Intercepting Filter

Pola Desain Perangkat Lunak

Tjatur Kandaga G.

Fakultas IT - UK Maranatha

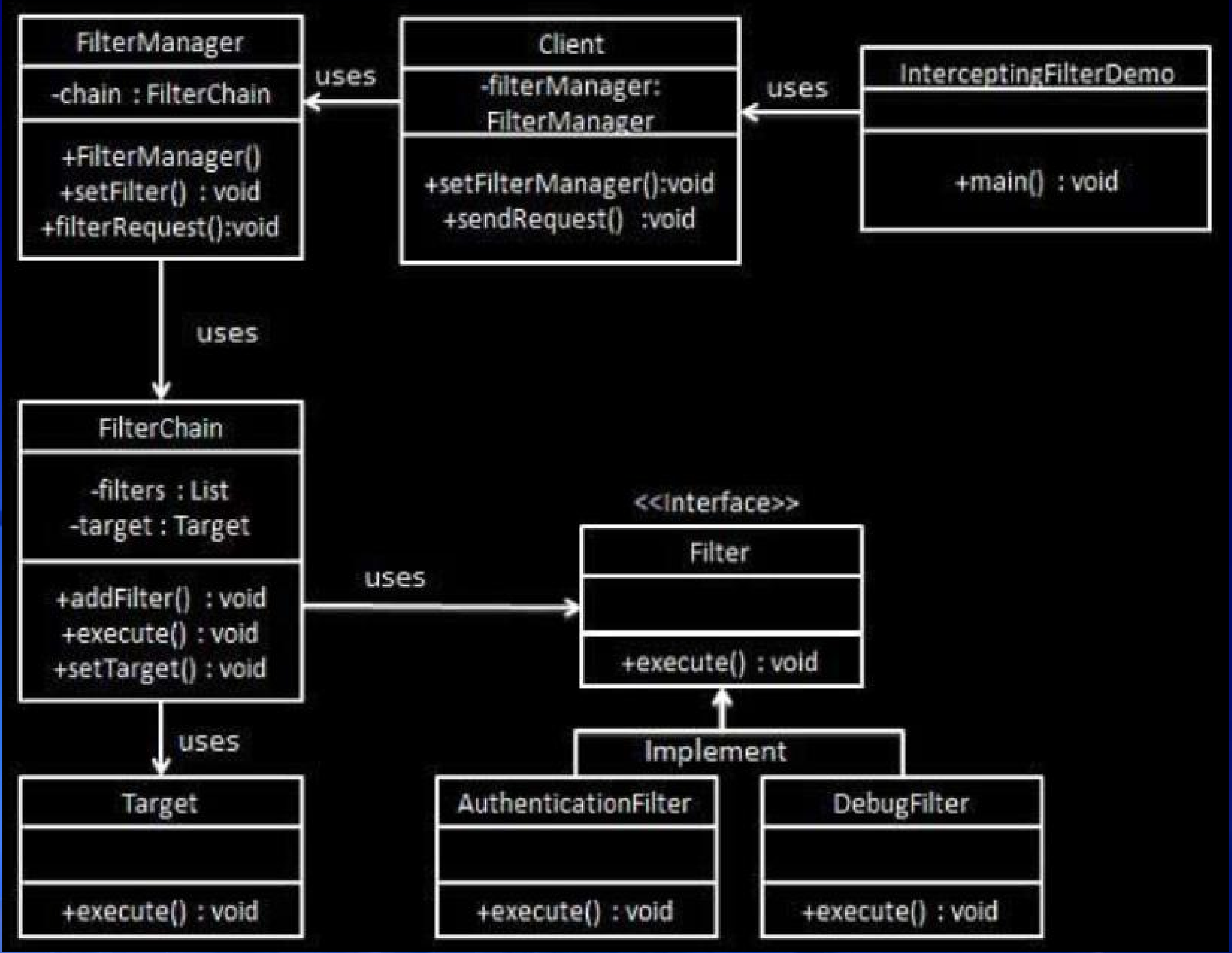
Mengenal Intercepting Filter

- Intercepting filter pattern digunakan ketika kita membutuhkan pre-processing / post-processing bersamaan dengan request atau response dari aplikasi.
- Filter-filter didefinisikan dan diterapkan pada sebuah request sebelum request tersebut diteruskan ke aplikasi targetnya.
- Filter-filter dapat melakukan authentication / authorization / logging atau tracking request kemudian baru mengirimkan request tersebut ke handlernya.

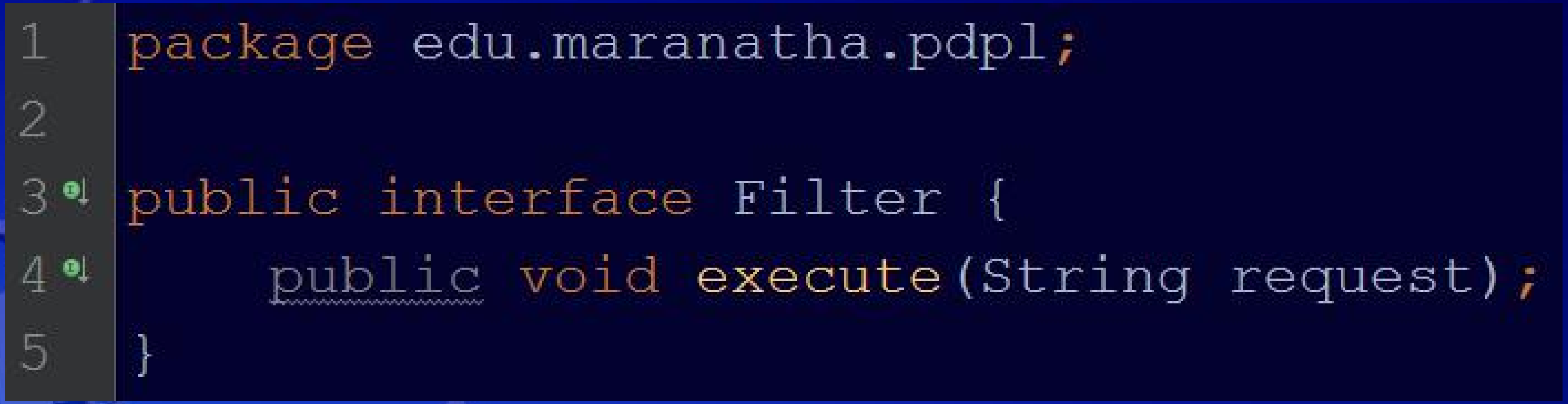
Mengenal Intercepting Filter

- Entitas-entitas yang terlibat dalam design pattern ini:
  - Filter: melakukan proses sebelum atau setelah request diproses oleh handler.
  - Filter Chain: berisi banyak filter yang akan dijalankan pada target sesuai urutan tertentu.
  - Target: request handler.
  - Filter Manager: mengatur filter-filter dan Filter Chain.
  - Client: yang meminta request ke target.

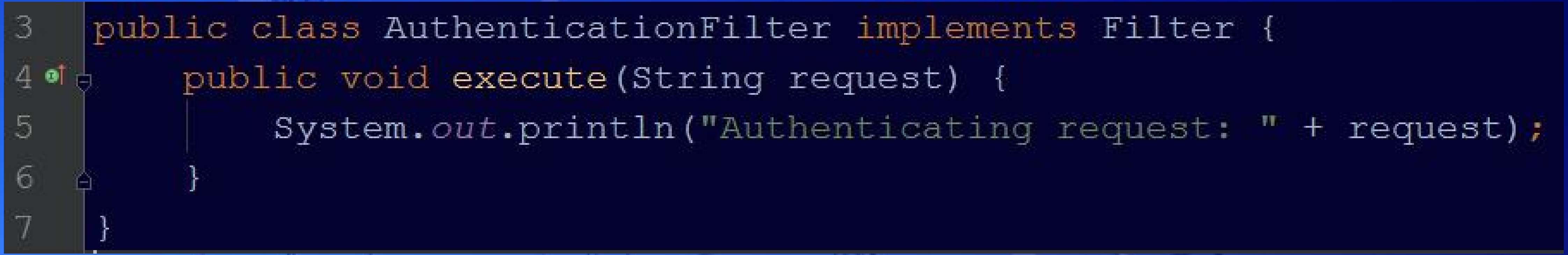
Diagram Kelas Intercepting Filter



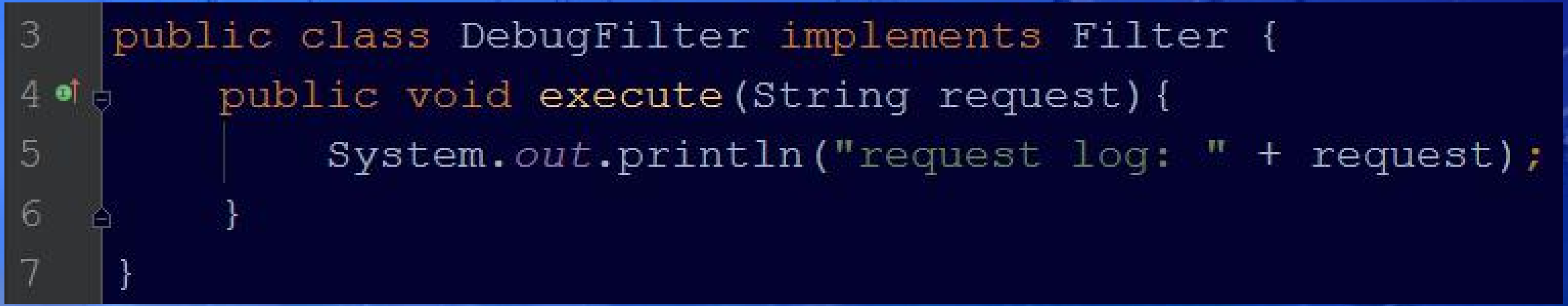
Contoh Intercepting Filter



```java
package edu.maranatha.pdpl;

public interface Filter {
    public void execute(String request);
}
```

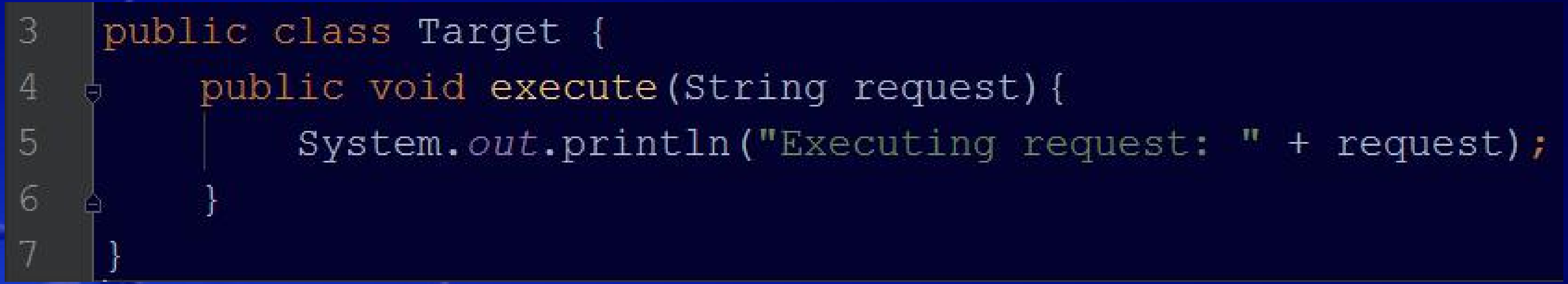


```java
public class AuthenticationFilter implements Filter {
    public void execute(String request) {
        System.out.println("Authenticating request: " + request);
    }
}
```

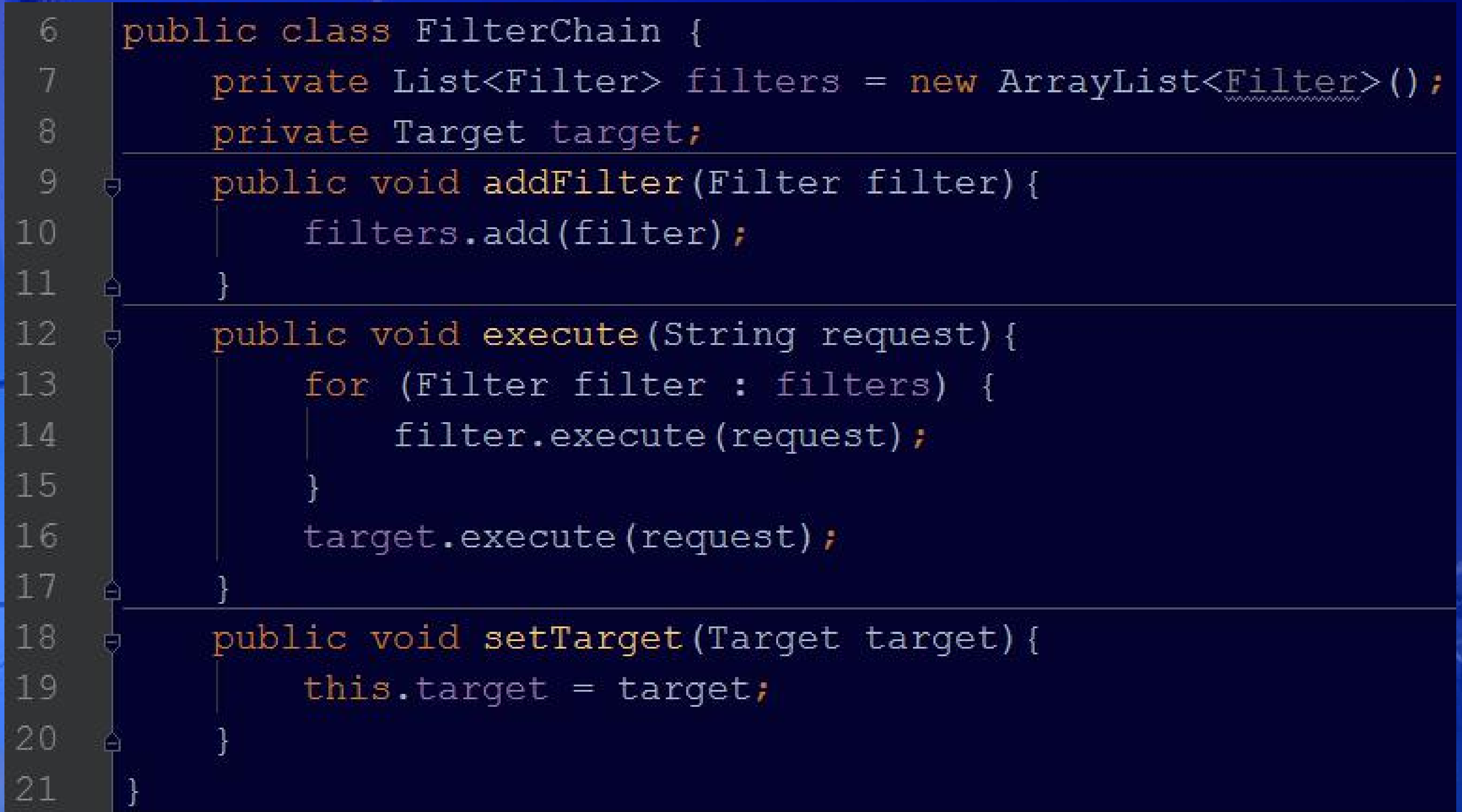


```java
public class DebugFilter implements Filter {
    public void execute(String request){
        System.out.println("request log: " + request);
    }
}
```

Contoh Intercepting Filter

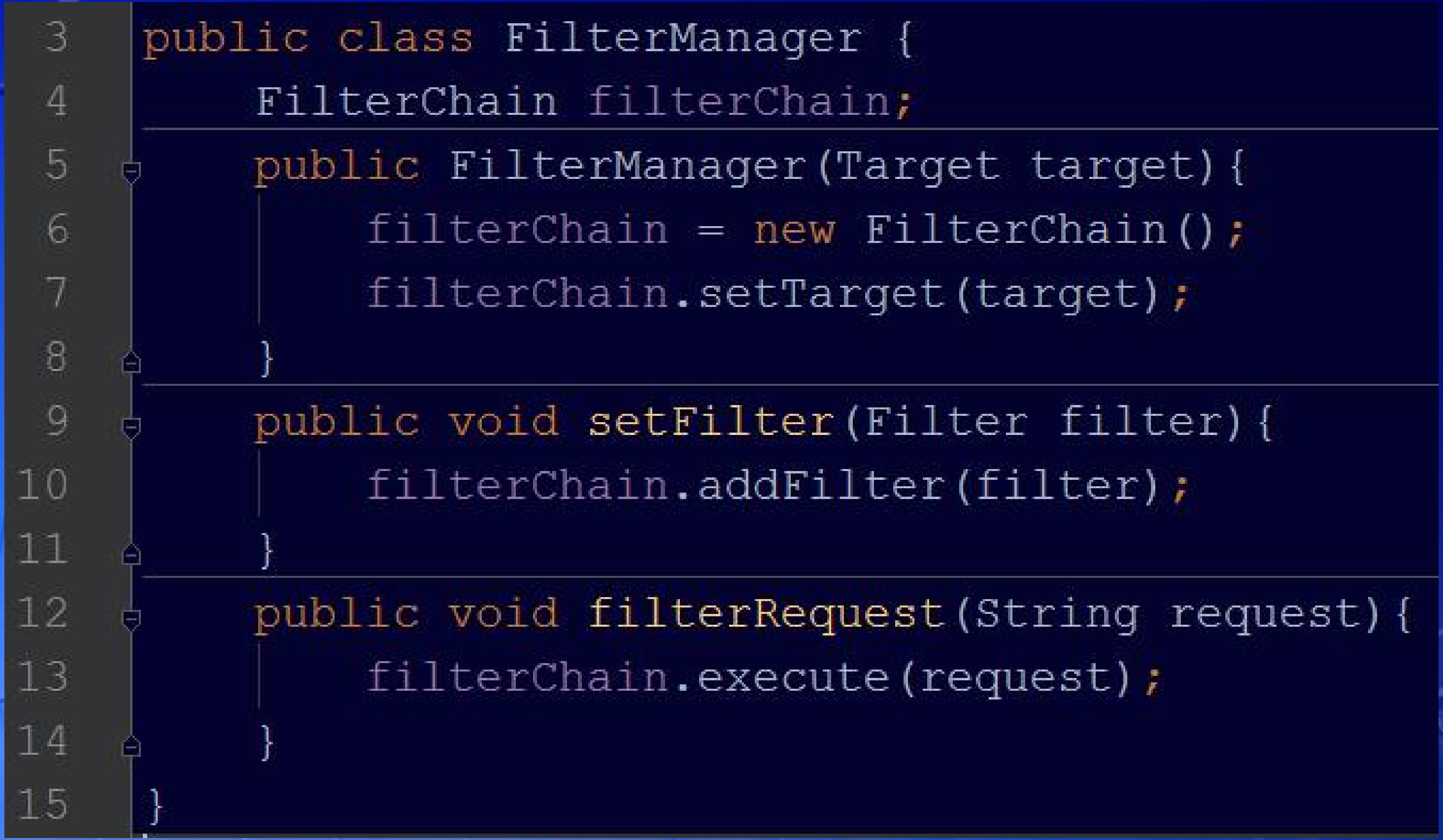


```java
public class Target {
    public void execute(String request){
        System.out.println("Executing request: " + request);
    }
}
```



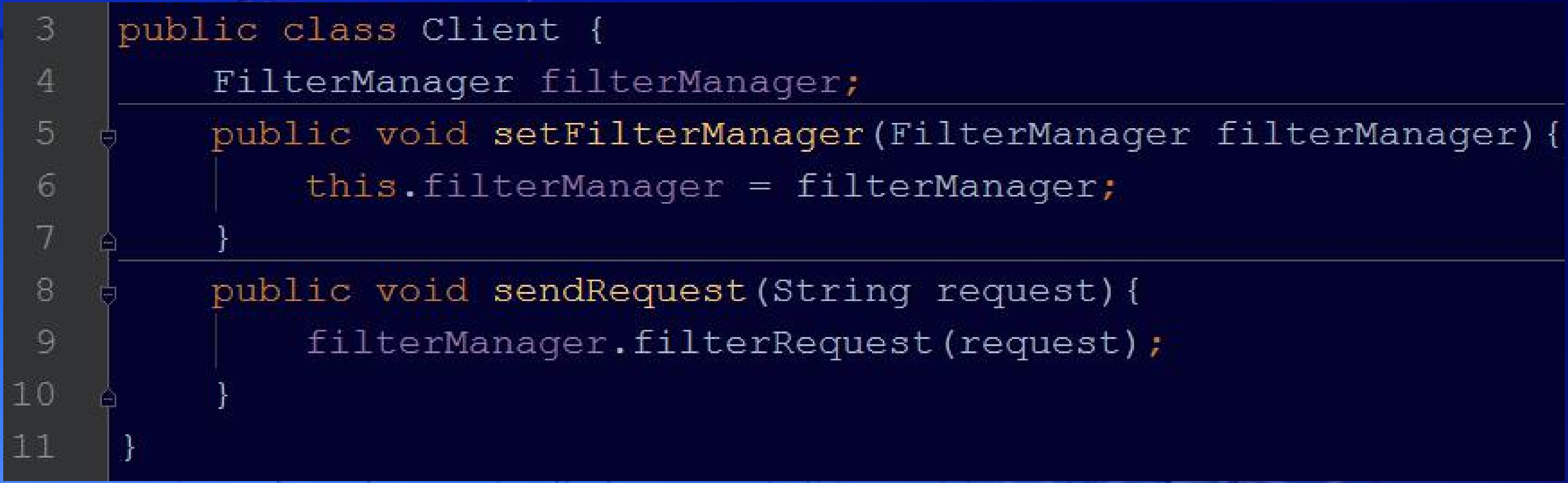
```java
public class FilterChain {
    private List<Filter> filters = new ArrayList<Filter>();
    private Target target;
    public void addFilter(Filter filter){
        filters.add(filter);
    }
    public void execute(String request){
        for (Filter filter : filters) {
            filter.execute(request);
        }
        target.execute(request);
    }
    public void setTarget(Target target){
        this.target = target;
    }
}
```

Contoh Intercepting Filter



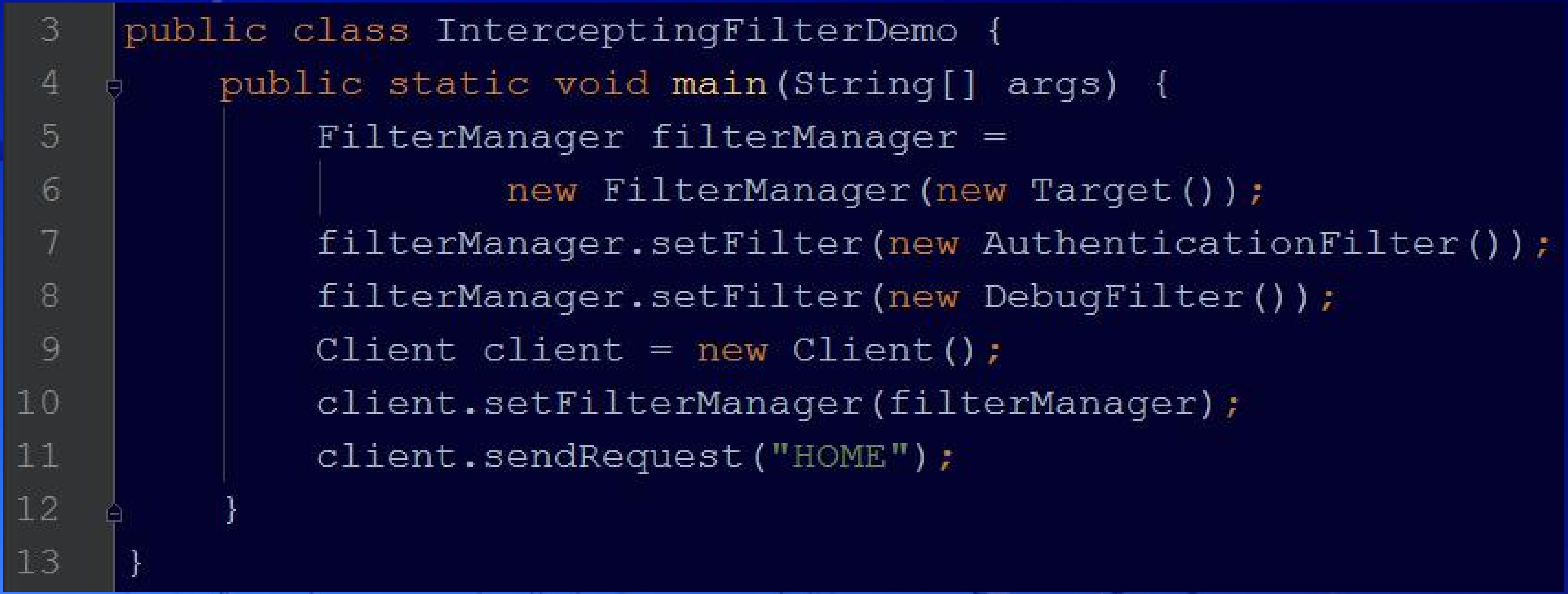
```java
public class FilterManager {
    FilterChain filterChain;
    public FilterManager(Target target){
        filterChain = new FilterChain();
        filterChain.setTarget(target);
    }
    public void setFilter(Filter filter){
        filterChain.addFilter(filter);
    }
    public void filterRequest(String request){
        filterChain.execute(request);
    }
}
```

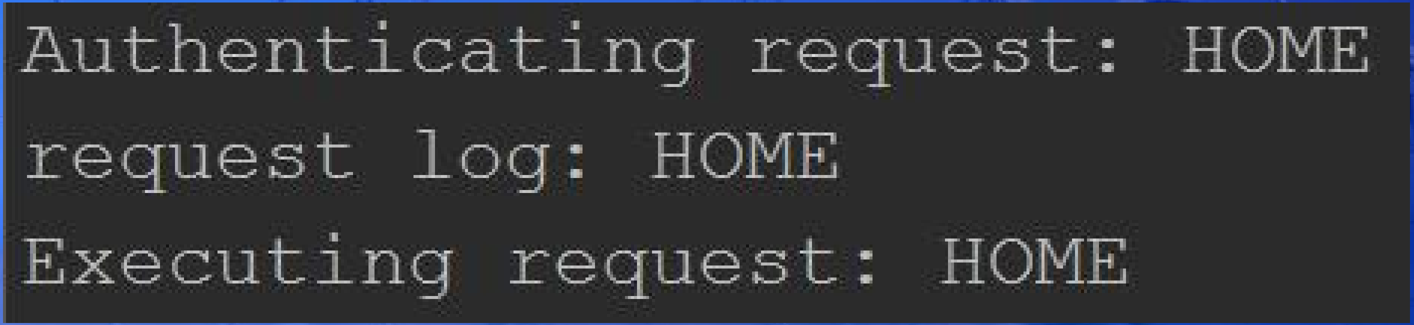
Contoh Intercepting Filter



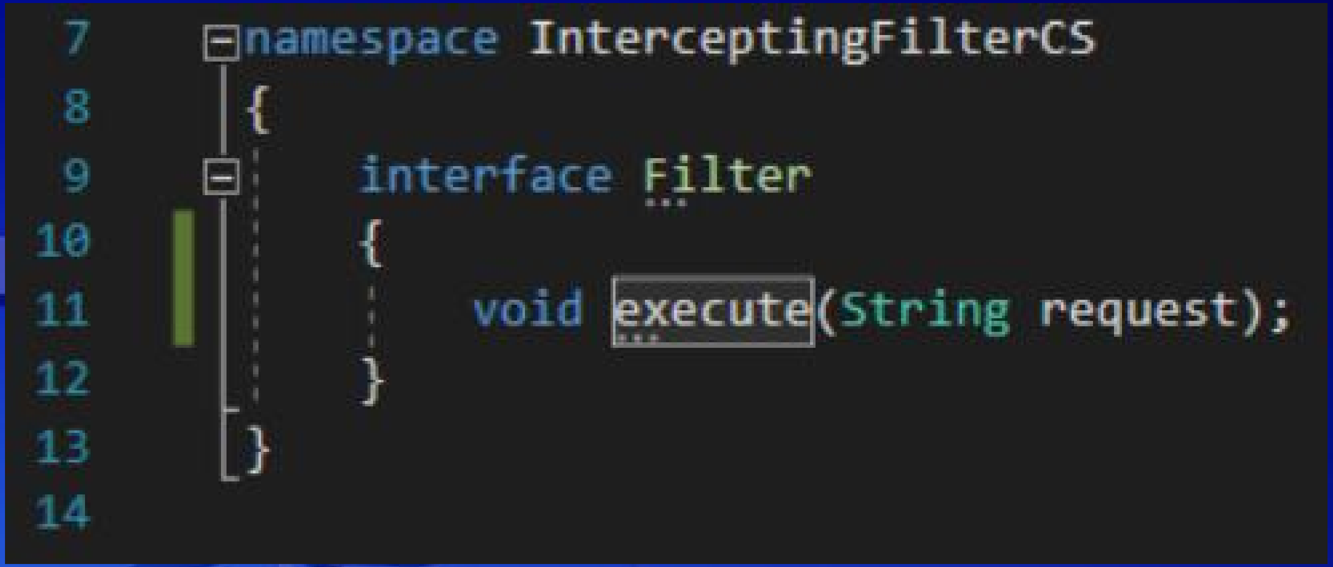
```
public class Client {
    FilterManager filterManager;
    public void setFilterManager(FilterManager filterManager){
        this.filterManager = filterManager;
    }
    public void sendRequest(String request){
        filterManager.filterRequest(request);
    }
}
```

Contoh Intercepting Filter

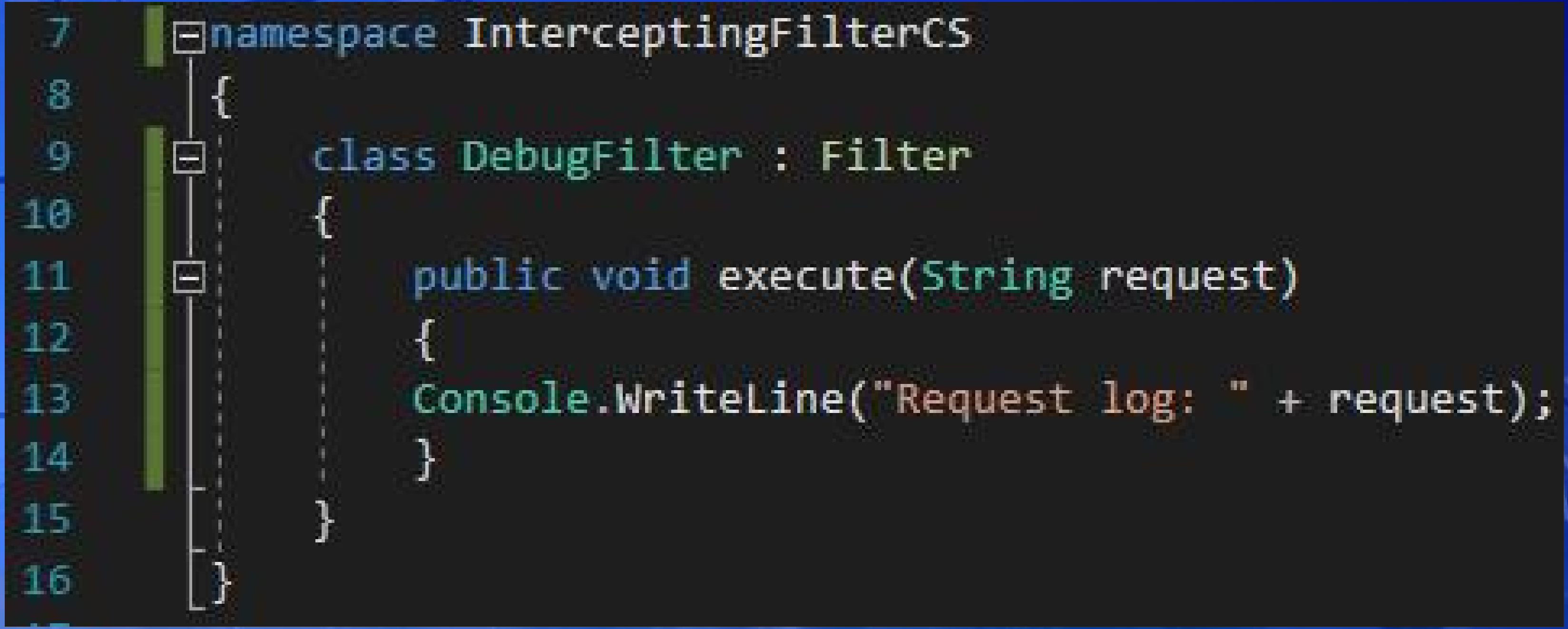

```java
public class InterceptingFilterDemo {
    public static void main(String[] args) {
        FilterManager filterManager =
                new FilterManager(new Target());
        filterManager.setFilter(new AuthenticationFilter());
        filterManager.setFilter(new DebugFilter());
        Client client = new Client();
        client.setFilterManager(filterManager);
        client.sendRequest("HOME");
    }
}
```


```
Authenticating request: HOME
request log: HOME
Executing request: HOME
```

Contoh Intercepting Filter (C#)

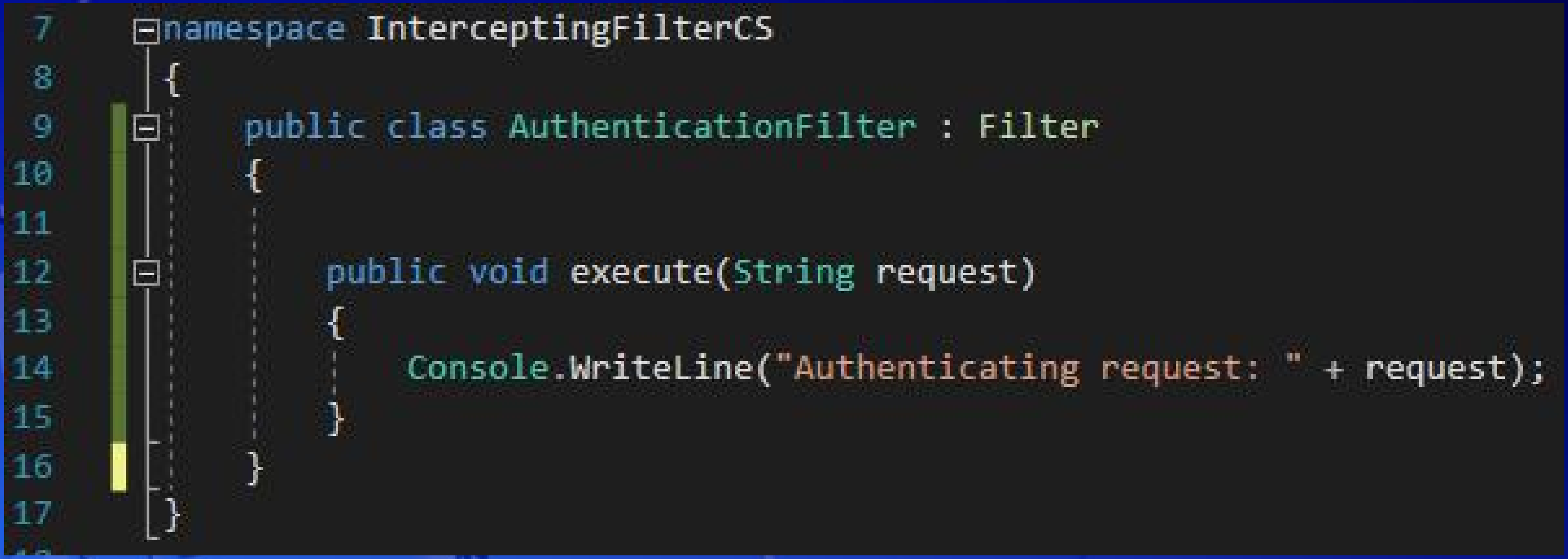


```
namespace InterceptingFilterCS
{
    interface Filter
    {
        void execute(String request);
    }
}
```

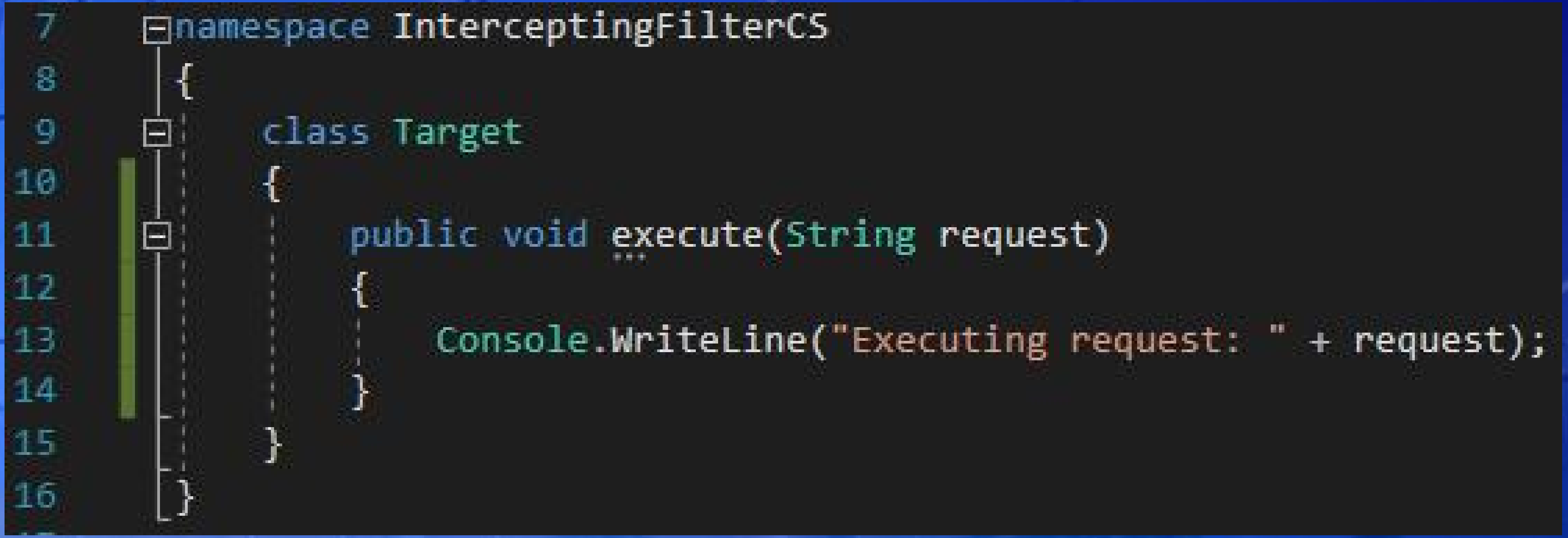


```
namespace InterceptingFilterCS
{
    class DebugFilter : Filter
    {
        public void execute(String request)
        {
        Console.WriteLine("Request log: " + request);
        }
    }
}
```

Contoh Intercepting Filter (C#)

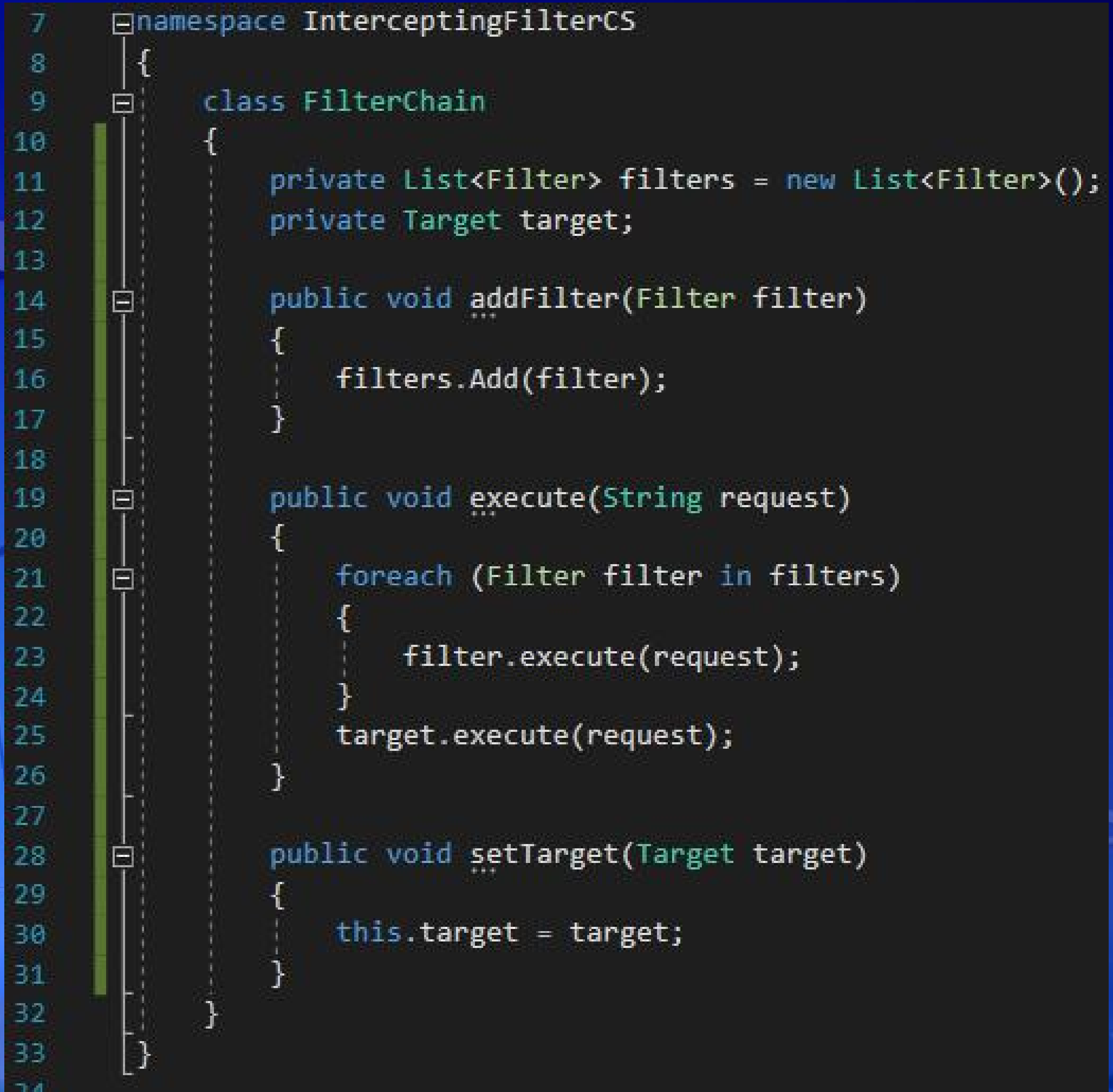


```
namespace InterceptingFilterCS
{
    public class AuthenticationFilter : Filter
    {

        public void execute(String request)
        {
            Console.WriteLine("Authenticating request: " + request);
        }
    }
}
```



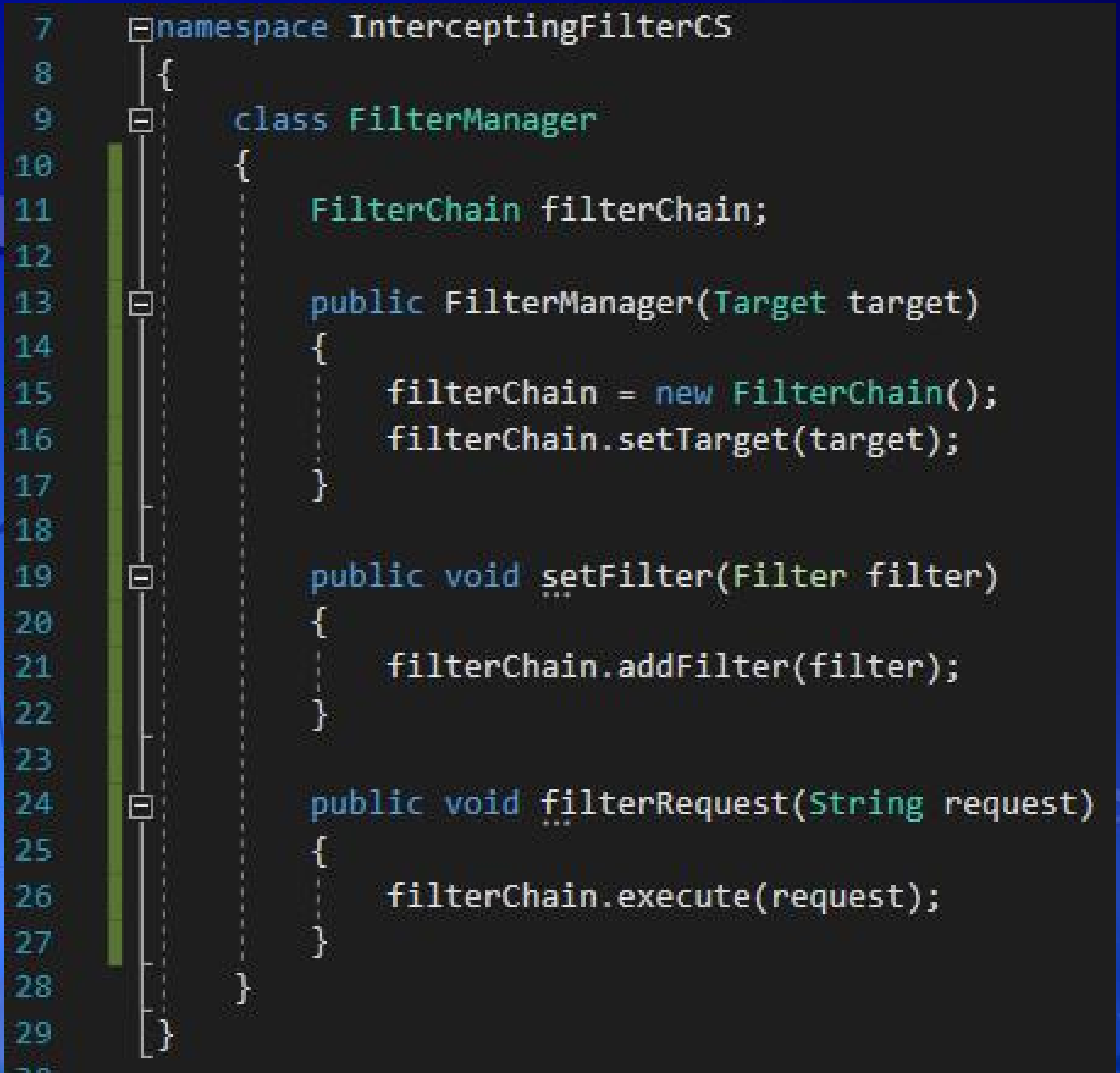
```
namespace InterceptingFilterCS
{
    class Target
    {
        public void execute(String request)
        {
            Console.WriteLine("Executing request: " + request);
        }
    }
}
```

Contoh Intercepting Filter (C#)



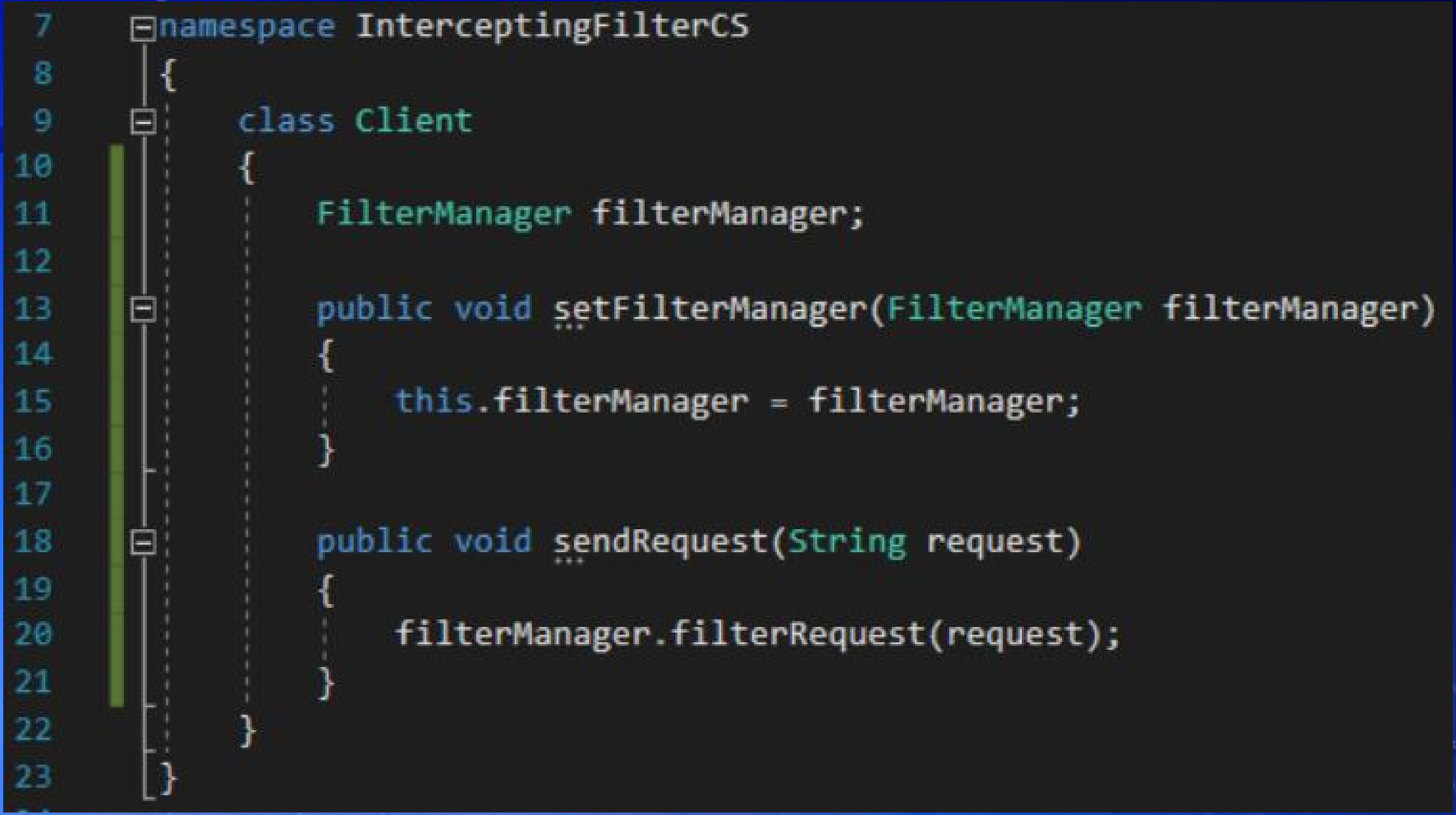
```
namespace InterceptingFilterCS
{
    class FilterChain
    {
        private List<Filter> filters = new List<Filter>();
        private Target target;

        public void addFilter(Filter filter)
        {
            filters.Add(filter);
        }

        public void execute(String request)
        {
            foreach (Filter filter in filters)
            {
                filter.execute(request);
            }
            target.execute(request);
        }

        public void setTarget(Target target)
        {
            this.target = target;
        }
    }
}
```

Contoh Intercepting Filter (C#)



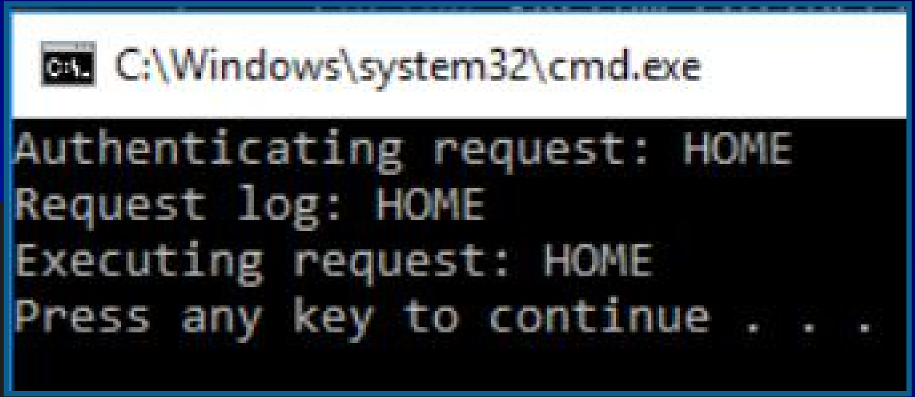
```
namespace InterceptingFilterCS
{
    class FilterManager
    {
        FilterChain filterChain;

        public FilterManager(Target target)
        {
            filterChain = new FilterChain();
            filterChain.setTarget(target);
        }

        public void setFilter(Filter filter)
        {
            filterChain.addFilter(filter);
        }

        public void filterRequest(String request)
        {
            filterChain.execute(request);
        }
    }
}
```

Contoh Intercepting Filter (C#)

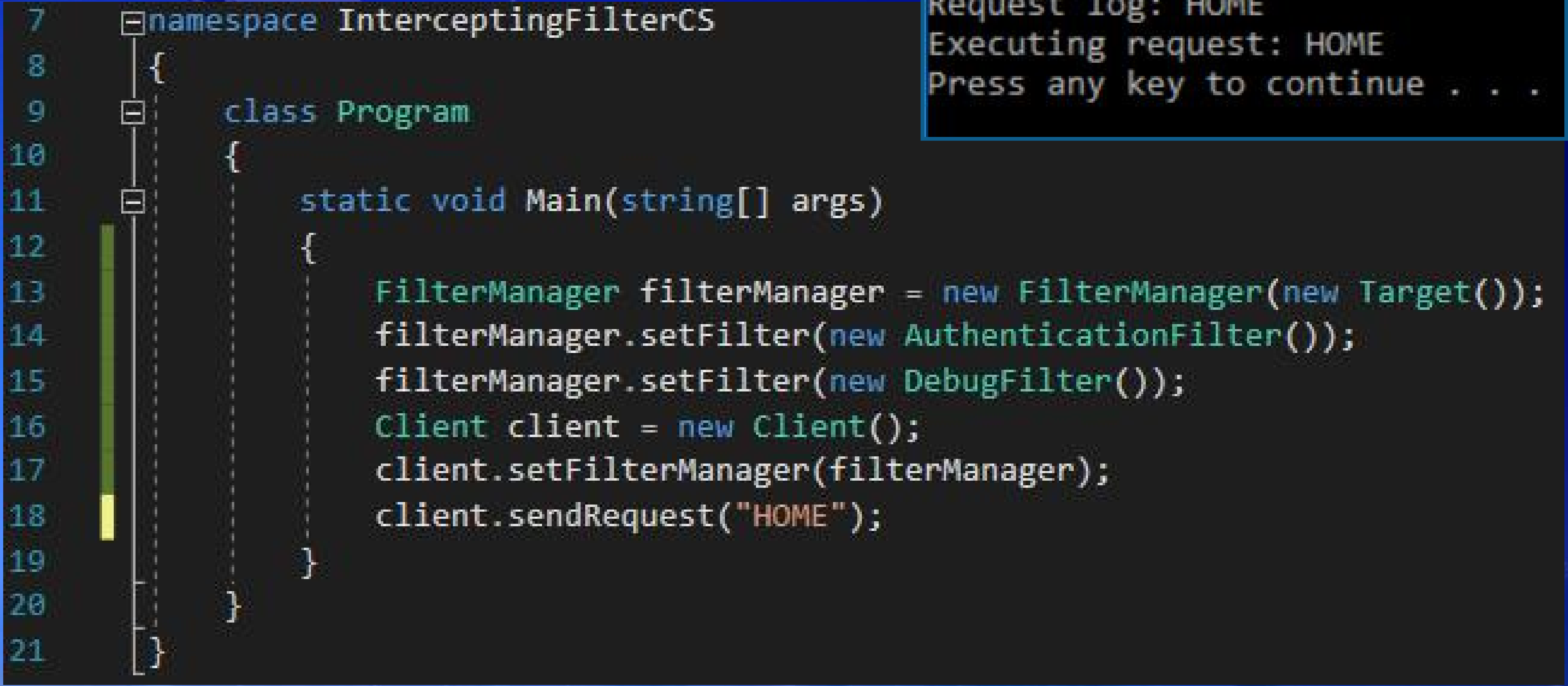


```
namespace InterceptingFilterCS
{
    class Client
    {
        FilterManager filterManager;

        public void setFilterManager(FilterManager filterManager)
        {
            this.filterManager = filterManager;
        }

        public void sendRequest(String request)
        {
            filterManager.filterRequest(request);
        }
    }
}
```

Contoh Intercepting Filter (C#)


C:\Windows\system32\cmd.exe

```
Authenticating request: HOME
Request log: HOME
Executing request: HOME
Press any key to continue . . .
```



```csharp
namespace InterceptingFilterCS
{
    class Program
    {
        static void Main(string[] args)
        {
            FilterManager filterManager = new FilterManager(new Target());
            filterManager.setFilter(new AuthenticationFilter());
            filterManager.setFilter(new DebugFilter());
            Client client = new Client();
            client.setFilterManager(filterManager);
            client.sendRequest("HOME");
        }
    }
}
```

Intercepting Filter add pre/post-processings to client request.