Lab Patterns

- **Egileak:** Asier Aldai eta Aimar Villegas

- **Github helbidea:** https://github.com/Aimarville/labpatterns.git

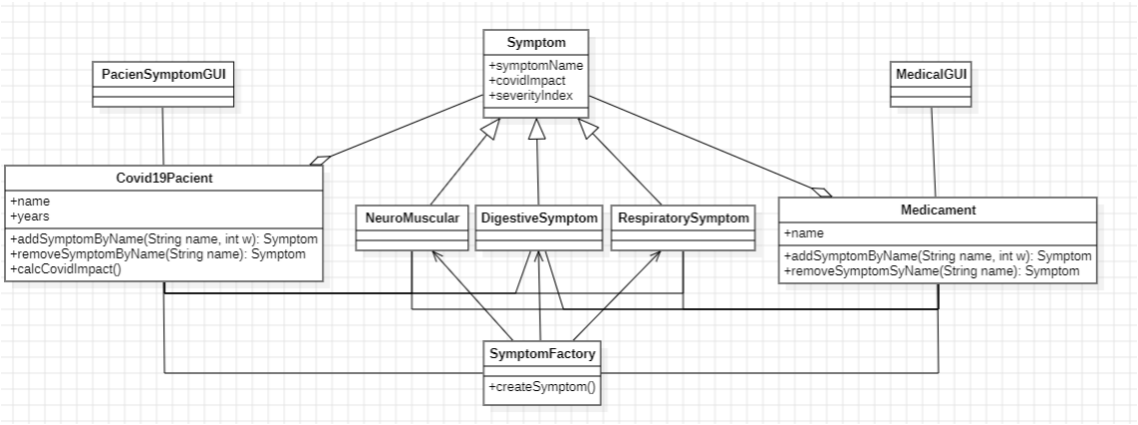
Simple factory:

Ahultasunak:

1. **Zer gertatzen da, sintoma mota berri bat agertzen bada (adb: MovilitySymptom)?**
    - Sintoma mota berri bat gehitu nahi dugunean Covid19Pacient eta Medicament klaseetan createSymptom metodoa aldatu beharko da. Hau konpontzeko, SymptomFactory klasea sortu dugu, non, createSymptom metodoa inplementatzen den. Horrela, Covid19Patient eta Medicament klaseetan Symptom berri bat sortu nahi den aldiro Factory-ra deituko du.
2. **Nola sortu daiteke sintoma berri bat orain arte dauden klaseak aldatu gabe (OCP printzipioa)?**
    - Sintoma berri bat sortzeko SymptomFactory klaseko createSymptom metodoan zerrendatan sintoma eta honen pisua gehitu beharko da bakarrik.
3. **Zenbat erresponsabilitate dauzkate *Covid19Pacient* eta *Medicament* klaseak (SRP printzipioa)?**
    - Covid19Pacient klaseak 3 errespontsabilitate ditu, sintoma bat bere izenaz gehitzea, sintoma bat bere izenaz ezabatzea eta covid inpaktua kalkulatzea. Bestetik, Medicament klasearen errespontsabilitateak Covid19Pacient klasearen lehen bi errespontsabilitate berdinak ditu.

Eskatzen da:

1. UML



- SymptomFactory klase berri bat sortu dut Symptom motak sortzeko. Klase hau Covid19Pacient eta Medicament klaseek erabiltzen dute Symptom berri bat sortzeko.

2. Mareos sintoma gehitu

- createSymptom metodoan impact1 zerrendan eta neuroMuscularSymtpom zerrendan “mareos” sintoma gehitu dut. Baita ere, index1 zerrendan covid inpaktua gehitu dut.

3. Sintoma objektu bakarrak

- Lehenik eta behin, singleton patroia aplikatu dut SymptomFactory klasera. Honen bidez, sisteman SymptomFactory instantzia bakarra egongo da. Hortaz gain, “cache” moduko bat egin dut non orain arte momenturaino sortutako sintoma guztiak gordeta dauden. Azkenik, Covid19Patient eta Medicament klaseetan SymptomFactory klaseko instantzia sortu beharko da getInstance() metodoaren bidez.

Observer:

Lehen prototipoa:

```
public Symptom addSymptomByName(String symptom, Integer w){
    Symptom s=null;
    s=sm.createSymptom(symptom);
    if (s!=null)
        symptoms.put(s,w);
    setChanged();
    notifyObservers();
    return s;
}
```

```
public class PacientObserverGUI extends JFrame implements Observer{

    private JPanel contentPane;
    private final JLabel symptomLabel = new JLabel("");
    private Observable obs;

    /**
     * Create the frame.
     */
    public PacientObserverGUI(Observable obs) {
        setTitle("Pacient symptoms");
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setBounds(650, 100, 200, 300);
        contentPane = new JPanel();
        contentPane.setBorder(new EmptyBorder(5, 5, 5, 5));
        setContentPane(contentPane);
        contentPane.setLayout(null);
        symptomLabel.setBounds(19, 38, 389, 199);
        contentPane.add(symptomLabel);
        symptomLabel.setText("Still no symptoms");
        this.setVisible(true);
        this.obs = obs;
        obs.addObserver(this);
    }
```

```
  public PacientSymptomGUI(Covid19Pacient p) {
      this.pacient = p;
```

```
 //addSymptomByName ...
 p.addSymptomByName(((Symptom)symptomComboBox.getSelectedItem()).getName(), Integer.parseInt(weightField.getText()));
```

```
//removeSymptomByName...
p.removeSymptomByName(((Symptom)symptomComboBox.getSelectedItem()).getName());
```

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


    Observable pacient2=new Covid19Pacient("mikel", 20);
    new PacientObserverGUI (pacient2);
    new PacientSymptomGUI ((Covid19Pacient)pacient2);
```

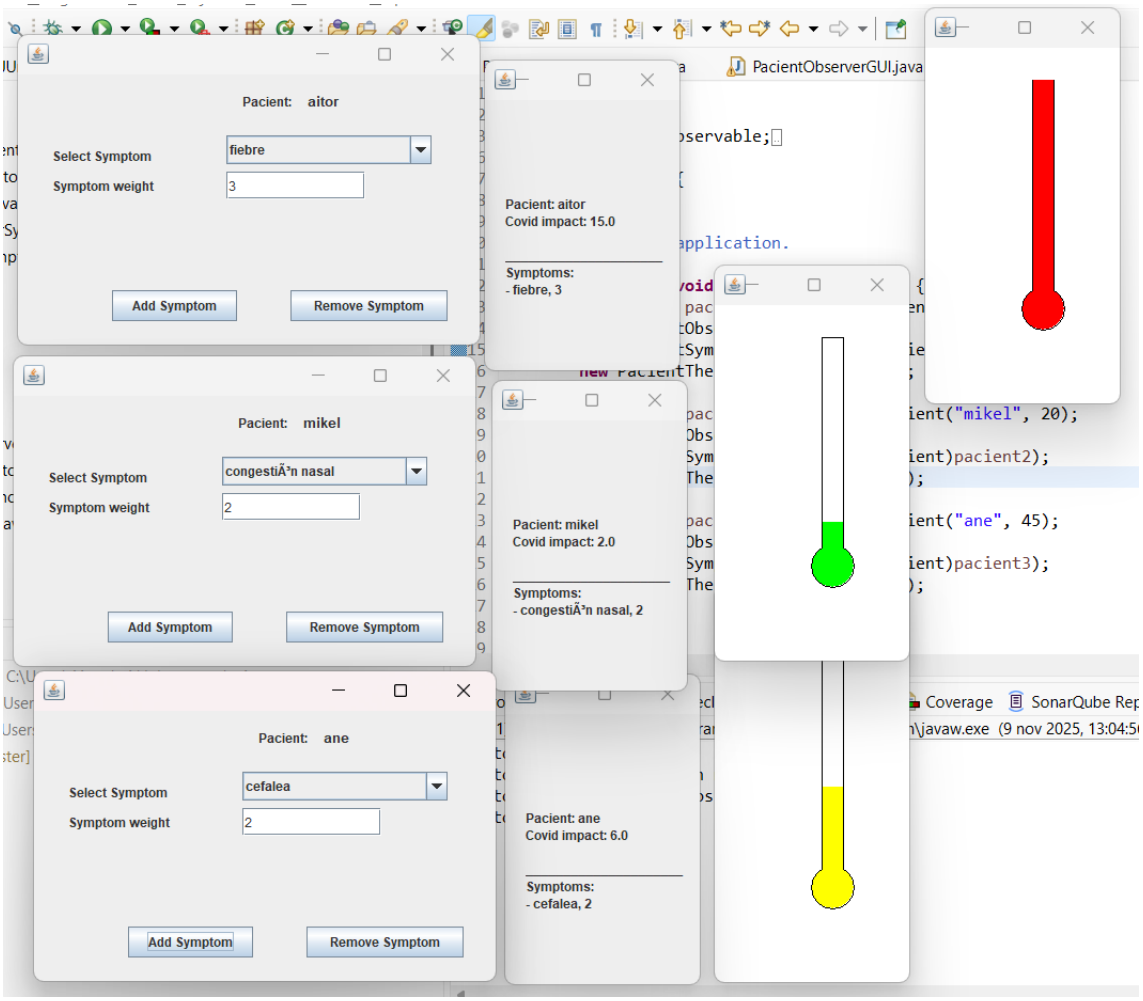
Eskatzen da:

- Bigarren bloke bat gehitu dugu non bigarren paziente bat sortzen den.

Bigarren prototipoa

```
private Observable obs;

public PacientThermometerGUI(Observable obs){
    super("Temperature Gauge");
    Panel Top = new Panel();
    add("North", Top);
    gauges = new TemperatureCanvas(0,15);
    gauges.setSize(500,280);
    add("Center", gauges);
    setSize(200, 380);
    setLocation(0, 100);
    setVisible(true);
    this.obs = obs;
    obs.addObserver(this);
}
```

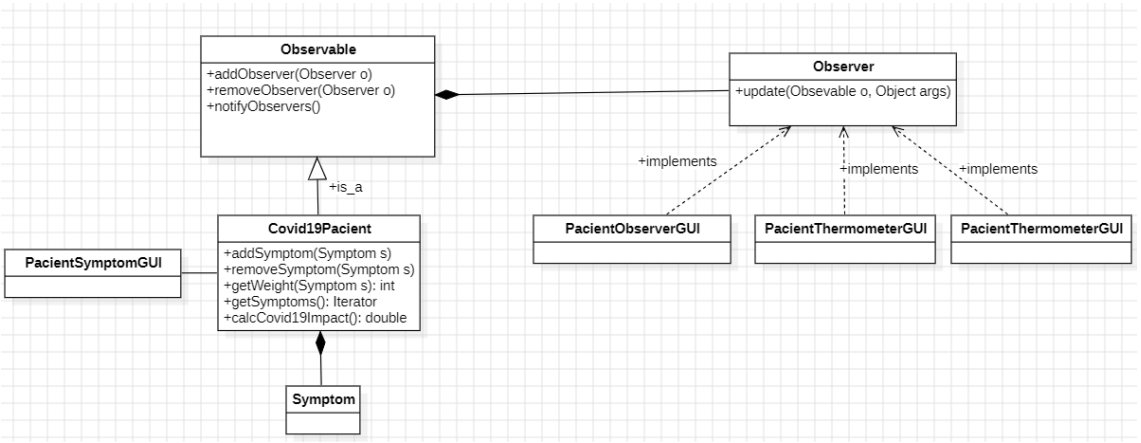
Eskatzen da:



Hirugarren prototipoa

Eskatzen da:

1.



2.

```
    private Observable obs;

    /** stores the associated ConcreteSubject */
    public SemaphorGUI (Observable obs) {
        setSize(100, 100);
        setLocation(350,10);
        Color c=Color.green;
        getContentPane().setBackground(c);
        repaint();
        setVisible(true);
        this.obs = obs;
        obs.addObserver(this);
    }

    @Override
    public void update(Observable o, Object arg) {
        Covid19Pacient p=(Covid19Pacient)o;
        Color c;
        double current=p.covidImpact();
        if (current<5) c=Color.green;
        else if (current<=10) c=Color.yellow;
        else c=Color.red;
        getContentPane().setBackground(c);
        repaint();
    }
```

```
public static void main(String[] args) {
    Observable pacient=new Covid19Pacient("aitor", 35);
    new PacientObserverGUI (pacient);
    new PacientSymptomGUI ((Covid19Pacient)pacient);
    new PacientThermometerGUI(pacient);
    new SemaphorGUI(pacient);

    Observable pacient2=new Covid19Pacient("mikel", 20);
    new PacientObserverGUI (pacient2);
    new PacientSymptomGUI ((Covid19Pacient)pacient2);
    new PacientThermometerGUI(pacient2);
    new SemaphorGUI(pacient2);

    Observable pacient3=new Covid19Pacient("ane", 45);
    new PacientObserverGUI (pacient3);
    new PacientSymptomGUI ((Covid19Pacient)pacient3);
    new PacientThermometerGUI(pacient3);
    new SemaphorGUI(pacient3);
}
```

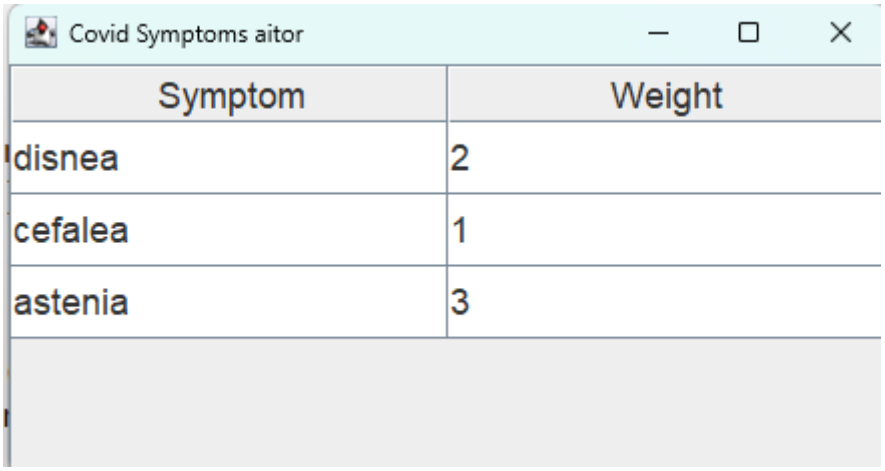
Adapter:

Eskatzen da:

1. **Covid19PacientTableModelAdapter inplementatu eta emaitza konprobatu.**

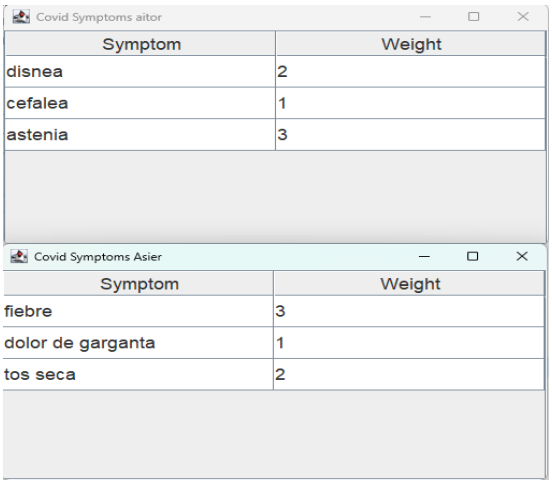
```
import java.util.ArrayList;

public class Covid19PacientTableModelAdapter extends AbstractTableModel {
      protected Covid19Pacient pacient;
      protected String[] columnNames =
        new String[] {"Symptom", "Weight" };

      public Covid19PacientTableModelAdapter(Covid19Pacient p) {
        this.pacient=p;
      }

      @Override
      public int getColumnCount() {
        return columnNames.length;
      }

      @Override
      public String getColumnName(int i) {
        return columnNames[i];
      }

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

      @Override
      public Object getValueAt(int row, int col) {
        List<Symptom> symptomList = new ArrayList<>(pacient.getSymptoms());
        Symptom s = symptomList.get(row);

        Object ret = null;
        if (col == 0) {
            ret = s.getName();
        }else if (col == 1) {
            ret = pacient.getWeight(s);
        }
        return ret;
      }
    }
```

- getColumnCount metodorako columnNames atributuaren luzera itzuli dut, zuzenean hor daukagulako. getColumnName metodoan, parametro bezala sartzen den String array-aren posizioa bueltatu dut. getRowCount metodorako pacient-aren Symptom list lortu eta honen luzera bueltatu dut. Azkenik, getValueAt metodoan, pacient.getSymptoms() metodoak bueltatzen duen Set-a lista batean sartu dut, parametro bezala row sartzen delako indizea bezala, eta ezin dut Set batentzat erabili. Gero, sartzen den col arabera sintomaren izena edo pixua bueltatzen dut.

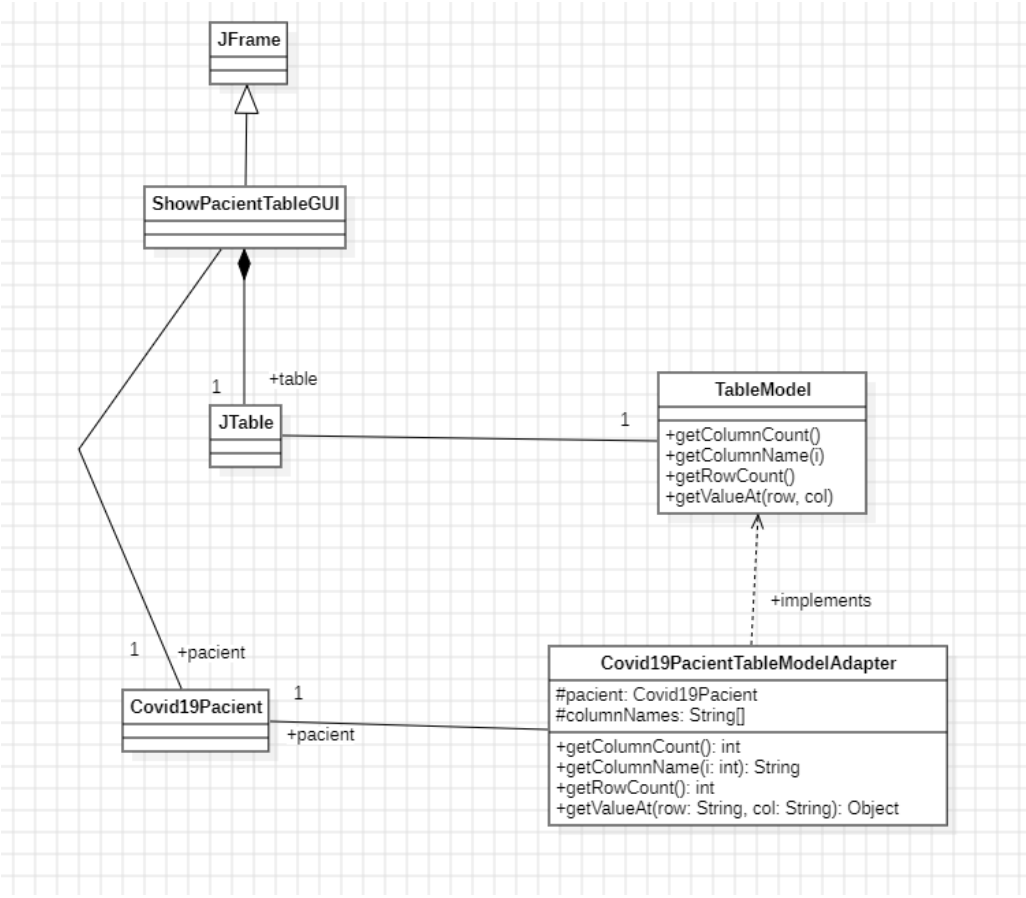


2. Beste paziente (bere sintomekin) bat gehitu eta emaitza konprobatu

```
package adapter2;

import domain.Covid19Pacient;

public class Main {

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

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

        Covid19Pacient pacient2=new Covid19Pacient("Asier", 20);

        pacient2.addSymptomByName("fiebre", 3);
        pacient2.addSymptomByName("tos seca", 2);
        pacient2.addSymptomByName("dolor de garganta", 1);

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

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


    }

}
```



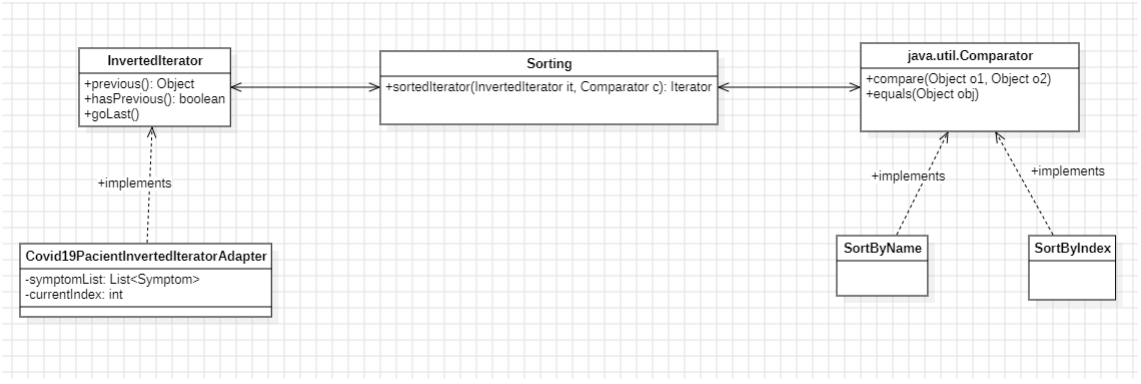
Amaierako UML:



Adapter Iterator eta Patroiak:

Eskatzen da:

1. **UML:**



2. **Bi comparator klase definitu, izenaz eta larritasunaz ordenatzen dutenak.**

```java
package iterator;

import java.util.Comparator;

public class SortByName implements Comparator{

    public int compare(Object s1, Object s2) {
        String sn1 = ((Symptom)s1).getName();
        String sn2 = ((Symptom)s2).getName();

        return sn1.compareTo(sn2);
    }

}
```

```java
package iterator;

import java.util.Comparator;

public class SortByIndex implements Comparator{

    public int compare(Object s1, Object s2) {
        int si1 = ((Symptom)s1).getSeverityIndex();
        int si2 = ((Symptom)s2).getSeverityIndex();
        return Integer.compare(si1, si2);
    }

}
```

3. Covid19Pacient klasea InvertedIterator interfazera adaptadore bat sortu.

```
package iterator;

import java.util.ArrayList;

public class Covid19PacientInvertedIteratorAdapter implements InvertedIterator{

    private List<Symptom> symptomList;
    private int currentIndex;

    public Covid19PacientInvertedIteratorAdapter (Covid19Pacient p) {
        Set<Symptom> symptoms = p.getSymptoms();
        this.symptomList = new ArrayList<> (symptoms);
    }

    @Override
    public Object previous() {
        return symptomList.get(currentIndex--);
    }

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

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

- Metodoak inplementatu ahal izateko, currentIndex atributu bat sortu dut. Constructor-ean pazientearen sintomen Set-a atera dut eta ArrayList bat bihurtu dut. previous metodorako currentIndex-i aurreko indizearen balioa eman diot. hasPrevious metodoan currentIndex lehen indizea ez dela konprobatzen dut. Azkenik, goLast metodoan currentIndex-i listaren azkeneko indizearen balioa ematen diot.

4. Programa nagusian paziente bat sortu 5 sintomekin eta sortutako bi modutan ordenatu lista.

```
package iterator;

import java.util.Iterator;

    public class Main {

        public static void main(String[] args) {
            Covid19Pacient p=new Covid19Pacient("Ane", 29);
            p.addSymptom(new Symptom("s1", 10, 5), 1);
            p.addSymptom(new Symptom("s2", 10, 4), 2);
            p.addSymptom(new Symptom("s3", 10, 2), 3);
            p.addSymptom(new Symptom("s4", 10, 3), 4);
            p.addSymptom(new Symptom("s5", 10, 1), 5);

            Covid19PacientInvertedIteratorAdapter adapter = new Covid19PacientInvertedIteratorAdapter(p);

            Iterator<Symptom> it1 = Sorting.sortedIterator(adapter, new SortByName());
            System.out.println("Izenaren arabera ordenatuta:");
            while (it1.hasNext()) {
                System.out.println(it1.next());
            }

            Iterator<Symptom> it2 = Sorting.sortedIterator(adapter, new SortByIndex());
            System.out.println("Larritasunaren arabera ordenatuta:");
            while (it2.hasNext()) {
                System.out.println(it2.next());
            }

        }

    }
```

- Pazientea eta honen sintomak sortu ondoren, lehen sortutako adapter-ari deitu dut gero erabiltzeko. Alde batetik, izenaren araberako ordenazioa probatzeko Iterator klaseko it2 instantzia bat sortu dut, parametro bezala sortutako adapter-a eta izen areberako comparator-a sartuz. Gero, iterator erabiliz elementu bakoitza erakutsi dut kontsolan. Beste aldetik, larritasunaren araberako ordenazioa probatzeko, it2 Iterator klaseko instantzia berriari, adapter eta larritasun araberako comparator-a sartu dut parametro bidez. Azkenik, listako elemenu bakoitza inprimatu.