Tomas Camilo Ostos Medina-202123214

Taller 5

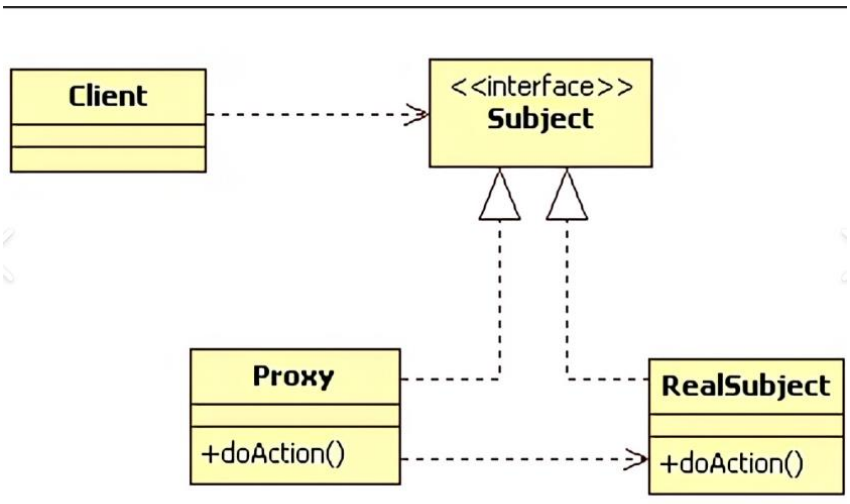
Patrones

En este documento se analizará el patrón proxy descrito en *"Design Patterns Elements of Reusable Object-Oriented Software. Erich Gamma, Richard Helm, Ralph Johnson, John M. Vlissides"*

Se buscará entender y documentar el código presentado para la justificación de la elección del patrón de diseño implementado. También se hará uso de UMLs para representar gráficamente los conceptos presentes en este.

Patrón de diseño proxy

El patrón de diseño proxy pertenece al tipo de patrones estructurales. La pertenecía a esta categoría de patrones se debe a que con este se crean estructuras complejas a partir de otras más simples. simplifica un objeto complejo para representarlo de una manera mas simple. Este permite posponer la creación de un objeto hasta que se necesariamente necesaria su implementación. Comúnmente el proxy posee los mismos métodos que al objeto que representa, pero se llaman solo cuando el objeto ha sido cargado completamente.



- La interfaz **Subject**, un ojeto **RealSubject** que es el objeto al que accede a través del **Proxy**. El objeto **Proxy** mantiene la referencia al objeto **RealSubject** y controla el acceso a sus métodos, introduciendo las capacidades adicionales que fuesen necesarias. Tanto la clase **RealSubject** como la clase Proxy implementan la interfaz **Subject**.

Ejemplo

Aplicación:

• Un objeto, como una imagen grande, puede tardar mucho en cargarse.

• Un objeto se encuentra en una máquina remota solamente accesible por red.

• El objeto tiene el acceso restringido, el Proxy puede encargarse de validar los permisos.

Supongamos una interfaz como el de cualquier navegador de internet y que tenemos un panel en el cual queremos mostrar una imagen que es muy grande. Como sabemos que va a tardar bastante en cargarse utilizaremos un Proxy.

En este caso se abre un nuevo hilo en el que utilizando el MediaTracker intenta cargar la imagen. Mientras en el método paint() comprueba si se ha cargado la imagen, en caso afirmativo la muestra y si todavía no hemos podido cargarla muestra un rectángulo vacío.

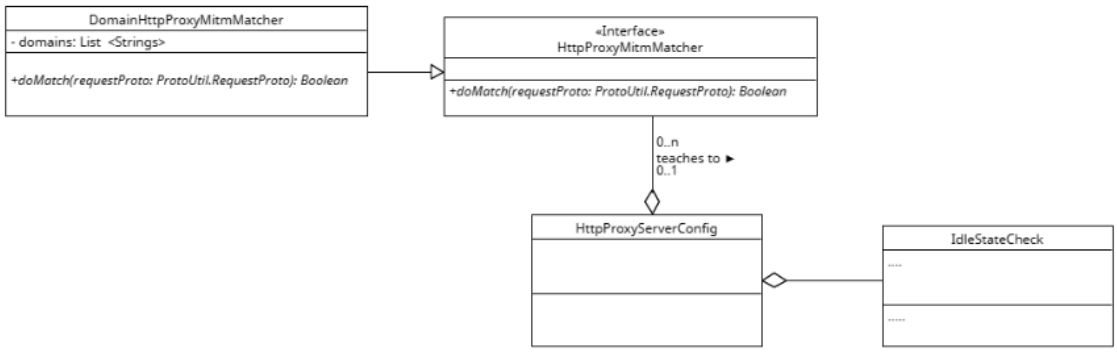
Repositorio de análisis

https://github.com/monkeyWie/proxyee.git

Información sobre el proyecto

Proxyee, biblioteca de java creada por el usuario de github monkeyWie se utiliza para crear servidores proxy HTTP y poderlos utilizar en poderlos utilizar en distintos proyectos. Esto significa que ayuda a dirigir el tráfico de internet, como el acceso a sitios web, a través de un servidor intermedio. La biblioteca es capaz de manejar los protocolos HTTP, HTTPS y Websocket. Además, permite realizar ataques MITM, lo que significa que puede interceptar y modificar paquetes de datos que se envían a través de la red. Para usar esta biblioteca, se debe incluir como una dependencia en el proyecto de desarrollo.

La clase "HttpProxyServerConfig" es una de las clases principales en las que se implementa el patrón de diseño Proxy y está diseñada para configurar un servidor proxy HTTP. Esta clase contiene propiedades para configurar aspectos como el puerto del servidor, las reglas de proxy, la autenticación, entre otros. Además, la librería tiene otras clases e interfaces relacionadas, como "ProxyConfig", "ProxyHandleFactory", "HttpProxyServer", "ProxyInterceptInitializer", y "ProxyInterceptPipeline". Estas clases e interfaces están interconectadas para proporcionar funcionalidades como la creación de servidores proxy, la manipulación de peticiones y respuestas, y la configuración de reglas de interceptación.



En este contexto, la clase “HttpProxyServerConfig” funciona como un proxy que regula el acceso al objeto real, que es el servidor HTTP en este caso. Hace uso de otras clases como “ProxyConfig”,

“ProxyHandleFactory”, “HttpProxyServer”, entre otras, para ofrecer una capa de abstracción que permite interceptar y modificar paquetes HTTP, HTTPS y Websocket de manera uniforme. Este método facilita la personalización y ampliación de las funcionalidades del servidor proxy según sea necesario.

El objetivo de implementar el patrón Proxy en esta biblioteca es ocultar la complejidad de los servidores HTTP reales y gestionar el acceso a ellos. Algunos de los beneficios de este patrón incluyen la posibilidad de crear una interfaz para otros recursos, ocultar la complejidad de los servidores HTTP, personalizar y expandir las funcionalidades del servidor proxy, y proporcionar un medio para interceptar y alterar paquetes HTTP, HTTPS y Websocket. Además, el patrón Proxy permite un acceso seguro a los objetos, simplifica objetos complejos y representa objetos remotos de forma local, lo que aporta flexibilidad y seguridad a la biblioteca.

El uso del patrón Proxy en este caso puede introducir capas adicionales de abstracción, lo que podría incrementar la complejidad del código. Asimismo, como con cualquier patrón de diseño, una implementación incorrecta o excesiva podría resultar en un uso excesivo de memoria o un impacto negativo en el rendimiento. No obstante, estas desventajas pueden ser mitigadas con una implementación cuidadosa y considerada del patrón.

Una alternativa a la implementación de esta biblioteca sería gestionar directamente las solicitudes Http que recibe el sistema en la aplicación principal sin necesidad de verificación. En lugar de utilizar un proxy para controlar el acceso a los servidores reales, se podría escribir código personalizado para gestionar las solicitudes, enviarlas a los servidores reales y procesar las respuestas directamente en la aplicación principal. Sin embargo, esto podría incrementar la complejidad de la aplicación y dificultar su mantenimiento y extensión en el futuro, por lo que no resulta ser un intercambio más rentable que el que se tiene usando el patrón Proxy.

Código con el patrón implementado

Clase domainHttpProxyMitmMatcher

```java
package com.github.monkeywie.proxyee.server.accept;

import com.github.monkeywie.proxyee.util.ProtoUtil;

import java.util.List;

/**
 * @Author LiWei
 * @Description 通过域名配置是否走中间人攻击
 * @Date 2023/06/07 11:11
 */
public class DomainHttpProxyMitmMatcher implements HttpProxyMitmMatcher{

    private List<String> domains;

    public DomainHttpProxyMitmMatcher(List<String> domains) {
        this.domains = domains;
    }

    @Override
    public boolean doMatch(ProtoUtil.RequestProto requestProto) {
        if(domains == null || domains.isEmpty()){
            return false;
        }
        return domains.stream().anyMatch(host -> requestProto.getHost().equals(host));
    }
}
```

Interfaz HttpProxyMitmMatcher

```java
package com.github.monkeywie.proxyee.server.accept;

import com.github.monkeywie.proxyee.util.ProtoUtil;

/**
 * @Author LiWei
 * @Description 用于匹配请求是否需要走中间人攻击
 * @Date 2023/06/07 11:11
 */
public interface HttpProxyMitmMatcher {
    /**
     * 客户端有新的连接建立时触发
     *
     * @param requestProto
     * @return 返回true表示走中间人攻击，返回false则直接转发
     */
    boolean doMatch(ProtoUtil.RequestProto requestProto);
}
```

Tomas Camilo Ostos Medina-202123214

Clase HttpProxyServerConfig

```
18  public class HttpProxyServerConfig {
19      private SslContext clientSslCtx;
20      private String issuer;
21      private Date caNotBefore;
22      private Date caNotAfter;
23      private PrivateKey caPriKey;
24      private PrivateKey serverPriKey;
25      private PublicKey serverPubKey;
26      private EventLoopGroup proxyLoopGroup;
27      private int bossGroupThreads;
28      private int workerGroupThreads;
29      private int proxyGroupThreads;
30      private boolean handleSsl;
31      private HttpProxyAcceptHandler httpProxyAcceptHandler;
32      private HttpProxyAuthenticationProvider authenticationProvider;
33      private HttpProxyMitmMatcher mitmMatcher;
34      private final AddressResolverGroup<? extends SocketAddress> resolver;
35      private Iterable<String> ciphers;
36      private int maxInitialLineLength = HttpObjectDecoder.DEFAULT_MAX_INITIAL_LINE_LENGTH;
37      private int maxHeaderSize = HttpObjectDecoder.DEFAULT_MAX_HEADER_SIZE;
38      private int maxChunkSize = HttpObjectDecoder.DEFAULT_MAX_CHUNK_SIZE;
39      private IdleStateCheck idleStateCheck;
40
41      public HttpProxyServerConfig() {
42          this(DefaultAddressResolverGroup.INSTANCE);
43      }
44
45      public HttpProxyServerConfig(final AddressResolverGroup<? extends SocketAddress> resolver) {
46          this.resolver = resolver;
47      }
48
```

```
49      private HttpProxyServerConfig(Builder builder) {
50          this.clientSslCtx = builder.clientSslCtx;
51          this.issuer = builder.issuer;
52          this.caNotBefore = builder.caNotBefore;
53          this.caNotAfter = builder.caNotAfter;
54          this.caPriKey = builder.caPriKey;
55          this.serverPriKey = builder.serverPriKey;
56          this.serverPubKey = builder.serverPubKey;
57          this.proxyLoopGroup = builder.proxyLoopGroup;
58          this.bossGroupThreads = builder.bossGroupThreads;
59          this.workerGroupThreads = builder.workerGroupThreads;
60          this.proxyGroupThreads = builder.proxyGroupThreads;
61          this.handleSsl = builder.handleSsl;
62          this.httpProxyAcceptHandler = builder.httpProxyAcceptHandler;
63          this.resolver = builder.resolver;
64          this.maxInitialLineLength = builder.maxInitialLineLength;
65          this.maxHeaderSize = builder.maxHeaderSize;
66          this.maxChunkSize = builder.maxChunkSize;
67          this.idleStateCheck = builder.idleStateCheck;
68      }
69
70      public SslContext getClientSslCtx() {
71          return clientSslCtx;
72      }
73
74      public void setClientSslCtx(SslContext clientSslCtx) {
75          this.clientSslCtx = clientSslCtx;
76      }
77
78      public String getIssuer() {
79          return issuer;
80      }
81
82      public void setIssuer(String issuer) {
83          this.issuer = issuer;
```

```
84          }
85
86          public Date getCaNotBefore() {
87              return caNotBefore;
88          }
89
90          public void setCaNotBefore(Date caNotBefore) {
91              this.caNotBefore = caNotBefore;
92          }
93
94          public Date getCaNotAfter() {
95              return caNotAfter;
96          }
97
98          public void setCaNotAfter(Date caNotAfter) {
99              this.caNotAfter = caNotAfter;
100         }
101
102         public PrivateKey getCaPriKey() {
103             return caPriKey;
104         }
105
106         public void setCaPriKey(PrivateKey caPriKey) {
107             this.caPriKey = caPriKey;
108         }
109
110         public PrivateKey getServerPriKey() {
111             return serverPriKey;
112         }
113
114         public void setServerPriKey(PrivateKey serverPriKey) {
115             this.serverPriKey = serverPriKey;
116         }
117
118         public PublicKey getServerPubKey() {
```

```
119             return serverPubKey;
120         }
121
122         public void setServerPubKey(PublicKey serverPubKey) {
123             this.serverPubKey = serverPubKey;
124         }
125
126         public EventLoopGroup getProxyLoopGroup() {
127             return proxyLoopGroup;
128         }
129
130         public void setProxyLoopGroup(EventLoopGroup proxyLoopGroup) {
131             this.proxyLoopGroup = proxyLoopGroup;
132         }
133
134         public boolean isHandleSsl() {
135             return handleSsl;
136         }
137
138         public void setHandleSsl(boolean handleSsl) {
139             this.handleSsl = handleSsl;
140         }
141
142         public int getBossGroupThreads() {
143             return bossGroupThreads;
144         }
145
146         public void setBossGroupThreads(int bossGroupThreads) {
147             this.bossGroupThreads = bossGroupThreads;
148         }
149
150         public int getWorkerGroupThreads() {
151             return workerGroupThreads;
152         }
153
```

```
154         public void setWorkerGroupThreads(int workerGroupThreads) {
155             this.workerGroupThreads = workerGroupThreads;
156         }
157 
158         public int getProxyGroupThreads() {
159             return proxyGroupThreads;
160         }
161 
162         public void setProxyGroupThreads(int proxyGroupThreads) {
163             this.proxyGroupThreads = proxyGroupThreads;
164         }
165 
166         public HttpProxyAcceptHandler getHttpProxyAcceptHandler() {
167             return httpProxyAcceptHandler;
168         }
169 
170         public void setHttpProxyAcceptHandler(final HttpProxyAcceptHandler httpProxyAcceptHandler) {
171             this.httpProxyAcceptHandler = httpProxyAcceptHandler;
172         }
173 
174         public HttpProxyAuthenticationProvider getAuthenticationProvider() {
175             return authenticationProvider;
176         }
177 
178         public void setAuthenticationProvider(final HttpProxyAuthenticationProvider authenticationProvider) {
179             this.authenticationProvider = authenticationProvider;
180         }
181 
182         public HttpProxyMitmMatcher getMitmMatcher() {
183             return mitmMatcher;
184         }
185 
186         public void setMitmMatcher(HttpProxyMitmMatcher mitmMatcher) {
187             this.mitmMatcher = mitmMatcher;
188         }
```

```
189
190         public AddressResolverGroup<?> resolver() {
191             return resolver;
192         }
193
194         public Iterable<String> getCiphers() {
195             return ciphers;
196         }
197
198         public void setCiphers(Iterable<String> ciphers) {
199             this.ciphers = ciphers;
200         }
201
202         public int getMaxInitialLineLength() {
203             return maxInitialLineLength;
204         }
205
206         public void setMaxInitialLineLength(int maxInitialLineLength) {
207             this.maxInitialLineLength = maxInitialLineLength;
208         }
209
210         public int getMaxHeaderSize() {
211             return maxHeaderSize;
212         }
213
214         public void setMaxHeaderSize(int maxHeaderSize) {
215             this.maxHeaderSize = maxHeaderSize;
216         }
217
218         public int getMaxChunkSize() {
219             return maxChunkSize;
220         }
221
222         public void setMaxChunkSize(int maxChunkSize) {
223             this.maxChunkSize = maxChunkSize;
```

```
224         }
225
226         public IdleStateCheck getIdleStateCheck() {
227             return idleStateCheck;
228         }
229
230         public void setIdleStateCheck(IdleStateCheck idleStateCheck) {
231             this.idleStateCheck = idleStateCheck;
232         }
233
234         public static class Builder {
235             private SslContext clientSslCtx;
236             private String issuer;
237             private Date caNotBefore;
238             private Date caNotAfter;
239             private PrivateKey caPriKey;
240             private PrivateKey serverPriKey;
241             private PublicKey serverPubKey;
242             private EventLoopGroup proxyLoopGroup;
243             private int bossGroupThreads;
244             private int workerGroupThreads;
245             private int proxyGroupThreads;
246             private boolean handleSsl;
247             private HttpProxyAcceptHandler httpProxyAcceptHandler;
248             private HttpProxyAuthenticationProvider authenticationProvider;
249             private final AddressResolverGroup<? extends SocketAddress> resolver;
250             private int maxInitialLineLength = HttpObjectDecoder.DEFAULT_MAX_INITIAL_LINE_LENGTH;
251             private int maxHeaderSize = HttpObjectDecoder.DEFAULT_MAX_HEADER_SIZE;
252             private int maxChunkSize = HttpObjectDecoder.DEFAULT_MAX_CHUNK_SIZE;
253             private IdleStateCheck idleStateCheck;
254
255             public Builder() {
256                 this(DefaultAddressResolverGroup.INSTANCE);
257             }
258
```

```
259             public Builder(final AddressResolverGroup<? extends SocketAddress> resolver) {
260                 this.resolver = resolver;
261             }
262
263             public Builder setClientSslCtx(SslContext clientSslCtx) {
264                 this.clientSslCtx = clientSslCtx;
265                 return this;
266             }
267
268             public Builder setIssuer(String issuer) {
269                 this.issuer = issuer;
270                 return this;
271             }
272
273             public Builder setCaNotBefore(Date caNotBefore) {
274                 this.caNotBefore = caNotBefore;
275                 return this;
276             }
277
278             public Builder setCaNotAfter(Date caNotAfter) {
279                 this.caNotAfter = caNotAfter;
280                 return this;
281             }
282
283             public Builder setCaPriKey(PrivateKey caPriKey) {
284                 this.caPriKey = caPriKey;
285                 return this;
286             }
287
288             public Builder setServerPriKey(PrivateKey serverPriKey) {
289                 this.serverPriKey = serverPriKey;
290                 return this;
291             }
292
```

```
293             public Builder setServerPubKey(PublicKey serverPubKey) {
294                 this.serverPubKey = serverPubKey;
295                 return this;
296             }
297 
298             public Builder setProxyLoopGroup(EventLoopGroup proxyLoopGroup) {
299                 this.proxyLoopGroup = proxyLoopGroup;
300                 return this;
301             }
302 
303             public Builder setHandleSsl(boolean handleSsl) {
304                 this.handleSsl = handleSsl;
305                 return this;
306             }
307 
308             public Builder setBossGroupThreads(int bossGroupThreads) {
309                 this.bossGroupThreads = bossGroupThreads;
310                 return this;
311             }
312 
313             public Builder setWorkerGroupThreads(int workerGroupThreads) {
314                 this.workerGroupThreads = workerGroupThreads;
315                 return this;
316             }
317 
318             public Builder setProxyGroupThreads(int proxyGroupThreads) {
319                 this.proxyGroupThreads = proxyGroupThreads;
320                 return this;
321             }
322 
323             public Builder setHttpProxyAcceptHandler(final HttpProxyAcceptHandler httpProxyAcceptHandler) {
324                 this.httpProxyAcceptHandler = httpProxyAcceptHandler;
325                 return this;
326             }
```

```
327
328             public Builder setAuthenticationProvider(final HttpProxyAuthenticationProvider authenticationProvider)
329                 this.authenticationProvider = authenticationProvider;
330                 return this;
331             }
332
333             public Builder setMaxInitialLineLength(int maxInitialLineLength) {
334                 this.maxInitialLineLength = maxInitialLineLength;
335                 return this;
336             }
337
338             public Builder setMaxHeaderSize(int maxHeaderSize) {
339                 this.maxHeaderSize = maxHeaderSize;
340                 return this;
341             }
342
343             public Builder setMaxChunkSize(int maxChunkSize) {
344                 this.maxChunkSize = maxChunkSize;
345                 return this;
346             }
347
348             public Builder setIdleStateCheck(IdleStateCheck idleStateCheck) {
349                 this.idleStateCheck = idleStateCheck;
350                 return this;
351             }
352
353             public HttpProxyServerConfig build() {
354                 HttpProxyServerConfig config = new HttpProxyServerConfig(this);
355                 return config;
356             }
357         }
358     }
```

Tomas Camilo Ostos Medina-202123214

Clase IdleStateCheck

```java
public class IdleStateCheck {
    private long readerIdleTime;
    private long writerIdleTime;
    private long allIdleTime;

    public IdleStateCheck() {
        this.readerIdleTime = 10;
        this.writerIdleTime = 15;
        this.allIdleTime = 20;
    }

    public IdleStateCheck(long readerIdleTime, long writerIdleTime, long allIdleTime) {
        this.readerIdleTime = readerIdleTime;
        this.writerIdleTime = writerIdleTime;
        this.allIdleTime = allIdleTime;
    }

    public long getReaderIdleTime() {
        return readerIdleTime;
    }

    public long getWriterIdleTime() {
        return writerIdleTime;
    }

    public long getAllIdleTime() {
        return allIdleTime;
    }

    public void setReaderIdleTime(long readerIdleTime) {
        this.readerIdleTime = readerIdleTime;
    }

    public void setWriterIdleTime(long writerIdleTime) {
        this.writerIdleTime = writerIdleTime;
    }
```

Codigo de uso en proyecto personal

```xml
<dependency>
    <groupId>com.github.monkeywie</groupId>
    <artifactId>proxyee</artifactId>
    <version>1.7.6</version>
</dependency>
```

Bibliografía

1. monkeyWie/proxyee: HTTP proxy server,support HTTPS&websocket.MITM impl,intercept and tamper HTTPS traffic. (github.com)
2. Tratando de entenderlo: Patrones de diseño: Proxy
3. Patrones de Diseno de Programacion Orientada a Objetos (soloentendidos.com)
4. (1435) CURSO DE PATRONES DE DISEÑO: PROXY - YouTube

Tomas Camilo Ostos Medina-202123214