(action);
    }

    @Override
    public int size() {
        carregar();
        return grupos.size();
    }

    @Override
    public boolean isEmpty() {
        carregar();
        return grupos.isEmpty();
    }

    @Override
    public boolean contains(Object o) {
        carregar();
        return grupos.contains(o);
    }

    @Override
    public Iterator<Grupo> iterator() {
```

Proxy



```java
package school.cesar.esa.padroes.estruturais.proxy2;

import java.sql.ResultSet;

public class UsuarioRepositorio {
    private Repositorio repositorio;
    private GrupoRepositorio grupoRepositorio;

    public UsuarioRepositorio(Repositorio repositorio) {
        if (repositorio == null) {
            throw new IllegalArgumentException("O repositório não pode ser nulo");
        }
        this.repositorio = repositorio;
    }

    public List<Usuario> listar() throws Exception {
        String consulta = "select ID, EMAIL, NOME from USUARIO";

        List<Usuario> usuarios = repositorio.consultar(consulta, new ResultadoBuilder<List<Usuario>
            @Override
            public List<Usuario> criarConteiner() {
                return new ArrayList<>();
            }

            @Override
            public Usuario criarCarga(ResultSet resultado) throws Exception {
                String email = resultado.getString("EMAIL");
```

Proxy



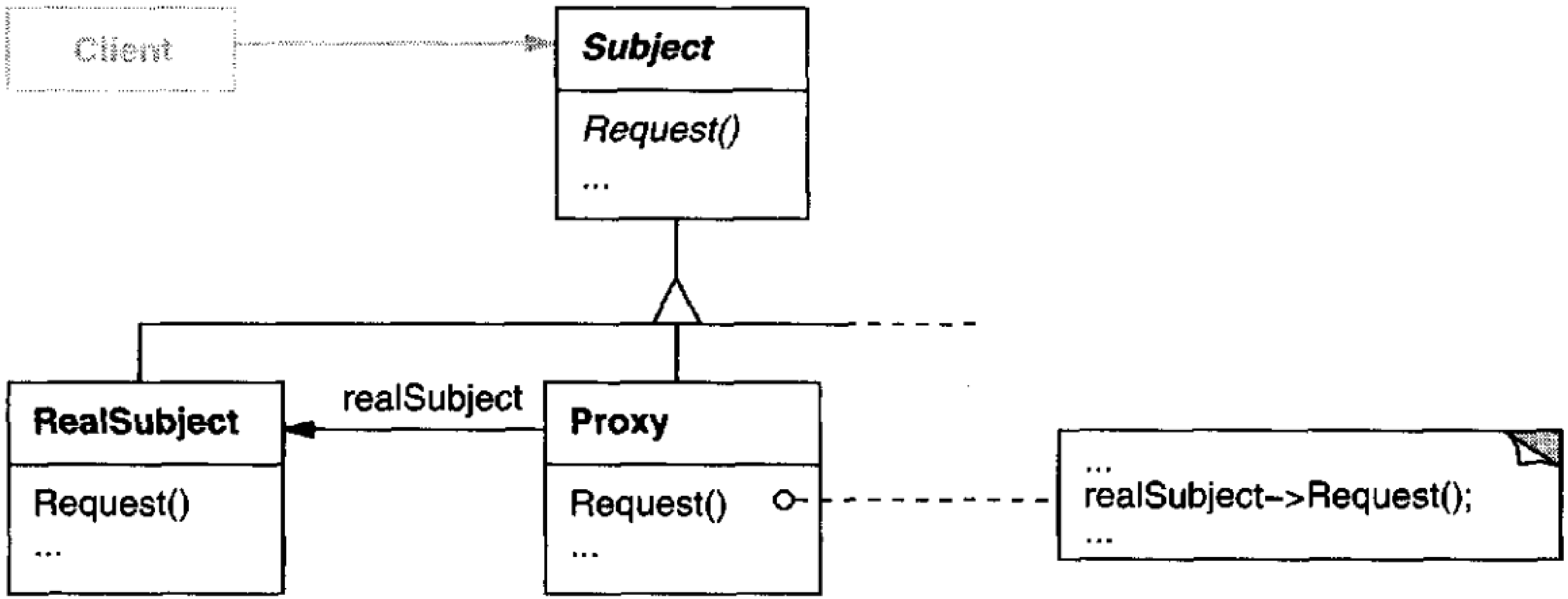
File Edit Source Refactor Navigate Search Project Run Window Help
UsuarioRepositorio.java

```java
            @Override
            public Usuario criarCarga(ResultSet resultado) throws Exception {
                String email = resultado.getString("EMAIL");
                String nome = resultado.getString("NOME");
                List<Grupo> grupos = new ArrayList<>();
                return new Usuario(email, nome, grupos);
            }

            @Override
            public void adicionarCarga(List<Usuario> conteiner, Usuario carga) {
                conteiner.add(carga);
            }

            @Override
            public void fecharConteiner(List<Usuario> conteiner) throws Exception {
            }
        });

        for (Usuario usuario : usuarios) {
            int usuarioId = usuario.getId();
            Collection<Grupo> grupos = new UsuarioGruposProxy(grupoRepositorio, usuarioId);
            usuario.setGrupos(grupos);
        }

        return usuarios;
    }
}
```

Proxy

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

Proxy



Client
Subject
Request()
...
RealSubject
Request()
...
realSubject
Proxy
Request()
...
...
realSubject->Request();
...

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