Chain of Responsibility

Example Problem - use this pattern when a pipeline / chain of objects is required for processing a request. A web server is an example - at a minimum a server has to deal with a http request. One way of doing this may be to have a HttpRequest Class, a Web Server class with a handle method which takes a HttpRequest and then performs the various steps required -

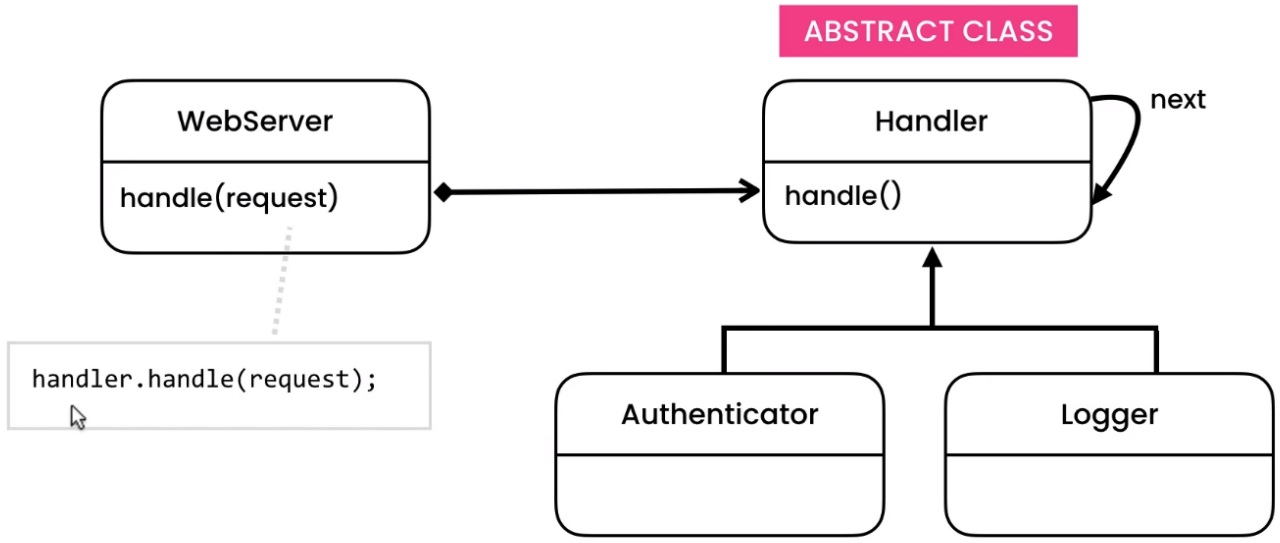
Authentication

Logging

Compression etc

With a class for each step to ensur the Single Responsibility Principle is not violated. Each clas can then be initialised in the handle method of the WebServer and each method called in turn. The issue with this approach is that the Web Server class is tightly coupled with each of the pipleline class and we should code to an interface and use inheritance to solve this problem and also ensure the OCP is not violated in the WebServer i.e. if you add a new task to the cahin you do not have to update this class.

Solution



Using one abstract class rather thatn a series of interfaces for each task ensures that the hanlde method in the WebServer class is loosely copled to the concrete implementations of the Handler Class and only calls handler.handle(request) once rather than a series of times

Handler	WebServer	HttpRequest	Compressor

```
public abstract class Handler {
  private Handler next;

 public Handler(Handler next) {
    this.next = next;
 }

 //If this is not the last handler then call the next handler in the chain

//Concrete Implementation of doHandle lives in the concrete handler (giving a consistent interface) - but it's called here. the constructor in each concrete handler will pas the next handler to this super class

  public void handle(HttpRequest request) {
    if (doHandle(request))
      return;

 if (next != null)
     next.handle(request);
 }

  public abstract boolean doHandle(HttpRequest request);
}
```

```
public class WebServer {
 //maintain a reference to the first handler in the chain

  private Handler handler;


 public WebServer (Handler handler) {
    this.handler = handler;
 }

  public void handle (HttpRequest request) {
    handler.handle (request);
 }
}
```

```
public class HttpRequest {
  private String username;
 private String password;

 public HttpRequest (String username, String password) {
    this.username = username;
 this.password = password;
 }

  public String getUsername() {
    return username;
 }

  public String getPassword() {
    return password;
 }
}
```

```
public class Compressor extends Handler {
  public Compressor (Handler next) {
    super(next);
 }

  @Override
  public boolean doHandle (HttpRequest request) {
    System.out.println("Compress");

 return false;
 }
}
```

Encryptor

```
public class Encryptor extends Handler {
  public Encryptor(Handler next) {
    super(next);
 }

  @Override
  public boolean doHandle (HttpRequest request) {
    System.out.println("Encryption");
 return false;
 }
}
```

Authenticator

```
public class Authenticator extends Handler {
  public Authenticator (Handler next) {
    super(next);
 }

  @Override
  public boolean doHandle (HttpRequest request) {
    var isValid = (request.getUsername() == "admin" &&

request.getPassword() == "1234");

 System.out.println("Authentication");

 return !isValid;
 }
}
```

Logger

```
public class Logger extends Handler {
  public Logger (Handler next) {
    super(next);
 }

  @Override
  public boolean doHandle (HttpRequest request) {
    System.out.println("Log");

 return false;
 }
}
```

Main

```
public class Logger {


 public static void main(String[] args) {
```

//authenticator logger commperssor - start with last

```
var compressor = new Compressor(null);

var logger = new Logger(compressor);

var auth = new Authenticator(logger);

var server = new WebServer(auth);
```

//all three handlers are called in the pipeline unless you provide an invalid user and then the chain will stop there

//new steps can be added by extending the handler class and including in the main method no modification of the other classes are required

```
server.handle(new Httprequest("admin", "1234");

}

}
```