Filter

MAJD REZIK

SIMAAN SAADA

What is Filter Design Pattern?

Filter design pattern is self explanatory.
It belongs under Structural patterns.
What for?
we have data and we would like to filter it by some category or criteria and get a focused smaller data size which is more relevant or easier to work with.

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.

Uses of Filter Design Pattern

Examples:

- **You want to throw a party, inviting your family and friends.**
  Want to filter up those who drink alcohol and those underage.

- **You want to make a list of people, filter them according to their public status, married / divorced / single / male / female / etc.**

- **A company wants to do a statistical research about its employees:**
  1. **Employees with experience less than a year.**
  2. **Employees with experience between a year and 5 years.**
  3. **Employees with experience exceeding 5 years.**

Uses of Filter Design Pattern – Cont.

As you can see from the examples, the filter design pattern has many uses and is actually very easy to implement and use.

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 is also fairly easy to run over a list and filter out stuff using “if statement” but the design pattern makes it more tidy especially when you would like to filter out many lists under many criterias.

Person.java

```java
public class Person {

   private String name;
   private String gender;
   private String maritalStatus;

   public Person(String name, String gender, String maritalStatus){
      this.name = name;
      this.gender = gender;
      this.maritalStatus = maritalStatus;
   }

   public String getName() {
      return name;
   }
   public String getGender() {
      return gender;
   }
   public String getMaritalStatus() {
      return maritalStatus;
   }
}
```

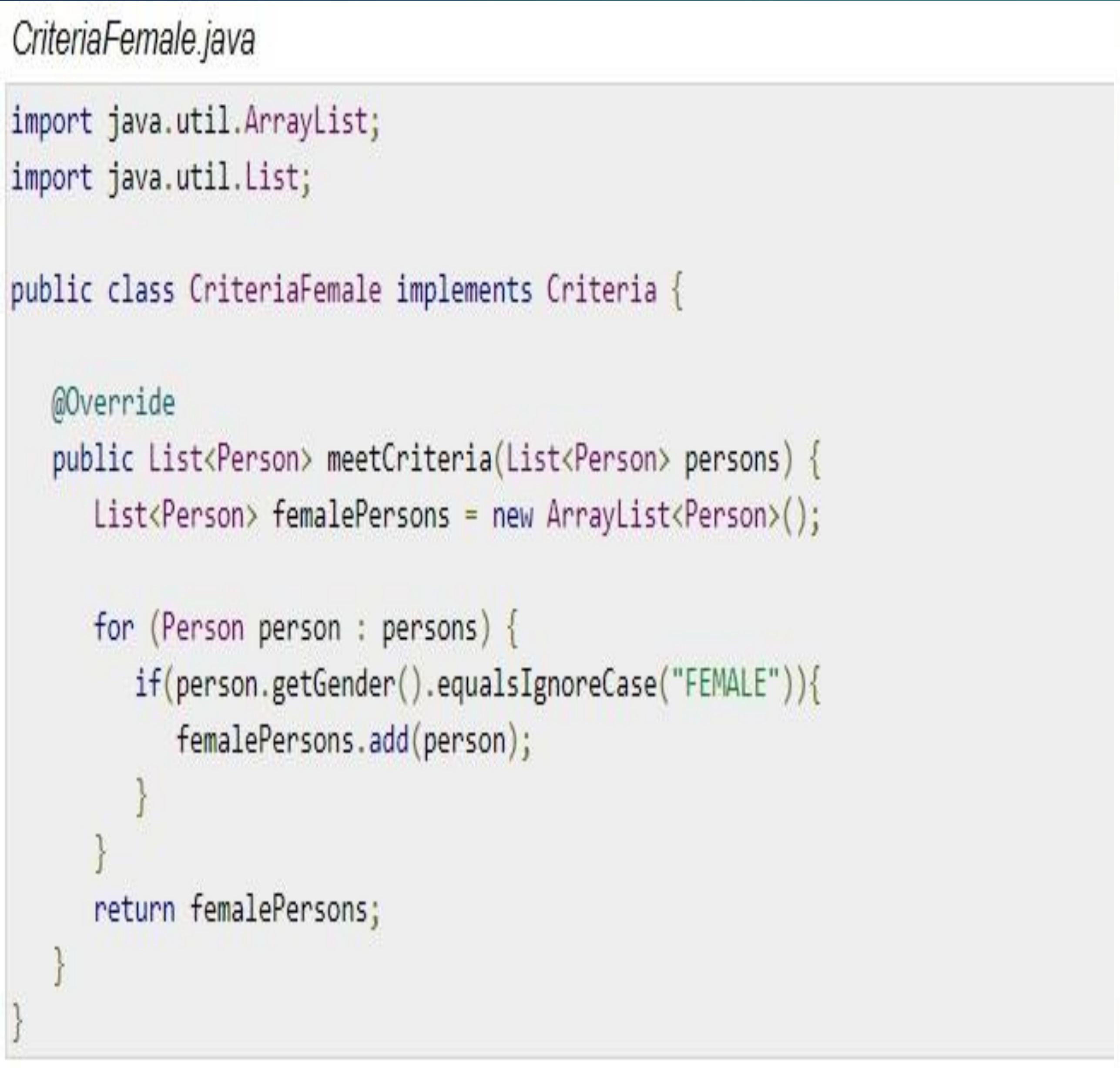
Criteria.java

```java
import java.util.List;

public interface Criteria {
   public List<Person> meetCriteria(List<Person> persons);
}
```

CriteriaMale.java

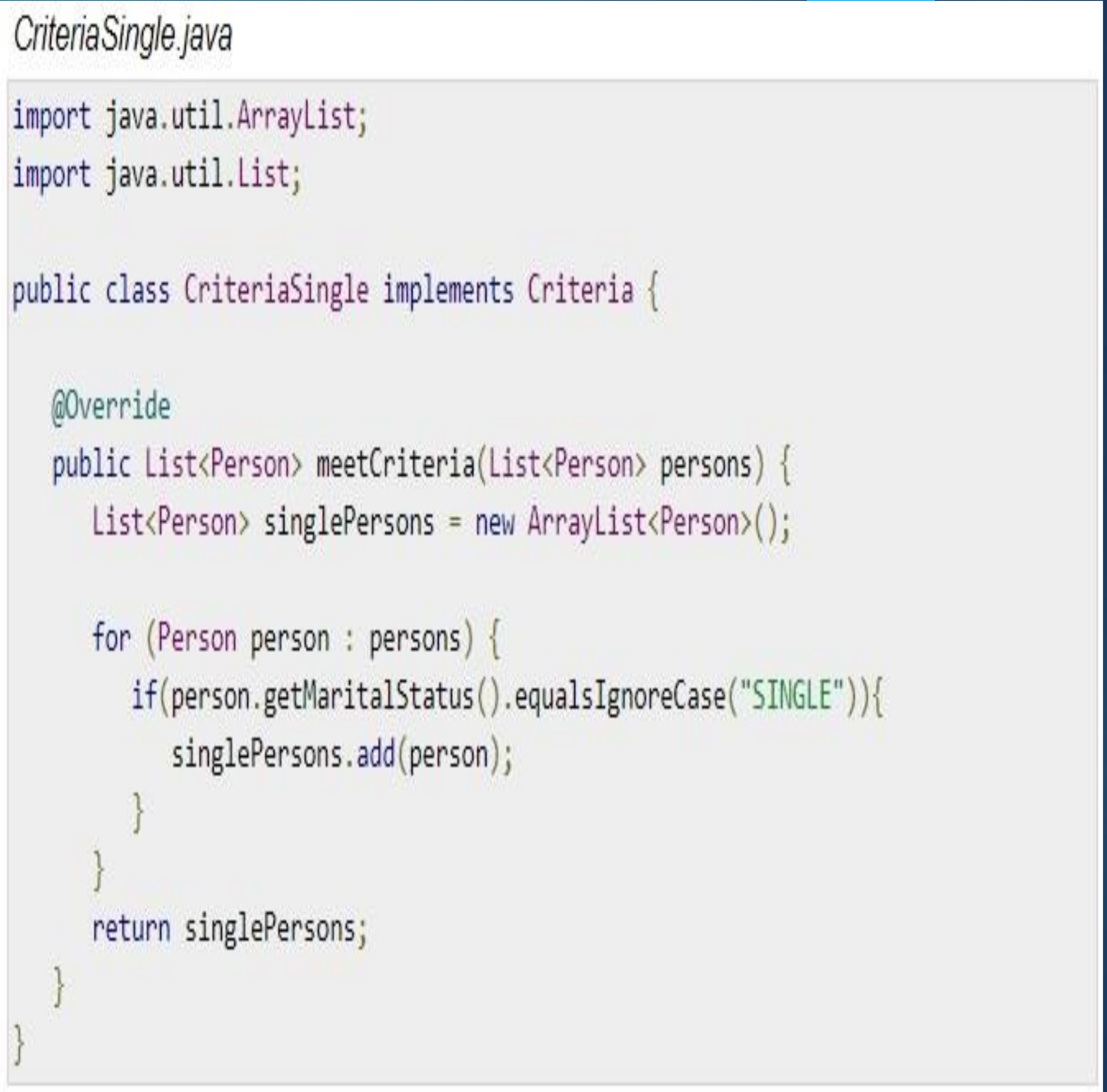
```java
import java.util.ArrayList;
import java.util.List;

public class CriteriaMale implements Criteria {

   @Override
   public List<Person> meetCriteria(List<Person> persons) {
      List<Person> malePersons = new ArrayList<Person>();

      for (Person person : persons) {
         if(person.getGender().equalsIgnoreCase("MALE")){
            malePersons.add(person);
         }
      }
      return malePersons;
   }
}
```

More criterias examples:



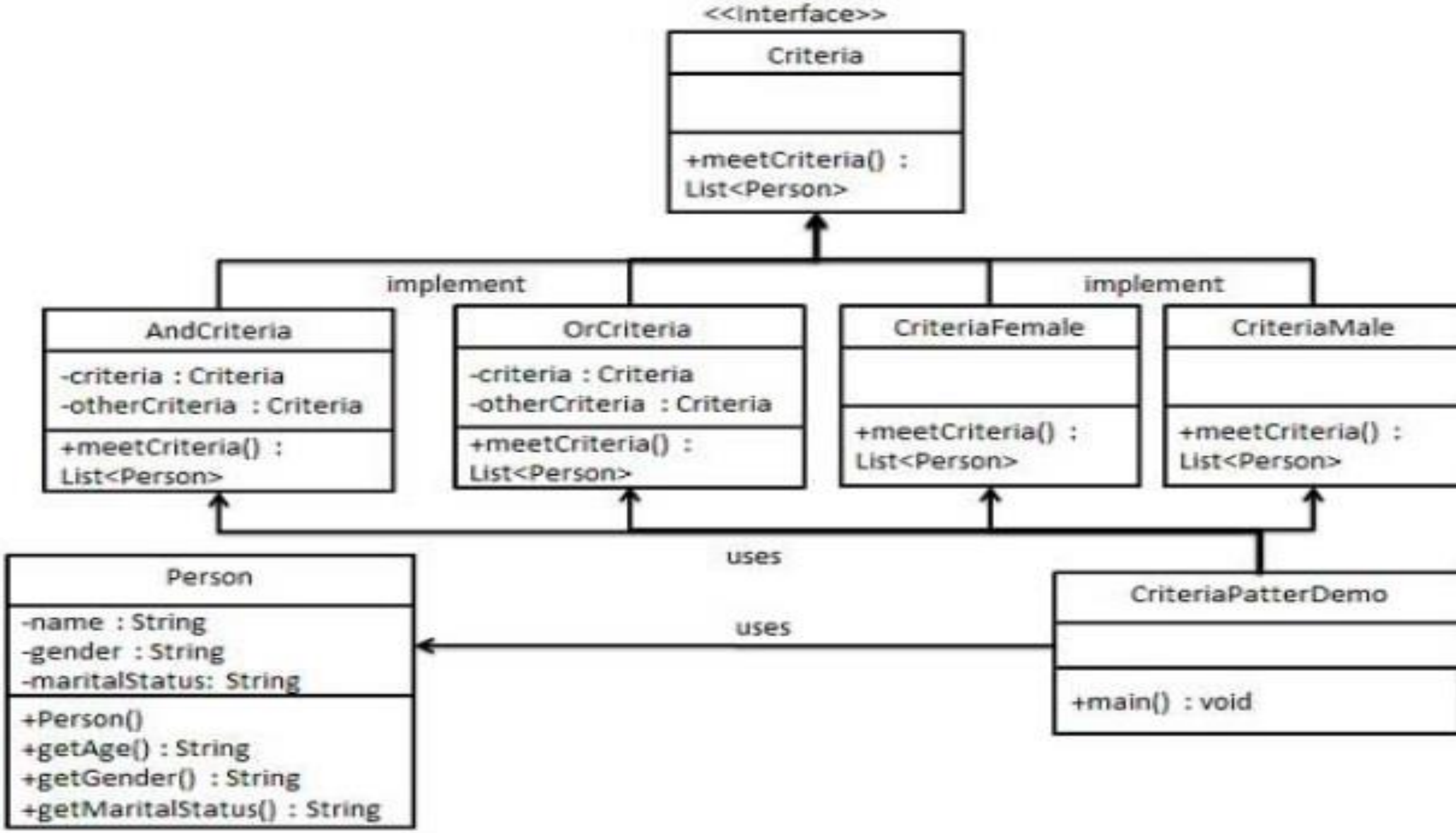
CriteriaFemale.java

```java
import java.util.ArrayList;
import java.util.List;

public class CriteriaFemale implements Criteria {

   @Override
   public List<Person> meetCriteria(List<Person> persons) {
      List<Person> femalePersons = new ArrayList<Person>();

      for (Person person : persons) {
         if(person.getGender().equalsIgnoreCase("FEMALE")){
            femalePersons.add(person);
         }
      }
      return femalePersons;
   }
}
```



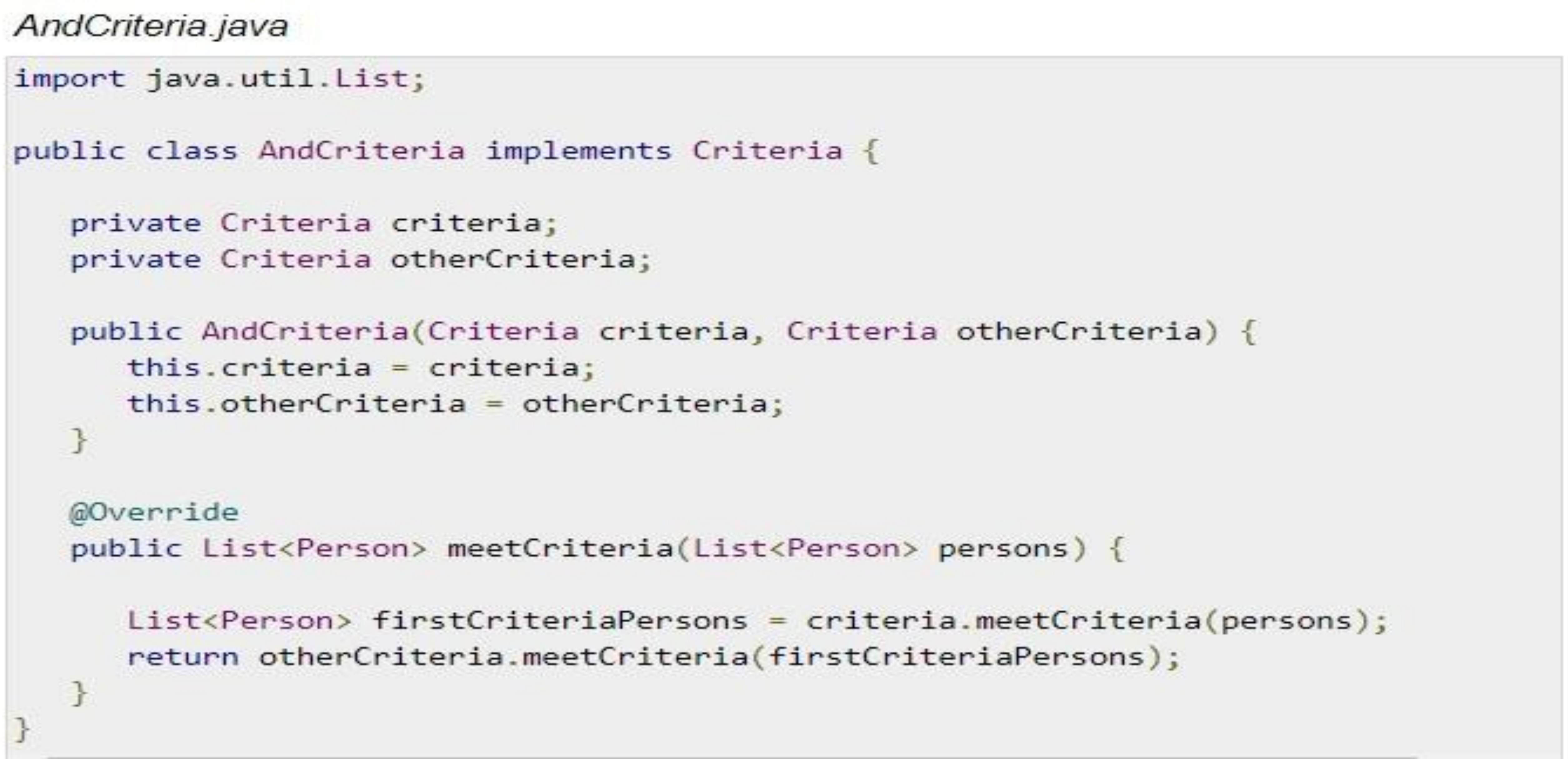
CriteriaSingle.java

```java
import java.util.ArrayList;
import java.util.List;

public class CriteriaSingle implements Criteria {

   @Override
   public List<Person> meetCriteria(List<Person> persons) {
      List<Person> singlePersons = new ArrayList<Person>();

      for (Person person : persons) {
         if(person.getMaritalStatus().equalsIgnoreCase("SINGLE")){
            singlePersons.add(person);
         }
      }
      return singlePersons;
   }
}
```



<<Interface>>
Criteria
+meetCriteria() : List<Person>
implement
implement
AndCriteria
-criteria : Criteria
-otherCriteria : Criteria
+meetCriteria() : List<Person>
OrCriteria
-criteria : Criteria
-otherCriteria : Criteria
+meetCriteria() : List<Person>
CriteriaFemale
+meetCriteria() : List<Person>
CriteriaMale
+meetCriteria() : List<Person>
uses
Person
-name : String
-gender : String
-maritalStatus: String
+Person()
+getAge() : String
+getGender() : String
+getMaritalStatus() : String
uses
CriteriaPatterDemo
+main() : void

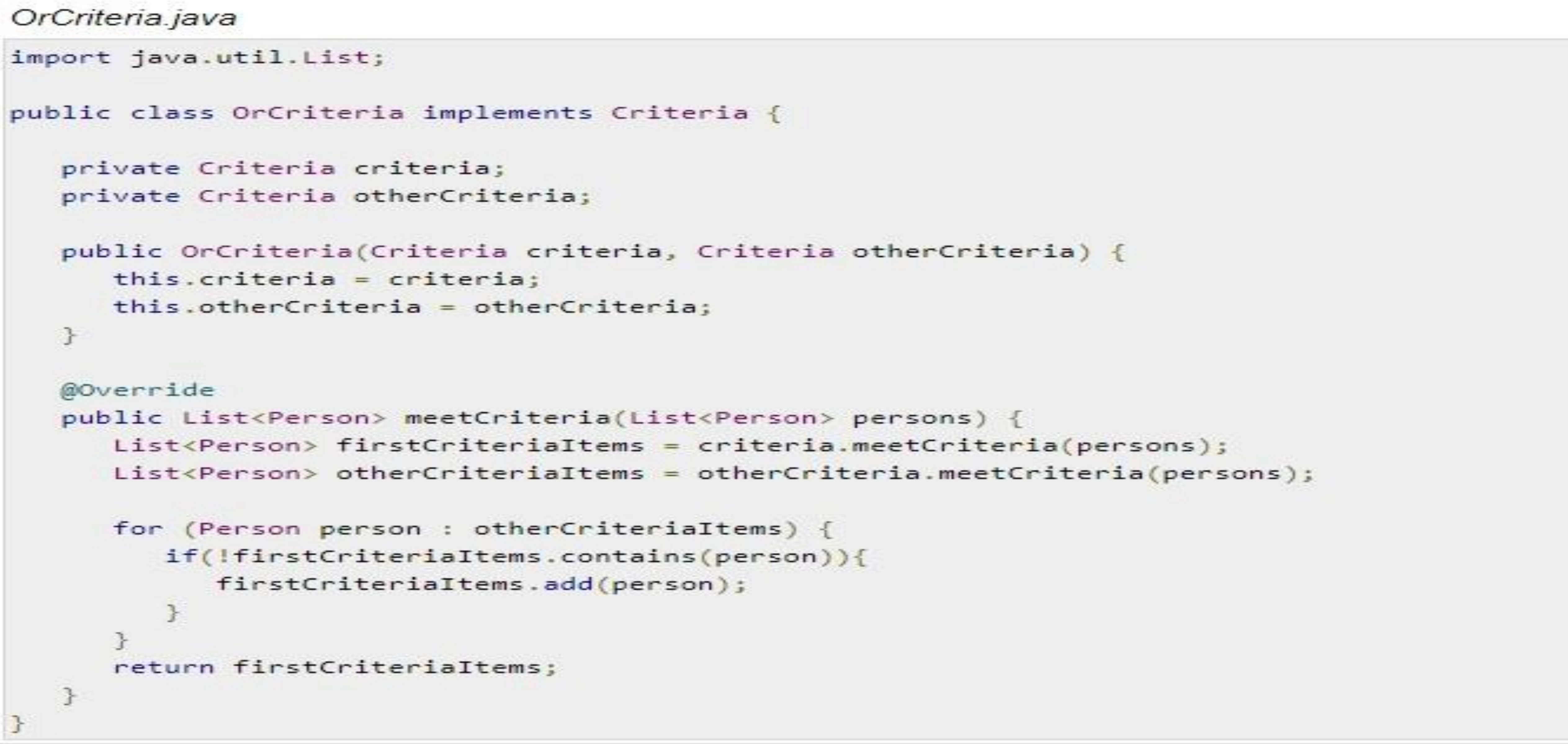
AND operation between criterias



AndCriteria.java

```java
import java.util.List;

public class AndCriteria implements Criteria {

   private Criteria criteria;
   private Criteria otherCriteria;

   public AndCriteria(Criteria criteria, Criteria otherCriteria) {
      this.criteria = criteria;
      this.otherCriteria = otherCriteria;
   }

   @Override
   public List<Person> meetCriteria(List<Person> persons) {

      List<Person> firstCriteriaPersons = criteria.meetCriteria(persons);
      return otherCriteria.meetCriteria(firstCriteriaPersons);
   }
}
```

Or operation between criteria's


OrCriteria.java

```java
import java.util.List;

public class OrCriteria implements Criteria {

   private Criteria criteria;
   private Criteria otherCriteria;

   public OrCriteria(Criteria criteria, Criteria otherCriteria) {
      this.criteria = criteria;
      this.otherCriteria = otherCriteria;
   }

   @Override
   public List<Person> meetCriteria(List<Person> persons) {
      List<Person> firstCriteriaItems = criteria.meetCriteria(persons);
      List<Person> otherCriteriaItems = otherCriteria.meetCriteria(persons);

      for (Person person : otherCriteriaItems) {
         if(!firstCriteriaItems.contains(person)){
            firstCriteriaItems.add(person);
         }
      }
      return firstCriteriaItems;
   }
}
```

Code files

Click here to display the files in GitHub

Exercise 1:

1. **Print out the students with final grade above 50.**
2. **Print out the male students with final grade above 50.**

Output 1:

```
students with final grade above 50:
Student : [ Name : Alex, Gender : male, ID : 1, Final Grade: 79.5 ]
Student : [ Name : Bob, Gender : male, ID : 2, Final Grade: 60.0 ]
Student : [ Name : Alice, Gender : female, ID : 3, Final Grade: 55.0 ]
Student : [ Name : Mark, Gender : male, ID : 4, Final Grade: 100.0 ]
Student : [ Name : Steve, Gender : male, ID : 5, Final Grade: 95.0 ]
Student : [ Name : Noemi, Gender : female, ID : 8, Final Grade: 85.0 ]
Student : [ Name : Laila, Gender : female, ID : 9, Final Grade: 61.0 ]
```

Output 2:

```
male students with final grade above 50:
Student : [ Name : Alex, Gender : male, ID : 1, Final Grade: 79.5 ]
Student : [ Name : Bob, Gender : male, ID : 2, Final Grade: 60.0 ]
Student : [ Name : Mark, Gender : male, ID : 4, Final Grade: 100.0 ]
Student : [ Name : Steve, Gender : male, ID : 5, Final Grade: 95.0 ]
```

Exercise 2: (2 parts)

Part 1:

- **Calculate and print the average of ALL the students with final grade above 60. And print those students.**

Output :

```
Average of students with final grade above 60:
84.1

Student : [ Name : Alex, Gender : male, ID : 1, Final Grade: 79.5 ]
Student : [ Name : Mark, Gender : male, ID : 4, Final Grade: 100.0 ]
Student : [ Name : Steve, Gender : male, ID : 5, Final Grade: 95.0 ]
Student : [ Name : Noemi, Gender : female, ID : 8, Final Grade: 85.0 ]
Student : [ Name : Laila, Gender : female, ID : 9, Final Grade: 61.0 ]
```

Part 2:

- **Print out the <u>male</u> students with final grade between 60 and 100 <u>OR</u> <u>female</u> students with final grade between 20 and 50.**

Output :

```
Student : [ Name : Diana, Gender : female, ID : 6, Final Grade: 29.0 ]
Student : [ Name : Elina, Gender : female, ID : 10, Final Grade: 49.0 ]
Student : [ Name : Alex, Gender : male, ID : 1, Final Grade: 79.5 ]
Student : [ Name : Mark, Gender : male, ID : 4, Final Grade: 100.0 ]
Student : [ Name : Steve, Gender : male, ID : 5, Final Grade: 95.0 ]
```

Thank you.