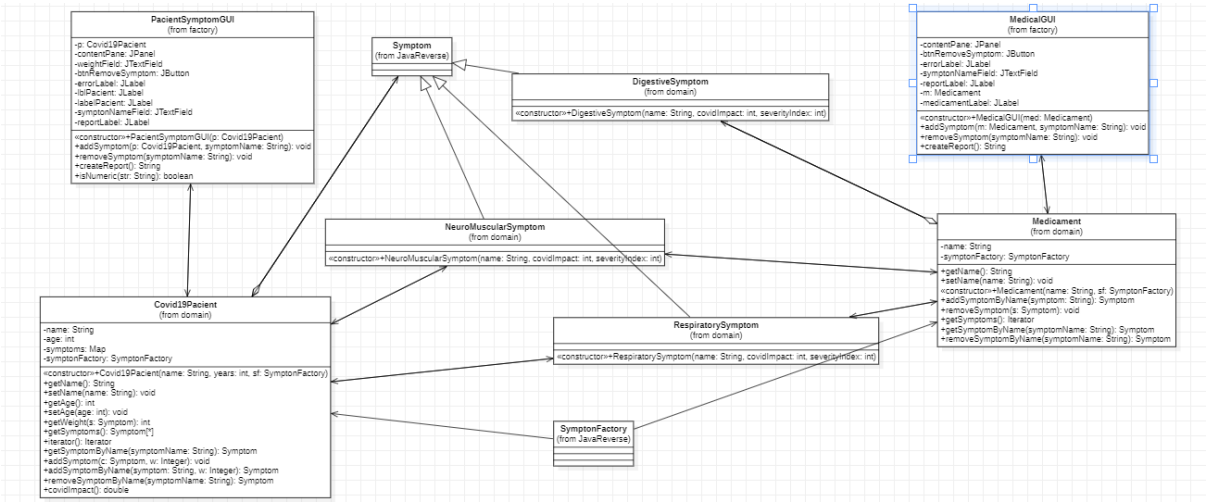
Simple Factory

Aldaketa hau egin baino lehen, covid19Patient-en eta Medicament-en createSympton metodoa geneukan. Atal hau egiteko SymtomFactory klasea sortu dugu. Modu honetan createSymptom metodo bakarra egongo da eta ez bi. Horrela symptom berri bat sortzen badugu, biek eukiko dute datu berri hau.



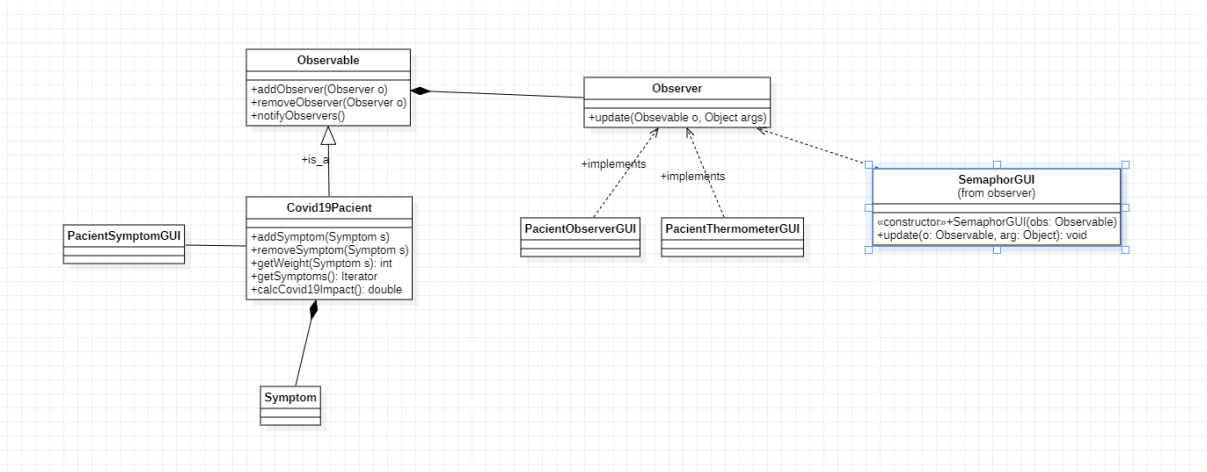
Observer Patroia

Aldaketa hauek egiteko laborategiak esaten digun pausuak jarraitu ditugu. addSymptomByName eta removeSymptomByName metodoetan hurrengo instrukzioak gehitu ditugu: setChanged(); notifyObservers();

Gero PacientObserverGUI-ean Observer inplementatu dugu, metodo eraikitzaileari "obs" izeneko Observable parametro bat gehitu, metodoaren eraikitzailearen bukaeran, obs.addObserver(this) gehitu, eta update metodoa eguneratu.

PacientSymptomGUI klasean bi listener gehitu ditugu botoientzako. Azkenik programa nagusia sortu dugu.

PacientObserverGUI-ean egindakoa Observer-ekin errepikatu egin dugu PacientThermometerGUI eta SemaphorGUI.



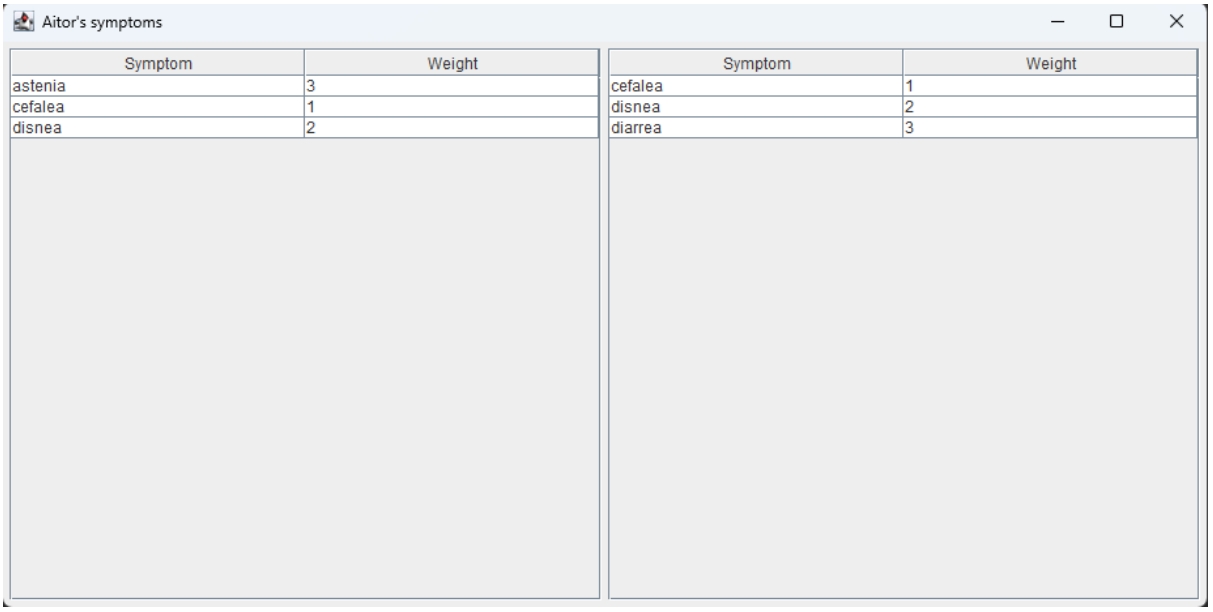
Adapter Patroia

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

import javax.swing.table.AbstractTableModel;

import domain.Covid19Pacient;
import domain.Symptom;



public class Covid19PacientModelAdapter extends AbstractTableModel {

    private final List<Symptom> symptoms;
    private Covid19Pacient pacient;
    private String[] colNames = new String[] {"Symptom", "Weight"};


    public Covid19PacientModelAdapter(Covid19Pacient p) {
        //copy the HashMap data to a sequential data structure
        symptoms=new ArrayList<Symptom>(p.getSymptoms());
        this.pacient=p;

    }

    @Override
    public String getColumnName(int col) {
        return colNames[col];
    }
    @Override
    public int getColumnCount() {
        return 2;
    }

    @Override
    public int getRowCount() {
        return symptoms.size();
    }

    @Override
    public Object getValueAt(int rowIndex, int columnIndex) {
       switch(columnIndex) {
            case 0: return ((Object) symptoms.get(rowIndex).getName());
            case 1: return ((Object) pacient.getWeight(symptoms.get(rowIndex)));


    }
       return null;
   }
}
```

Covid19PacientModelAdapter klasean AbstratcTableModel-eko metodoak inplmentatu ditugu.

Hori egin eta gero, Main2 klasea egin dugu taulak probatzeko:

```java
package jframeAdapter;

import javax.swing.JFrame;
import javax.swing.JPanel;
import javax.swing.JScrollPane;
import javax.swing.JTable;

import domain.Covid19Pacient;
import domain.Symptom;
import factory.SymptonFactory;

public class Main2 {

    public static void main(String[] args) {

        SymptonFactory sf = new SymptonFactory();
        Covid19Pacient p = new Covid19Pacient("Aitor", 29, sf);
        p.addSymptom(new Symptom("disnea", 10, 10), 2);
        p.addSymptom(new Symptom("cefalea", 10, 10), 1);
        p.addSymptom(new Symptom("astenia", 10, 10), 3);

        Covid19Pacient p2 = new Covid19Pacient("Xabi", 29, sf);
        p2.addSymptom(new Symptom("disnea", 10, 10), 2);
        p2.addSymptom(new Symptom("cefalea", 10, 10), 1);
        p2.addSymptom(new Symptom("diarrea", 10, 10), 3);

        Covid19PacientModelAdapter pacientModelAdapter = new Covid19PacientModelAdapter(p);
        Covid19PacientModelAdapter pacientModelAdapter2 = new Covid19PacientModelAdapter(p2);

        JTable table = new JTable(pacientModelAdapter);
        JTable table2 = new JTable(pacientModelAdapter2);

        JPanel panel = new JPanel();
        panel.add(new JScrollPane(table));
        panel.add(new JScrollPane(table2));

        JFrame j = new JFrame();
        j.add(panel);

        j.setTitle(p.getName() + "'s symptoms");
        j.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        j.pack();
        j.setVisible(true);
    }
}
```



Adapter Iterator Patroia

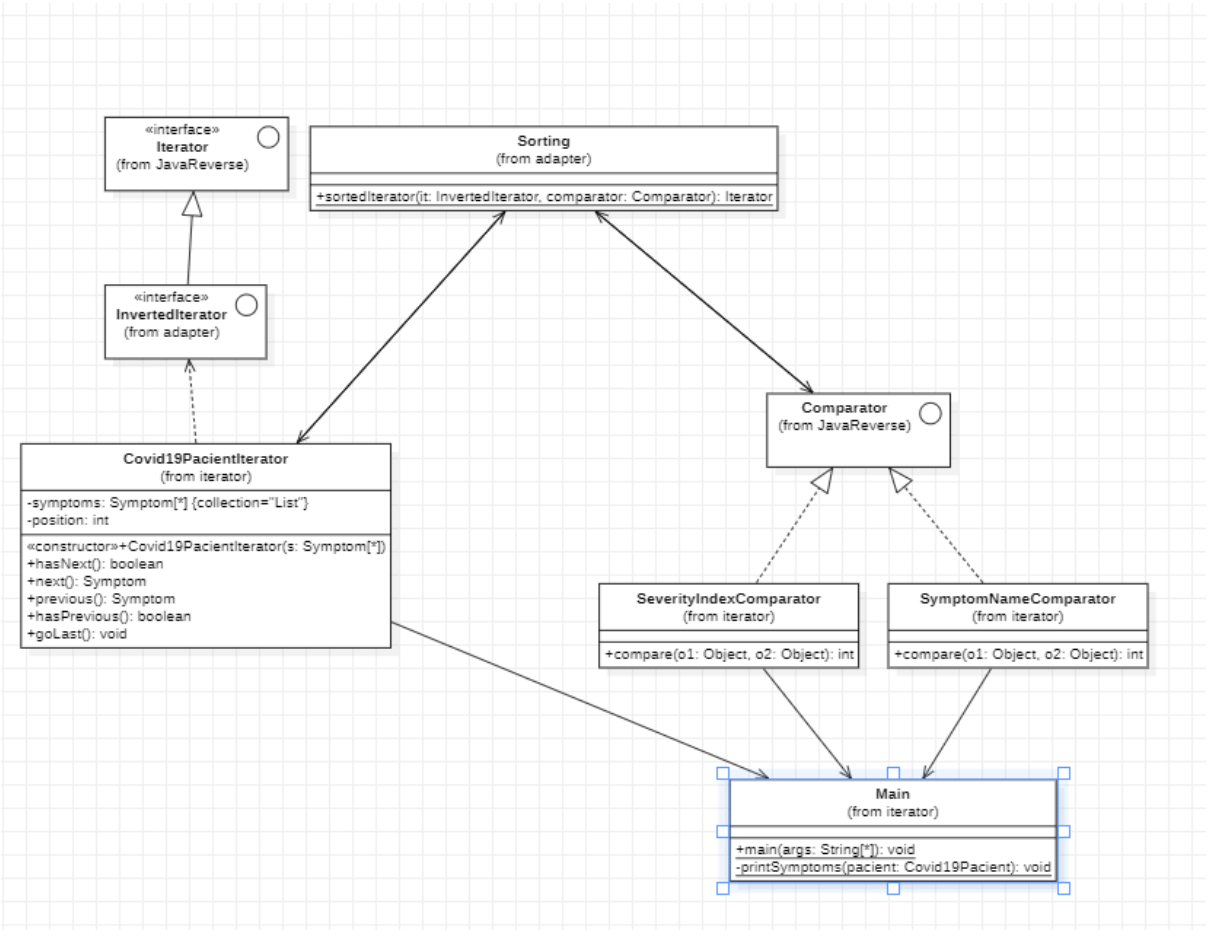
Iterator bat sortu dugu alderantziz ordenatzeko kapaza dena. Hortaz aparte bi klase sortu ditugu Comparator inplementatzen duena. Bat izena erabiliz eta bestea zorroztasun zenbakiarekin.

```java
package iterator;

import java.util.Comparator;

import domain.Symptom;

public class SeverityIndexComparator implements Comparator {
        @Override
        public int compare(Object o1, Object o2) {
                Symptom s1 = (Symptom) o1;
                Symptom s2 = (Symptom) o2;
                 return Integer.compare(s1.getSeverityIndex(), s2.getSeverityIndex());
        }
}
```

```java
package iterator;

import java.util.Comparator;

import domain.Symptom;

public class SymptomNameComparator implements Comparator {

        @Override
        public int compare(Object o1, Object o2) {
                Symptom s1 = (Symptom) o1;
                Symptom s2 = (Symptom) o2;
                return s1.getName().compareTo(s2.getName());
        }
}
```

```java
package iterator;

import java.util.ArrayList;
import java.util.Collections;
import java.util.Iterator;
import java.util.List;
import java.util.Set;

import adapter.InvertedIterator;
import domain.Symptom;

public class Covid19PacientIterator implements InvertedIterator {
    private List<Symptom> symptoms;
    private int position;

    public Covid19PacientIterator(ArrayList<Symptom> s) {
        this.symptoms = s;
        //Collections.reverse(symptoms); // Invert the list
        this.position = 0;
    }

    @Override
    public boolean hasNext() {
        return position < symptoms.size();
    }

    @Override
    public Symptom next() {
        return symptoms.get(position++);
    }

    @Override
    public Symptom previous() {
        if (hasPrevious()) {
            return symptoms.get(--position);
        }
        return null;
    }

    @Override
    public boolean hasPrevious() {
        return position > 0;
    }

    @Override
    public void goLast() {
        position = symptoms.size() - 1;
    }
}
```

```java
package iterator;

import java.util.ArrayList;
import java.util.Iterator;
import java.util.List;

import javax.swing.JFrame;
import javax.swing.JScrollPane;
import javax.swing.JTable;

import adapter.Sorting;
import domain.Covid19Pacient;
import domain.Symptom;
import factory.SymptonFactory;

        public class Main {

                public static void main(String[] args) {

                        SymptonFactory sf = new SymptonFactory();
                        Covid19Pacient p=new Covid19Pacient("Ane", 29, sf);
                        p.addSymptom(new Symptom("a", 10, 5), 1);
                        p.addSymptom(new Symptom("b", 10, 10), 2);
                        p.addSymptom(new Symptom("c", 10, 6), 3);
                        p.addSymptom(new Symptom("d", 10, 2), 4);
                        p.addSymptom(new Symptom("e", 10, 4), 5);


                         ArrayList<Symptom> pp =  new ArrayList<>(p.getSymptoms());
                     Sorting.sortedIterator(new Covid19PacientIterator(pp), new SymptomNameComparator());

                        // Print symptoms
                        printSymptoms(p);

                        // Sort by severityIndex
                        Sorting.sortedIterator(new Covid19PacientIterator(pp), new SeverityIndexComparator());

                        // Print symptoms
                        printSymptoms(p);
                    }

                    private static void printSymptoms(Covid19Pacient pacient) {
                        System.out.println("Symptoms for " + pacient.getName() + ":");
                        Iterator<Symptom> iterator = pacient.getSymptoms().iterator();
                        while (iterator.hasNext()) {
                            Symptom symptom = iterator.next();
                            System.out.println(symptom.getName() + " - Severity: " + symptom.getSeverityIndex());
                        }
                        System.out.println();
                    }
                }
```


«interface»
Iterator
(from JavaReverse)
Sorting
(from adapter)
+sortedIterator(it: InvertedIterator, comparator: Comparator): Iterator
«interface»
InvertedIterator
(from adapter)
Comparator
(from JavaReverse)
Covid19PacientIterator
(from iterator)
-symptoms: Symptom[*] {collection="List"}
-position: int
«constructor»+Covid19PacientIterator(s: Symptom[*])
+hasNext(): boolean
+next(): Symptom
+previous(): Symptom
+hasPrevious(): boolean
+goLast(): void
SeverityIndexComparator
(from iterator)
+compare(o1: Object, o2: Object): int
SymptomNameComparator
(from iterator)
+compare(o1: Object, o2: Object): int
Main
(from iterator)
+main(args: String[*]): void
-printSymptoms(pacient: Covid19Pacient): void