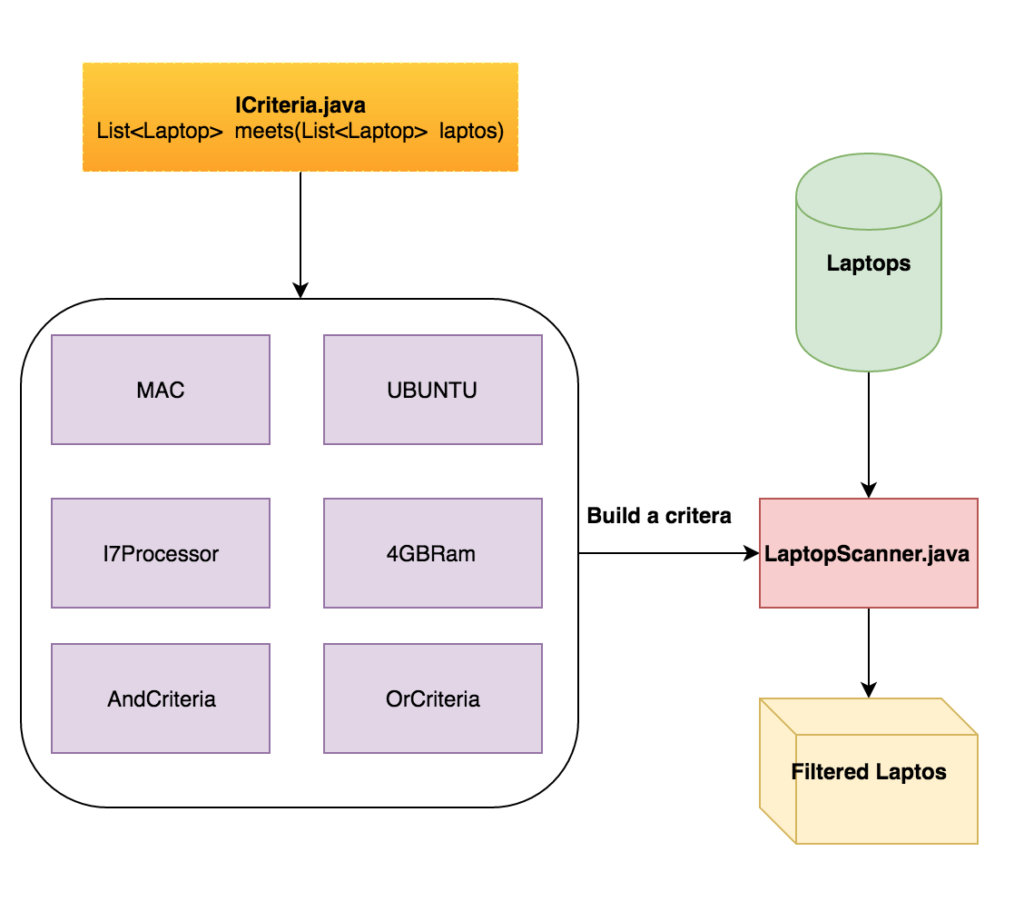
Filter design pattern is used for building a criteria to filter items or objects dynamically. You can choose your own criteria and apply it on your objects to filter out the desired objects.

It’s 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.

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 let’s implement the solution.



package model;

public class Laptop {

private Integer id;

private String hardDisk;

private String processor;

private String ram;

private String operatingSystem;

private String screenSize;

public Laptop(Integer id, String hardDisk, String processor, String ram, String operatingSystem, String screensize) {

this.id = id;

this.hardDisk = hardDisk;

this.processor = processor;

this.ram = ram;

this.operatingSystem = operatingSystem;

this.screenSize = screensize;

}

public String getHardDisk() {

return hardDisk;

}

public void setHardDisk(String hardDisk) {

this.hardDisk = hardDisk;

}

public Integer getId() {

return id;

}

public String getProcessor() {

return processor;

}

public String getRam() {

return ram;

}

public String getOperatingSystem() {

return operatingSystem;

}

public String getScreenSize() {

return screenSize;

}

public void prettyPrint() {

System.out.println("ID = " + id + ";\t" + "Hard Disk = " + hardDisk + ";\t" + "Processor = " + processor + ";\t"

+ "RAM = " + ram + ";\t" + "OS = " + operatingSystem + ";\t" + "Screen Size = " + screenSize);

}

}

import java.util.List;

import java.util.stream.Collectors;

import model.Laptop;

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

import java.util.Arrays;

interface Criteria{

List<Laptop> meets(List<Laptop> laptopList);

}

class Macintosh implements Criteria{

@Override

public List<Laptop> meets(List<Laptop> laptopList) {

return laptopList.stream().filter(l -> l.getOperatingSystem().equals("MAC")).collect(Collectors.toList());

}

}

class Ram4GB implements Criteria{

@Override

public List<Laptop> meets(List<Laptop> laptopList) {

return laptopList.stream().filter(l -> l.getRam().equals("4GB")).collect(Collectors.toList());

}

}

public class LaptopScanner {

public static void main(String[] args) {

List<Laptop> laptops = LaptopFactory.manufactureInBulk();

AndCriteria searchCriteria = new AndCriteria(new Ram4GB(), new Macintosh());

List<Laptop> filteredLaptps = searchCriteria.meets(laptops);

filteredLaptps.stream().forEach(Laptop::prettyPrint);

}

}

class AndCriteria implements Criteria{

private Criteria[] criterias;

public AndCriteria(Criteria...criterias) {

this.criterias = criterias;

}

@Override

public List<Laptop> meets(List<Laptop> laptopList) {

List<Laptop> filteredLaptops = laptopList;

for(Criteria criteria : criterias) {

filteredLaptops = criteria.meets(filteredLaptops);

}

return filteredLaptops;

}

}

class LaptopFactory {

public static List<Laptop> manufactureInBulk() {

return Arrays.asList(

new Laptop(100, "250GB", "i5", "4GB", "MAC", "15inch"),

new Laptop(101, "250GB", "i5", "2GB", "MAC", "13inch"),

new Laptop(102, "150GB", "i3", "2GB", "MAC", "13inch"),

new Laptop(103, "500GB", "i7", "4GB", "MAC", "15inch"),

new Laptop(104, "250GB", "i5", "4GB", "UBUNTU", "13inch"),

new Laptop(105, "250GB", "i5", "2GB", "UBUNTU", "13inch"),

new Laptop(106, "500GB", "i7", "8GB", "UBUNTU", "15inch"),

new Laptop(107, "200GB", "i3", "2GB", "UBUNTU", "13inch"),

new Laptop(108, "200GB", "i3", "2GB", "WINDOWS", "15inch"),

new Laptop(109, "250GB", "i7", "4GB", "WINDOWS", "15inch"),

new Laptop(110, "500GB", "i7", "8GB", "WINDOWS", "13inch")

);

}

}