Filter Design Pattern

Ajit Singh October 11, 2016 No Comments

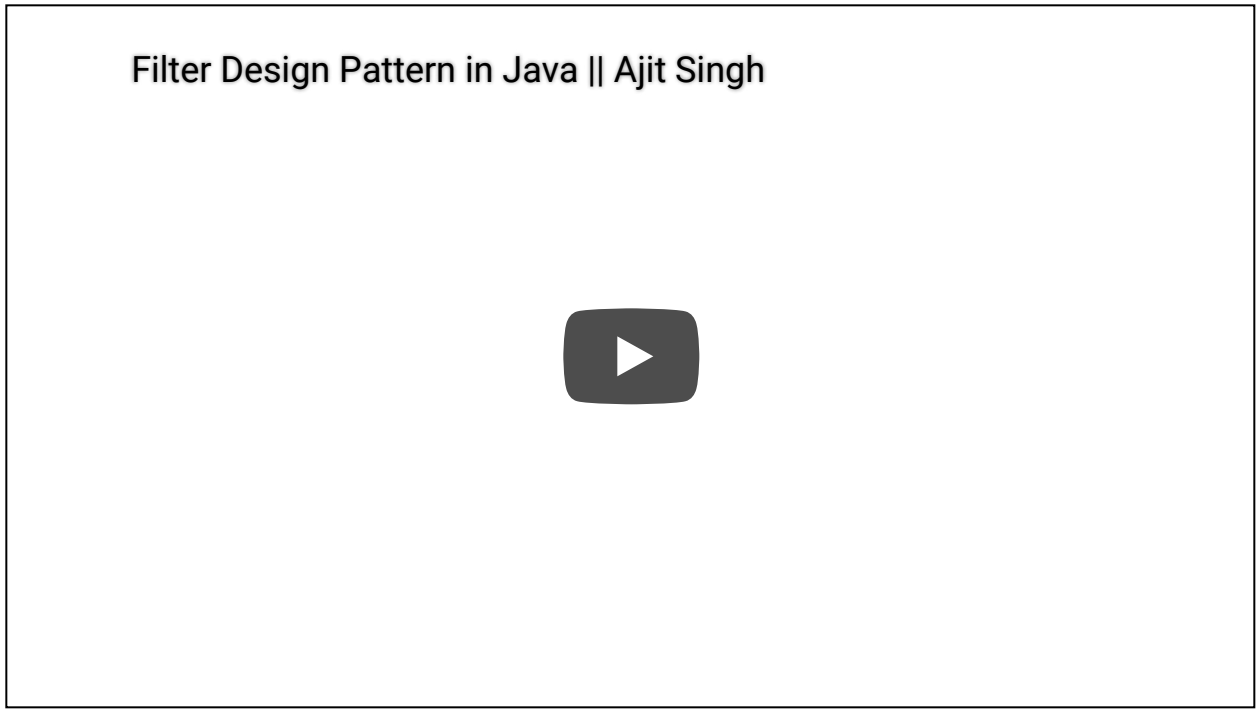
What Is Filter Design Pattern?

Filter design pattern is used for building a criteria to filter items or objects dynamically. You can choose your own criteria and apply it on your objects to filter out the desired objects.

When to use Filter Design Pattern?

Its easy to figure out when one should use filter design pattern. When you have a requirement where you want to add filters dynamically or you are implementing multiple functionalities and most of them require different filter criteria to filter something. In that case instead of hard coding the filters inside the functionalities, you can create filter criteria and re-use it wherever required.

Video



Example:

Lets understand Filter Design Pattern with an example. All the code used below is available on github.

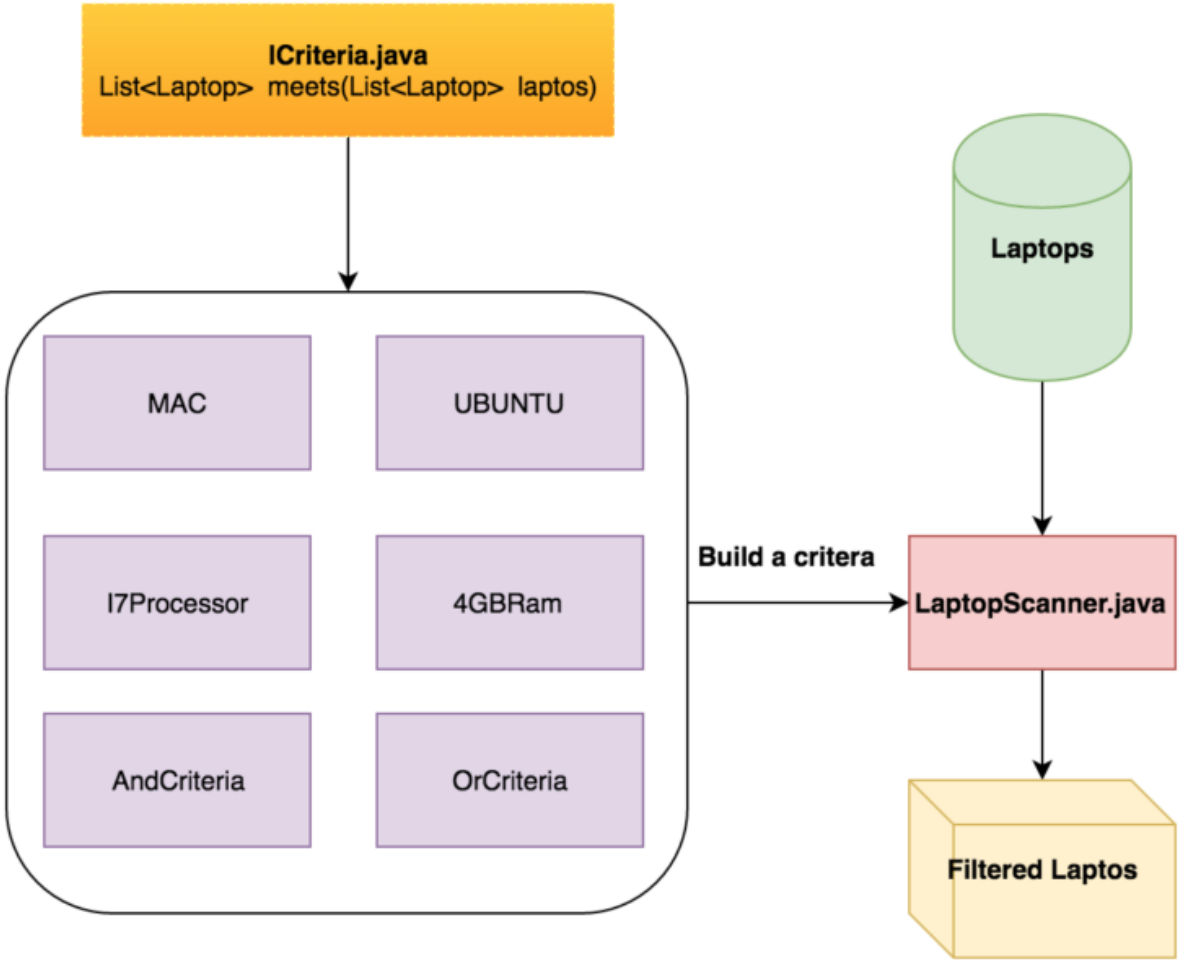
Lets build a hypothetical scenario to understand this pattern better.

You are building a tool to help tech-ops team in your company to distribute laptops to employees according to their configuration requirements. Each role in company has a different configuration requirement. Now tech-ops team have a very hard time to filter out the laptops as per the employee requirement because its a manual process.

The tool you are building provides a functionality to the employees to add filters to search for their desired laptop. The filters may change across the employees.

This is a very good place to use the filter design pattern, as you know that filters may change across employees. Now lets implement the solution.

Design



Lets define the interface which every criteria has to implement. This interface has a method which meets the employee requirement and returns the filtered laptops.

Criteria.java

```
package filter_pattern.filter_criteria;


import filter_pattern.model.Laptop;

import java.util.List;

public interface Criteria {
    List<Laptop> meets(List<Laptop> laptops);
}
```

Now lets implement a bunch of criteria to choose from.

Macintosh.java

```
package filter_pattern.filter_criteria;


import filter_pattern.model.Laptop;

import java.util.List;

import static java.util.stream.Collectors.toList;

public class Macintosh implements Criteria {
```

```
    @Override
    public List<Laptop> meets(List<Laptop> laptops) {
        return laptops.stream().filter(laptop -> laptop.getOperatingSystem().equals("MAC")).collect(to
    }
}
```

4GBRam.java

```
package filter_pattern.filter_criteria;

import filter_pattern.model.Laptop;

import java.util.List;

import static java.util.stream.Collectors.toList;

public class Ram4GB implements Criteria {
    @Override
    public List<Laptop> meets(List<Laptop> laptops) {
        return laptops.stream().filter(laptop -> laptop.getRam().equals("4GB")).collect(toList());
    }
}
```

I7Processor.java

```
package filter_pattern.filter_criteria;


import filter_pattern.model.Laptop;
```

```
import java.util.List;

import static java.util.stream.Collectors.toList;

public class I7Processor implements Criteria {
    @Override
    public List<Laptop> meets(List<Laptop> laptops) {
        return laptops.stream().filter(laptop -> laptop.getProcessor().equals("i7")).collect(toList())
    }
}
```

ScreenSize15Inch.java

```
package filter_pattern.filter_criteria;

import filter_pattern.model.Laptop;

import java.util.List;

import static java.util.stream.Collectors.toList;

public class ScreenSize15Inch implements Criteria {
    @Override
    public List<Laptop> meets(List<Laptop> laptops) {
        return laptops.stream().filter(laptop -> laptop.getScreenSize().equals("15inch")).collect(toLi
    }
}
```

Likewise there can be a lot of other criteria e.g Windows.java, Ubuntu.java, 8GBRam.java, I3Processor.java etc.

There is two more criteria which are very important AndCriteria and OrCriteria. Both are used to combine multiple criteria.

AndCritera.java

```
package filter_pattern.filter_criteria;


import filter_pattern.model.Laptop;

import java.util.List;

public class AndCriteria implements Criteria {
    private Criteria[] criterias;

    public AndCriteria(Criteria... criterias) {
        this.criterias = criterias;
    }

    @Override
    public List<Laptop> meets(List<Laptop> laptops) {
        List<Laptop> filteredLaptops = laptops;

        for (Criteria criteria : criterias) {
            filteredLaptops = criteria.meets(filteredLaptops);
        }

        return filteredLaptops;
    }
}
```

Now that we have all criteria implemented, lets use them to filter out the desired laptops.

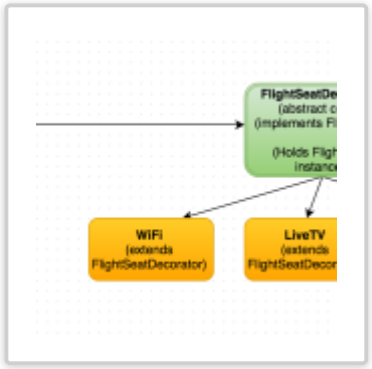
LaptopScanner.java

```
package filter_pattern;

import filter_pattern.filter_criteria.AndCriteria;
import filter_pattern.filter_criteria.I7Processor;
import filter_pattern.filter_criteria.Macintosh;
import filter_pattern.filter_criteria.Ram4GB;
import filter_pattern.model.Laptop;

import java.util.List;

public class LaptopScanner {

    public static void main(String[] args) {
        List<Laptop> laptops = LaptopFactory.manufactureInBulk();

        AndCriteria searchCriteria = new AndCriteria(new Ram4GB(), new Macintosh(), new I7Processor())
        List<Laptop> filteredLaptops = searchCriteria.meets(laptops);

        filteredLaptops.stream().forEach(Laptop::prettyPrint);
    }
}
```

This class combines multiple criteria and prints the laptops which matches with employees configuration.

Thats all folks, I hope now you have a better understanding of filter design pattern. If you have any comments, suggestions or feedback, please leave a comment below. Thanks!

RELATED POSTS



Decorator Design Pattern

Java custom annotations

Junit Rules

Schedule local notification in android

Android with sqlite database

Design Patterns, Java design pattern, java

Related posts

» Java custom annotations
» Decorator Design Pattern
» Schedule local notification in android
» Android with sqlite database
» Instrumentation Testing Of ListView

About Ajit Singh

Hi, I am Ajit Singh, author of singhajit.com. I work on a lot of different technologies and tools. I started my career as a software engineer in Chennai (India) and recently I moved to Gurgaon near to my home town. I like reading and writing about technology.

View all posts by Ajit Singh →