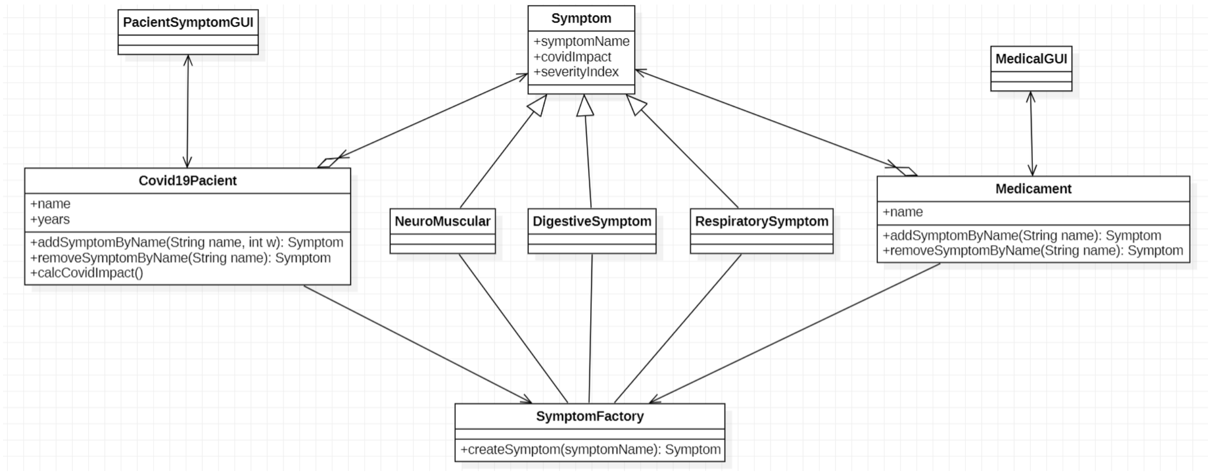
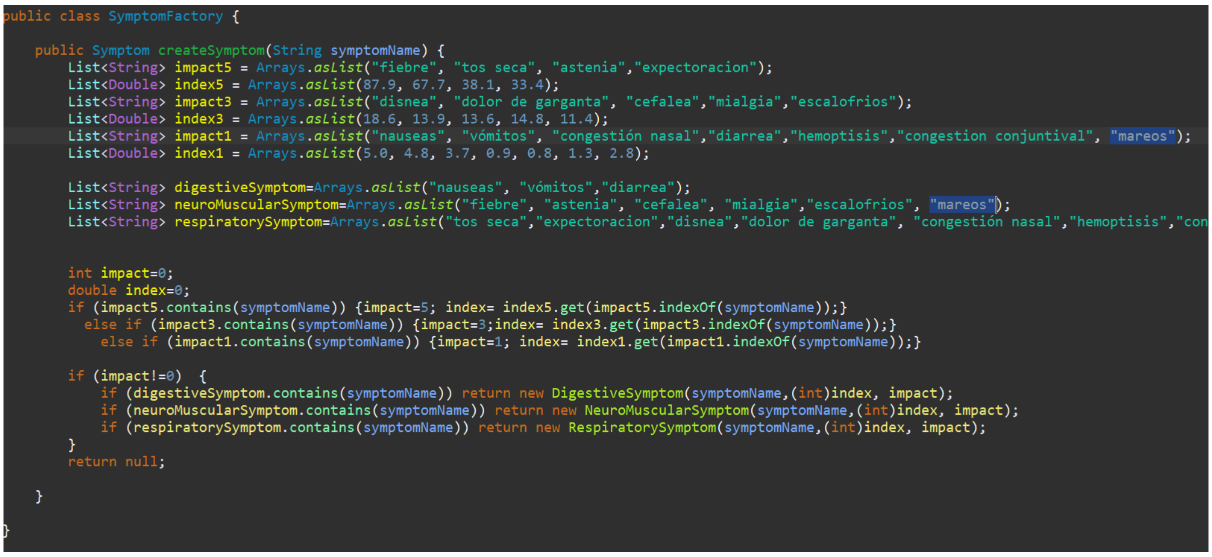
Egileak: Lander Soriano, Damian Vela, Ander Berruezo
Github esteka: https://github.com/anderberru/labpatterns

Simple Factory

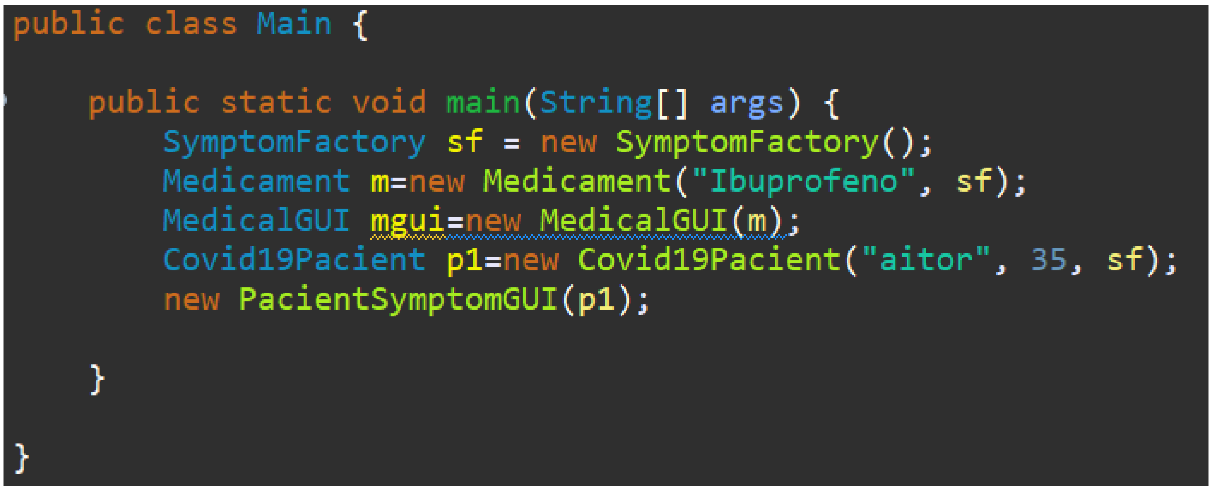

PacientSymptomGUI
Symptom
+symptomName
+covidImpact
+severityIndex
MedicalGUI
Covid19Pacient
+name
+years
+addSymptomByName(String name, int w): Symptom
+removeSymptomByName(String name): Symptom
+calcCovidImpact()
NeuroMuscular
DigestiveSymptom
RespiratorySymptom
Medicament
+name
+addSymptomByName(String name): Symptom
+removeSymptomByName(String name): Symptom
SymptomFactory
+createSymptom(symptomName): Symptom

SymptomFactory izeneko klase berri bat sortu dugu, sintomak sortzeaz arduratzen dena. Klase honen metodo bakarra createSymptom() da, Covid19Pacient eta Medicament klaseetatik mugitu duguna. Horrela, sintoma berri bat sortu nahi badugu edo sortzeko metodoan aldaketak egin nahi baditugu, bakarrik SymptomFactory klasea aldatu behar dugu, ez sintomak sortu behar dituzten klase guztiak. Horrela aldatu dugu metodoa “mareos” sintoma gehitzeko:



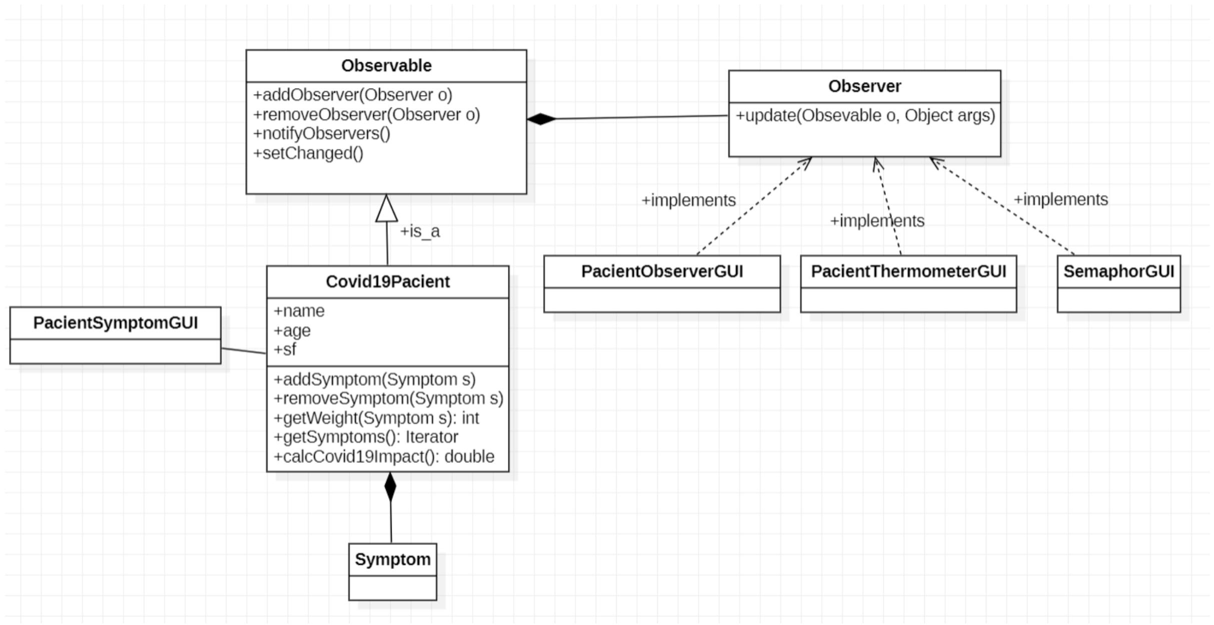
```
public class SymptomFactory {

    public Symptom createSymptom(String symptomName) {
        List<String> impact5 = Arrays.asList("fiebre", "tos seca", "astenia","expectoracion");
        List<Double> index5 = Arrays.asList(87.9, 67.7, 38.1, 33.4);
        List<String> impact3 = Arrays.asList("disnea", "dolor de garganta", "cefalea","mialgia","escalofrios");
        List<Double> index3 = Arrays.asList(18.6, 13.9, 13.6, 14.8, 11.4);
        List<String> impact1 = Arrays.asList("nauseas", "vómitos", "congestión nasal","diarrea","hemoptisis","congestion conjuntival", "mareos");
        List<Double> index1 = Arrays.asList(5.0, 4.8, 3.7, 0.9, 0.8, 1.3, 2.8);

        List<String> digestiveSymptom=Arrays.asList("nauseas", "vómitos","diarrea");
        List<String> neuroMuscularSymptom=Arrays.asList("fiebre", "astenia", "cefalea", "mialgia","escalofrios", "mareos");
        List<String> respiratorySymptom=Arrays.asList("tos seca","expectoracion","disnea","dolor de garganta", "congestión nasal","hemoptisis","con


        int impact=0;
        double index=0;
        if (impact5.contains(symptomName)) {impact=5; index= index5.get(impact5.indexOf(symptomName));}
          else if (impact3.contains(symptomName)) {impact=3;index= index3.get(impact3.indexOf(symptomName));}
            else if (impact1.contains(symptomName)) {impact=1; index= index1.get(impact1.indexOf(symptomName));}

        if (impact!=0)  {
            if (digestiveSymptom.contains(symptomName)) return new DigestiveSymptom(symptomName,(int)index, impact);
            if (neuroMuscularSymptom.contains(symptomName)) return new NeuroMuscularSymptom(symptomName,(int)index, impact);
            if (respiratorySymptom.contains(symptomName)) return new RespiratorySymptom(symptomName,(int)index, impact);
        }
        return null;

    }

}
```

Covid19Pacient eta Medicament klaseak SymptomFactory motako sf atributu bat dute createSymptom() metodoari deitzeko. Horrela geratu da main klasea:



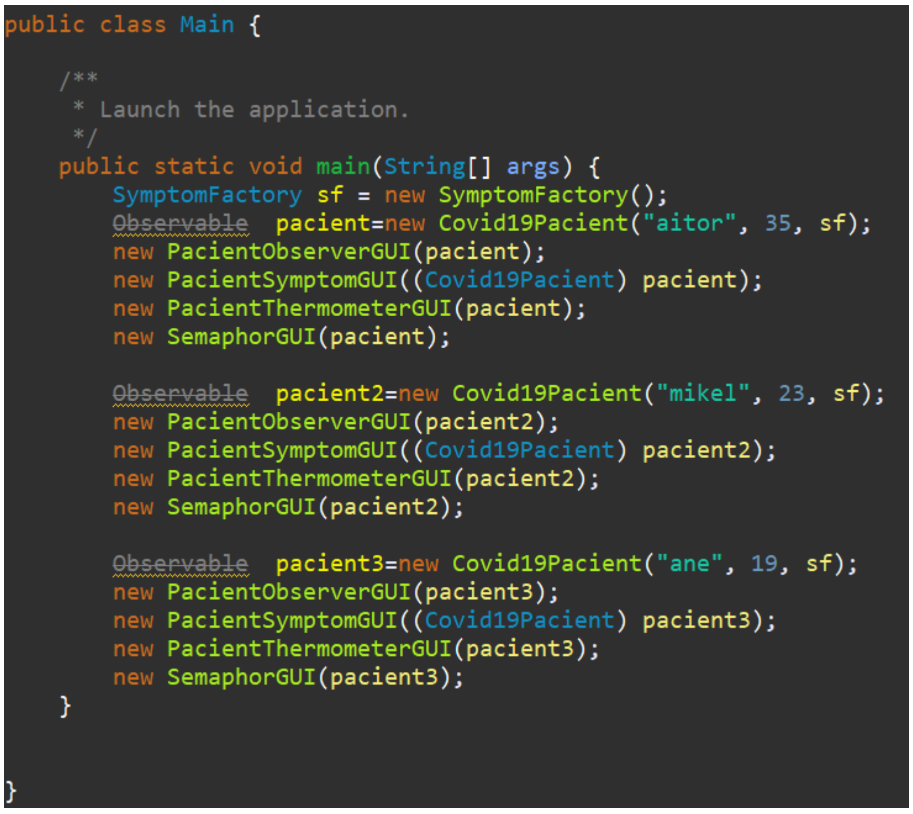
```java
public class Main {

    public static void main(String[] args) {
        SymptomFactory sf = new SymptomFactory();
        Medicament m=new Medicament("Ibuprofeno", sf);
        MedicalGUI mgui=new MedicalGUI(m);
        Covid19Pacient p1=new Covid19Pacient("aitor", 35, sf);
        new PacientSymptomGUI(p1);

    }

}
```

Observer Patroia



Hiru GUI klaseak Observable motako aldagaiak dauzkate bere eraikitzailetan, eta Observer klaseko update() metodoa implementatzen dute, sintomak gehitzean edo ezabatzean pantailan agertzen dena zuzenean aldatzeko. Covid19Pacient Observable interfazea implementatu behar du.

Horrela geratu da main klasea, hiru pazienteen sintomak, termometroa eta semaforoa agertzeko:

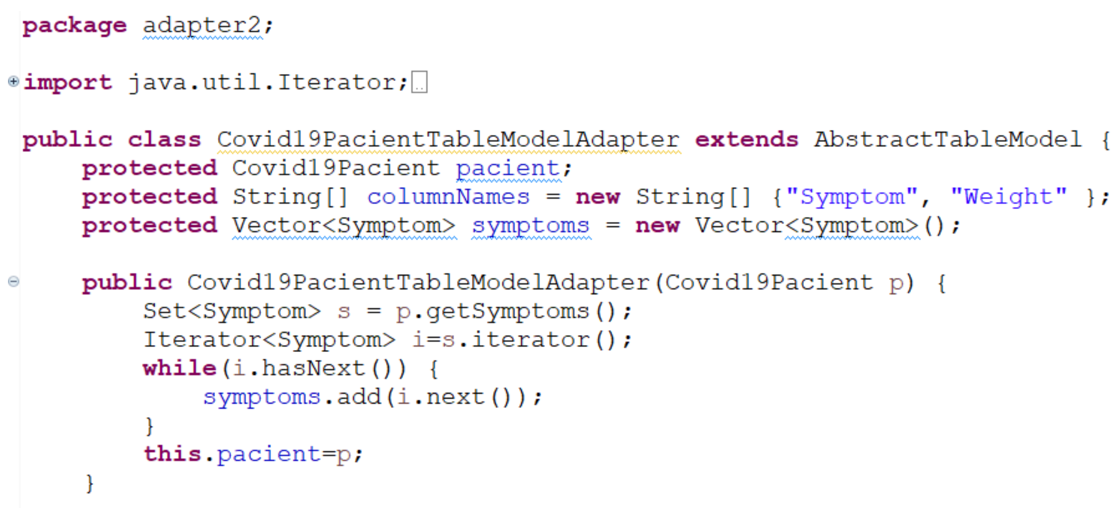


```java
public class Main {

    /**
     * Launch the application.
     */
    public static void main(String[] args) {
        SymptomFactory sf = new SymptomFactory();
        Observable  pacient=new Covid19Pacient("aitor", 35, sf);
        new PacientObserverGUI(pacient);
        new PacientSymptomGUI((Covid19Pacient) pacient);
        new PacientThermometerGUI(pacient);
        new SemaphorGUI(pacient);

        Observable  pacient2=new Covid19Pacient("mikel", 23, sf);
        new PacientObserverGUI(pacient2);
        new PacientSymptomGUI((Covid19Pacient) pacient2);
        new PacientThermometerGUI(pacient2);
        new SemaphorGUI(pacient2);

        Observable  pacient3=new Covid19Pacient("ane", 19, sf);
        new PacientObserverGUI(pacient3);
        new PacientSymptomGUI((Covid19Pacient) pacient3);
        new PacientThermometerGUI(pacient3);
        new SemaphorGUI(pacient3);
    }

}
```

<u>Adapter Patroia</u>

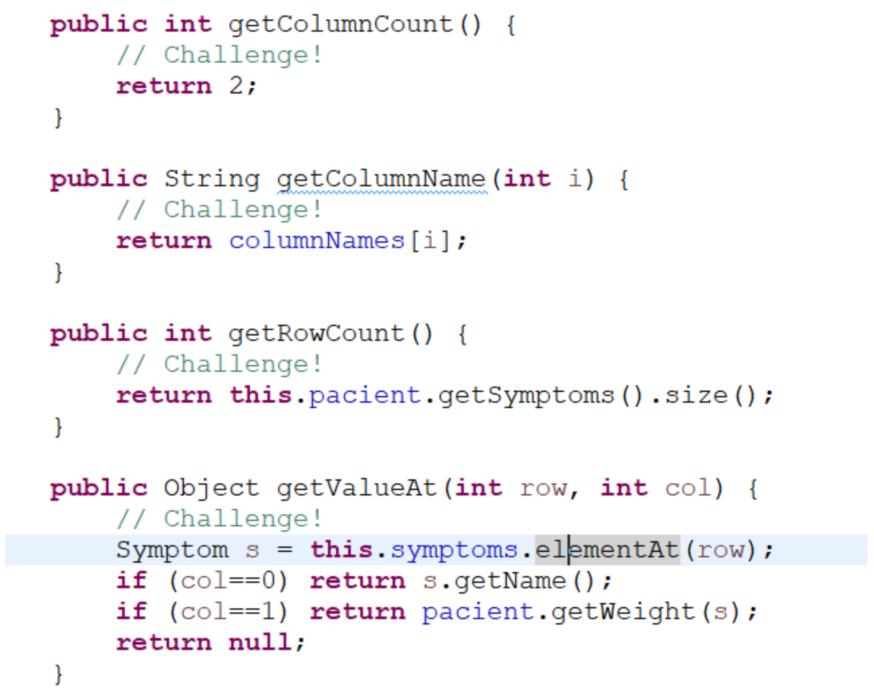
Paziente baten sintomak, taula batean aurkezteko adaptadore bat jarri behar izan diogu Covid19Pacient objetu bategatik JTable bat lortzeko. Horretarako erdian pauso bat jarri dugu non Covid19Pacient objetu batetik Covid19PacientTableModelAdapter bat lortzen dugu eta horrekin JTable bat.

Lehenengo eraikitzailea Covid19Pacient objetutik, bere sintoma multzoa lortzen dugu eta symptoms zerrendara gehitzen ditugu. Gainera, gordetzen dugu Covid19Pacient objetua pacient atributuan.



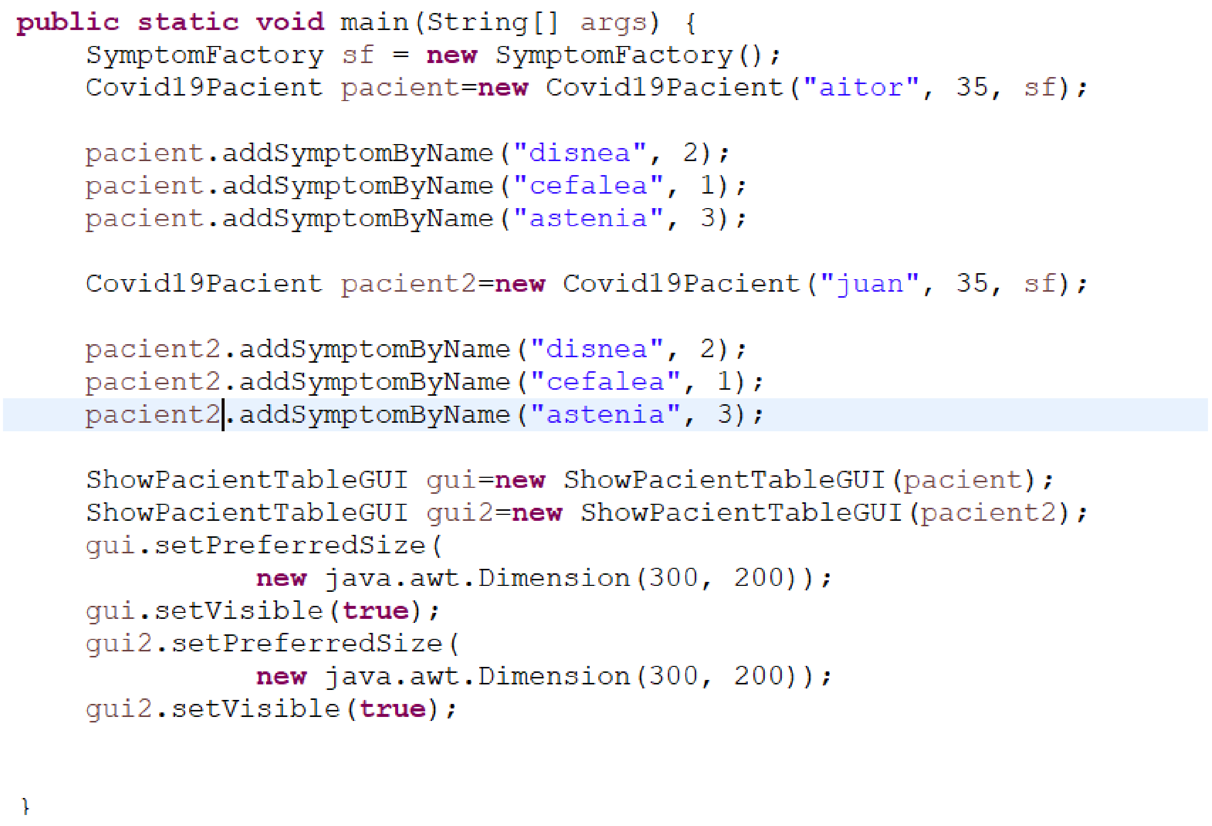
```java
package adapter2;

import java.util.Iterator;

public class Covid19PacientTableModelAdapter extends AbstractTableModel {
    protected Covid19Pacient pacient;
    protected String[] columnNames = new String[] {"Symptom", "Weight" };
    protected Vector<Symptom> symptoms = new Vector<Symptom>();

    public Covid19PacientTableModelAdapter(Covid19Pacient p) {
        Set<Symptom> s = p.getSymptoms();
        Iterator<Symptom> i=s.iterator();
        while(i.hasNext()) {
            symptoms.add(i.next());
        }
        this.pacient=p;
    }
```

Ondoren taula nolakoa izango den zehazteko TableModel erabiltzen dituen metodo hauek egin behar dira taula nolakoa nahi dugun adieraziz.



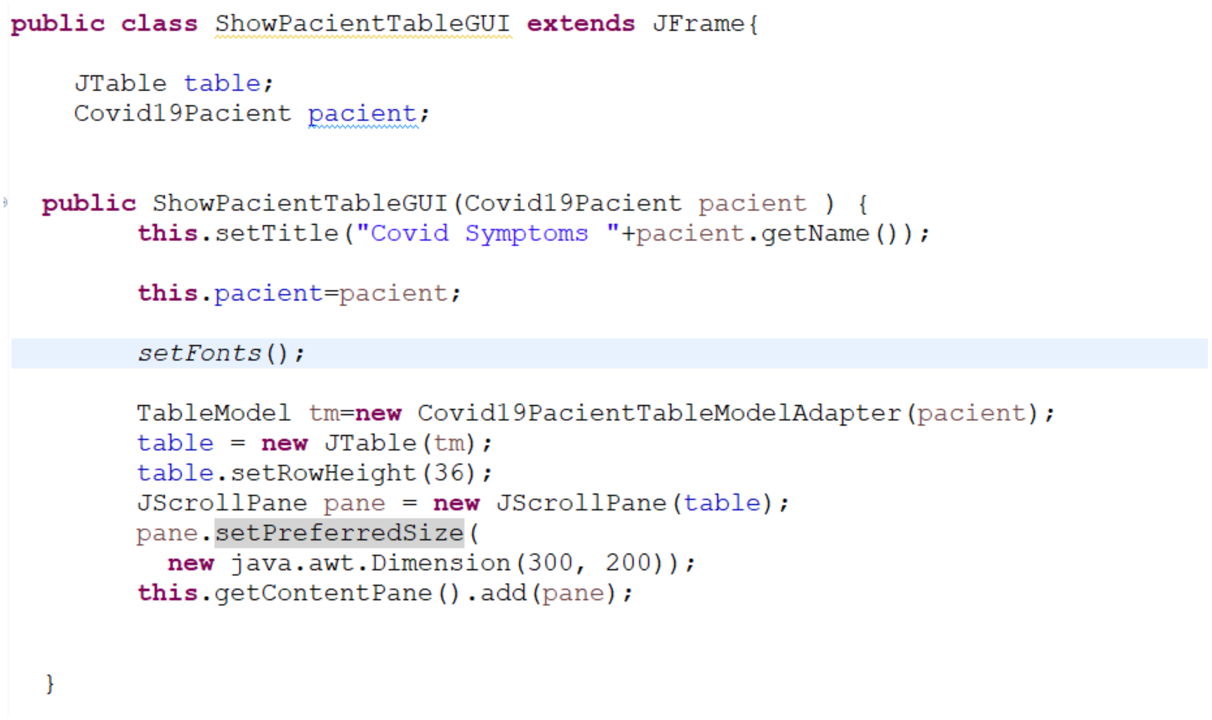
```java
public int getColumnCount() {
    // Challenge!
    return 2;
}

public String getColumnName(int i) {
    // Challenge!
    return columnNames[i];
}

public int getRowCount() {
    // Challenge!
    return this.pacient.getSymptoms().size();
}

public Object getValueAt(int row, int col) {
    // Challenge!
    Symptom s = this.symptoms.elementAt(row);
    if (col==0) return s.getName();
    if (col==1) return pacient.getWeight(s);
    return null;
}
```

Hurrengoa main metodoan 2 paziente 2 interfazeetan sortzen ditugu.



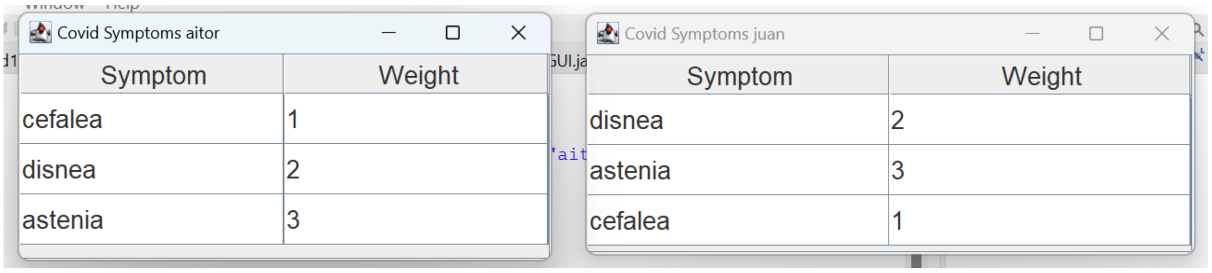
```
public static void main(String[] args) {
    SymptomFactory sf = new SymptomFactory();
    Covid19Pacient pacient=new Covid19Pacient("aitor", 35, sf);

    pacient.addSymptomByName("disnea", 2);
    pacient.addSymptomByName("cefalea", 1);
    pacient.addSymptomByName("astenia", 3);

    Covid19Pacient pacient2=new Covid19Pacient("juan", 35, sf);

    pacient2.addSymptomByName("disnea", 2);
    pacient2.addSymptomByName("cefalea", 1);
    pacient2.addSymptomByName("astenia", 3);

    ShowPacientTableGUI gui=new ShowPacientTableGUI(pacient);
    ShowPacientTableGUI gui2=new ShowPacientTableGUI(pacient2);
    gui.setPreferredSize(
              new java.awt.Dimension(300, 200));
    gui.setVisible(true);
    gui2.setPreferredSize(
              new java.awt.Dimension(300, 200));
    gui2.setVisible(true);


}
```

Azkenik, hemen ikusten da nola sortzen dugun gure Taulak interfazeetan.



```
public class ShowPacientTableGUI extends JFrame{

    JTable table;
    Covid19Pacient pacient;


  public ShowPacientTableGUI(Covid19Pacient pacient ) {
        this.setTitle("Covid Symptoms "+pacient.getName());

        this.pacient=pacient;

        setFonts();

        TableModel tm=new Covid19PacientTableModelAdapter(pacient);
        table = new JTable(tm);
        table.setRowHeight(36);
        JScrollPane pane = new JScrollPane(table);
        pane.setPreferredSize(
          new java.awt.Dimension(300, 200));
        this.getContentPane().add(pane);


  }
```

Atera zaigun erantzuna:



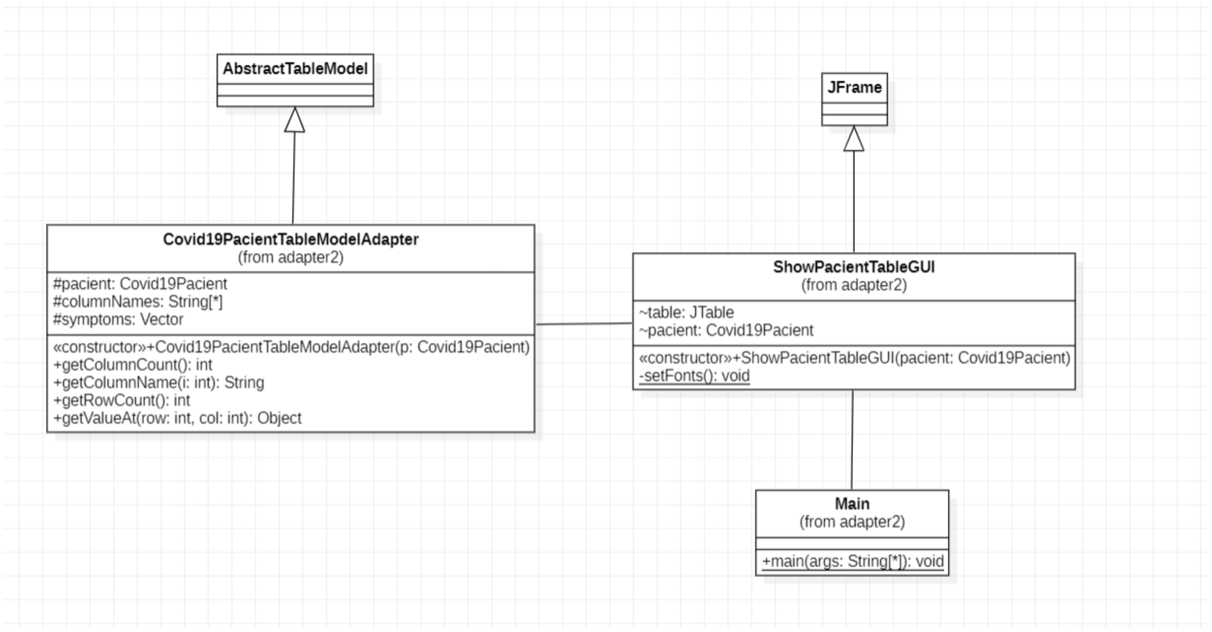
Covid Symptoms aitor

Symptom	Weight
cefalea	1
disnea	2
astenia	3

Covid Symptoms juan

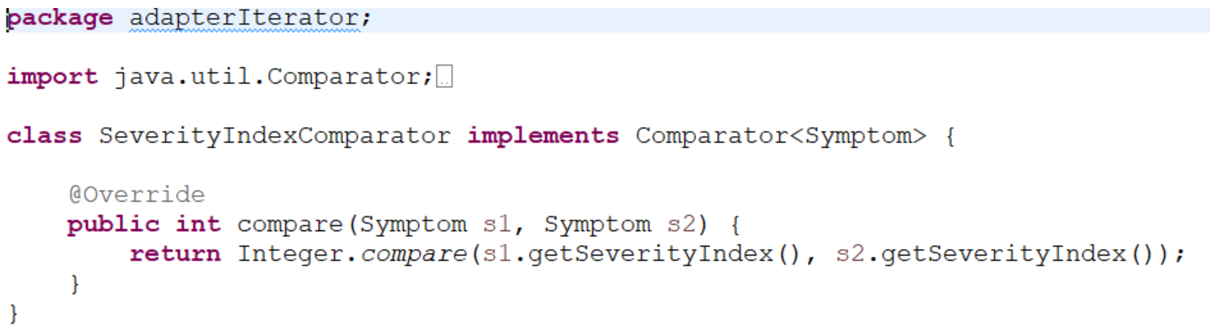
Symptom	Weight
disnea	2
astenia	3
cefalea	1

UML:

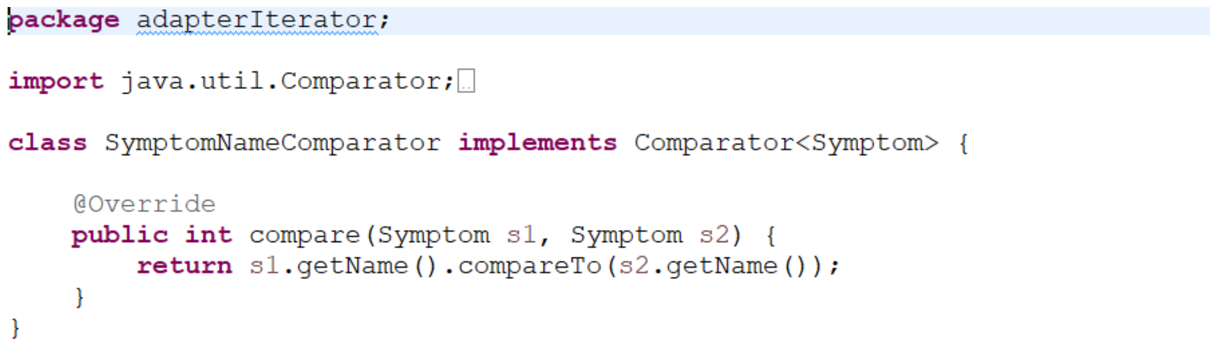


Adapter eta Iterator Patroiak

Lehenengo comparator Comparator interfazea inplementatzen dituzten bi klase definitu behar ziren elementuak symptomName eta severityIndex ordenatzen dituztenak hurrenez hurren.

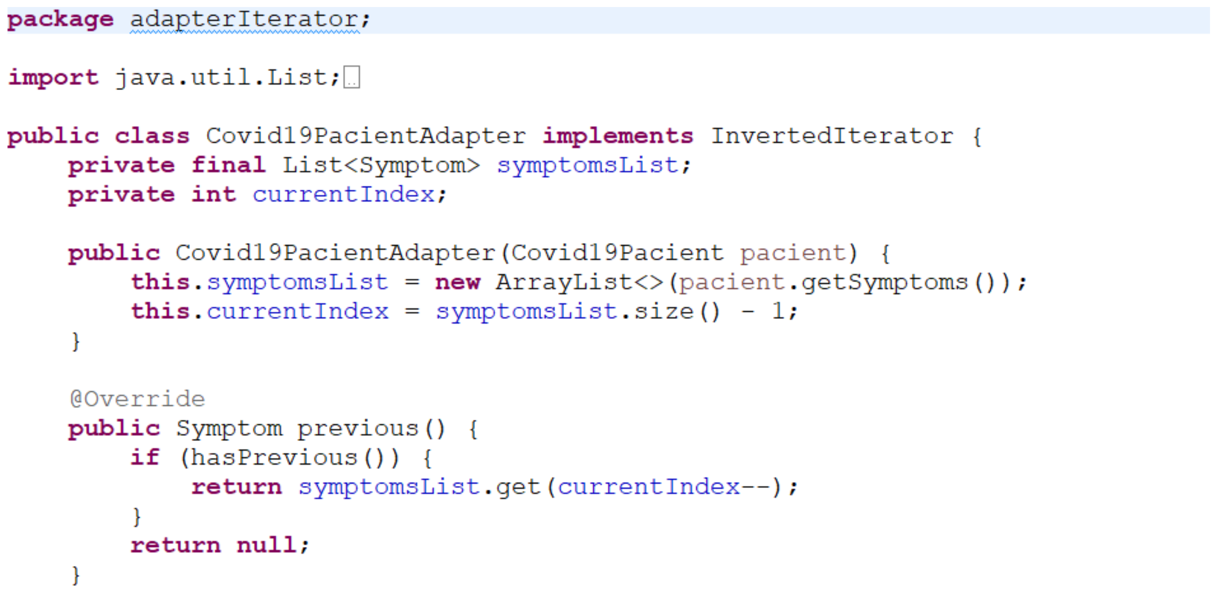


```java
package adapterIterator;

import java.util.Comparator;

class SeverityIndexComparator implements Comparator<Symptom> {

    @Override
    public int compare(Symptom s1, Symptom s2) {
        return Integer.compare(s1.getSeverityIndex(), s2.getSeverityIndex());
    }
}
```



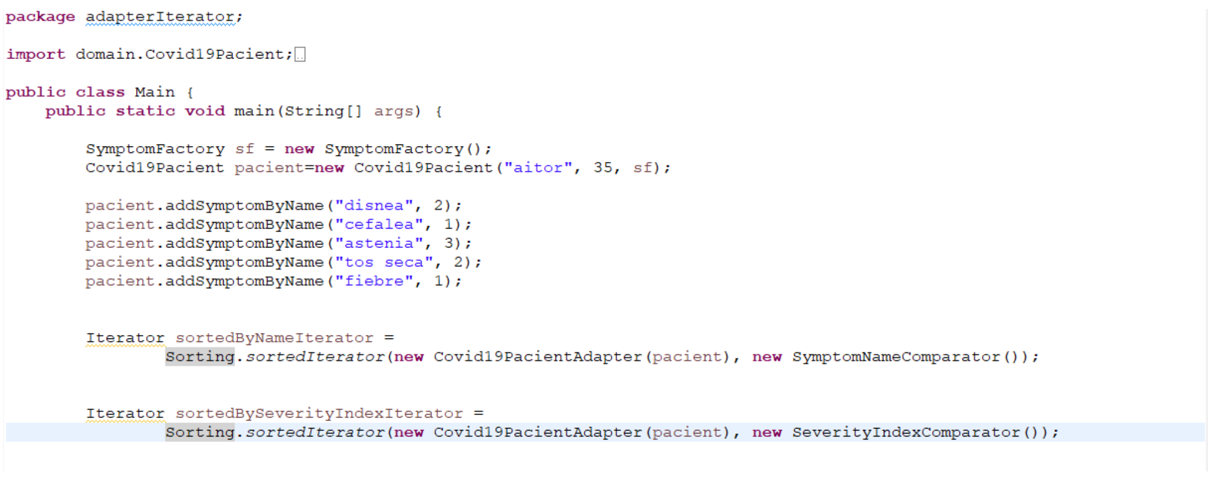
```java
package adapterIterator;

import java.util.Comparator;

class SymptomNameComparator implements Comparator<Symptom> {

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

Ondoren Covid19Pacient klasea InvertedIterator interfazera egokitzen duen klase adaptadorea sortu behar zen. Bertan previous, hasPrevious eta goLast metodoak egin genituen, hauek InvertedIterator interfazetik implementatzen ziren.



```java
package adapterIterator;

import java.util.List;

public class Covid19PacientAdapter implements InvertedIterator {
    private final List<Symptom> symptomsList;
    private int currentIndex;

    public Covid19PacientAdapter(Covid19Pacient pacient) {
        this.symptomsList = new ArrayList<>(pacient.getSymptoms());
        this.currentIndex = symptomsList.size() - 1;
    }

    @Override
    public Symptom previous() {
        if (hasPrevious()) {
            return symptomsList.get(currentIndex--);
        }
        return null;
    }
    @Override
    public boolean hasPrevious() {
        return currentIndex >= 0;
    }

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

Azkenik main klasean paziente bat 5 simtomekin sortu eta hauek 2 moduetan ordenatu eta imprimatu.



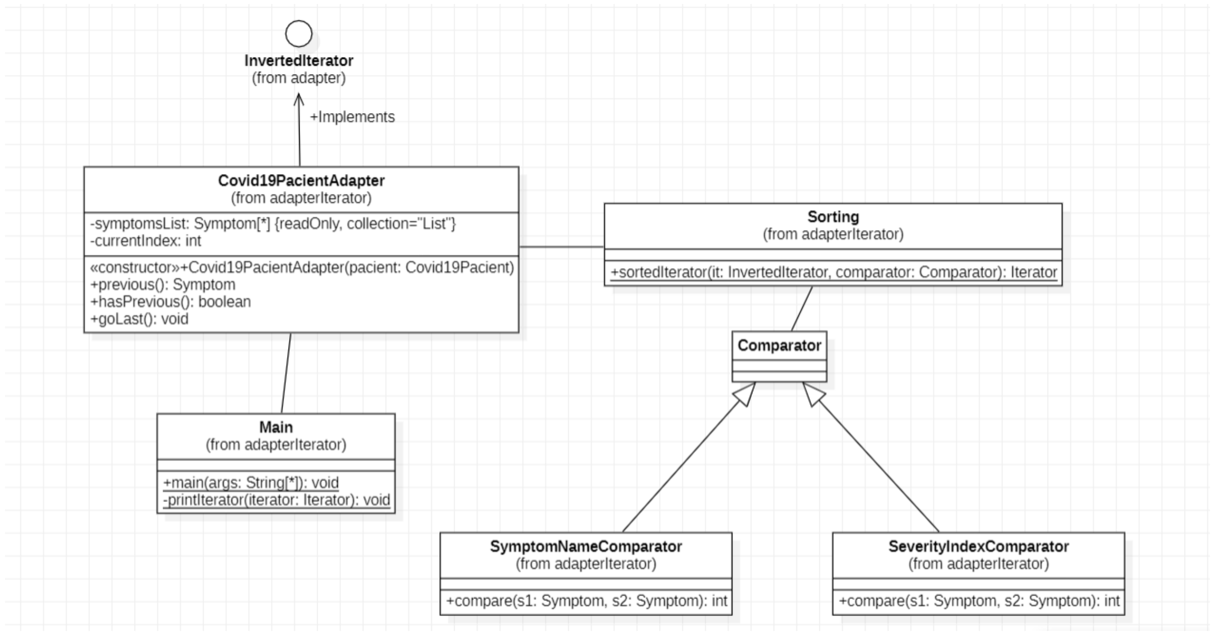
```java
package adapterIterator;

import domain.Covid19Pacient;

public class Main {
    public static void main(String[] args) {

        SymptomFactory sf = new SymptomFactory();
        Covid19Pacient pacient=new Covid19Pacient("aitor", 35, sf);

        pacient.addSymptomByName("disnea", 2);
        pacient.addSymptomByName("cefalea", 1);
        pacient.addSymptomByName("astenia", 3);
        pacient.addSymptomByName("tos seca", 2);
        pacient.addSymptomByName("fiebre", 1);


        Iterator sortedByNameIterator =
                Sorting.sortedIterator(new Covid19PacientAdapter(pacient), new SymptomNameComparator());


        Iterator sortedBySeverityIndexIterator =
                Sorting.sortedIterator(new Covid19PacientAdapter(pacient), new SeverityIndexComparator());
```

```java
        System.out.println("Symptoms ordered by Name:");
        printIterator(sortedByNameIterator);

        System.out.println("\nSymptoms ordered by Severity Index:");
        printIterator(sortedBySeverityIndexIterator);
    }

    private static void printIterator(Iterator iterator) {
        while (iterator.hasNext()) {
            Symptom symptom = (Symptom) iterator.next();
            System.out.println(symptom.getName() + " - Severity Index: " + symptom.getSeverityIndex());
        }
    }
```

Erantzuna:

```
Symptoms ordered by Name:
astenia - Severity Index: 5
cefalea - Severity Index: 3
disnea - Severity Index: 3
fiebre - Severity Index: 5
tos seca - Severity Index: 5

Symptoms ordered by Severity Index:
disnea - Severity Index: 3
cefalea - Severity Index: 3
fiebre - Severity Index: 5
astenia - Severity Index: 5
tos seca - Severity Index: 5
```

UML:



InvertedIterator
(from adapter)
+Implements
Covid19PacientAdapter
(from adapterIterator)
-symptomsList: Symptom[*] {readOnly, collection="List"}
-currentIndex: int
«constructor»+Covid19PacientAdapter(pacient: Covid19Pacient)
+previous(): Symptom
+hasPrevious(): boolean
+goLast(): void
Sorting
(from adapterIterator)
+sortedIterator(it: InvertedIterator, comparator: Comparator): Iterator
Comparator
Main
(from adapterIterator)
+main(args: String[*]): void
-printIterator(iterator: Iterator): void
SymptomNameComparator
(from adapterIterator)
+compare(s1: Symptom, s2: Symptom): int
SeverityIndexComparator
(from adapterIterator)
+compare(s1: Symptom, s2: Symptom): int