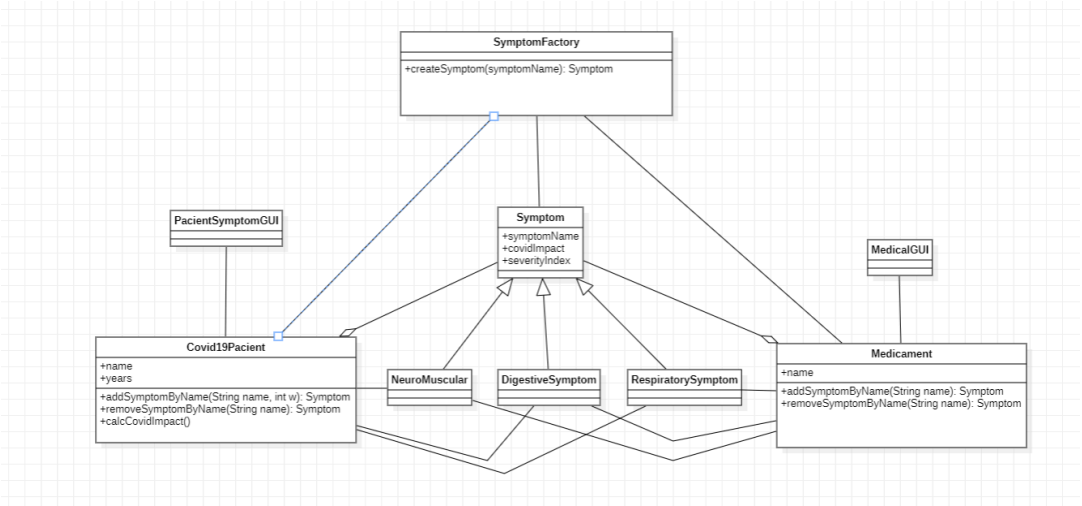
Github proiektuaren esteka: https://github.com/Errazkin08/labpatterns

Simple Factory

UML hedatua:



1) SymptomFactory izeneko klasea sortu dugu, sintomak sortzeko, sintoma motarekiko independientea dena. Modu hontan sintoma mota berri bat sartzen badugu, ez dugu aldaketa extrarik egin beharko.
2) Mareos beheko kodean agertzen da, digestive motako sintomen artean sartu dugu.
3) Hau lortzeko singleton patroia erabili behar da, sintoma bakoitza bakarra izateko. Ez dugu inplementatu.

SymptomFactory-ko kodea:

```
package domain;

import java.util.Arrays;

public class SymptomFactory {

    public SymptomFactory() {}

    public static 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",
        List<Double> index1 = Arrays.asList(5.0, 4.8, 3.7, 0.9, 0.8,3.0,1.0);

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


        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;

    }
}
```

Observer Patroia

Lehendabiziko prototipoa:

1. Pausoa CovidPacient Observable klasea:

Observable klasetik hedatzeko extends jartzen dugu, eta sintoma bat jartzen edo kentzen dugunean, abisatzeko notifyObservers() eta setChanged() metodoei deitzen diegu.

```
public class Covid19Pacient extends Observable{

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

    public Symptom removeSymptomByName(String symptomName) {
        Symptom s=getSymptomByName(symptomName);
        System.out.println("Simptom to remove: "+s);
        if (s!=null) {
            symptoms.remove(s);
            notifyObservers();
            setChanged();
        }
        return s;
    }
```

2. Pausoa: PacientObserverGUI Observer klasea

Observer gisa erabiliko dugun GUI-an, observer inplementatuko dugu, eta eraikitzailearen barruan, obs objektua jasoko dugu eta observer moduan jarriko dugu addObserver eginez.

```
1  package observer;
2
3  import java.util.Iterator;
15
16 public class PacientObserverGUI extends JFrame implements Observer{
17
18     private JPanel contentPane;
19     private final JLabel symptomLabel = new JLabel("");
20
21     /**
22      * Create the frame.
23      */
24     public PacientObserverGUI(Observable obs) {
25         setTitle("Pacient symptoms");
26         setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
27         setBounds(650, 100, 200, 300);
28         contentPane = new JPanel();
29         contentPane.setBorder(new EmptyBorder(5, 5, 5, 5));
30         setContentPane(contentPane);
31         contentPane.setLayout(null);
32         symptomLabel.setBounds(19, 38, 389, 199);
33         contentPane.add(symptomLabel);
34         symptomLabel.setText("Still no symptoms");
35         this.setVisible(true);
36         obs.addObserver(this);
37     }
```

Honez gain, Observer interfazea inplementatzen dugunez, honek duen update() metodoa gainidatzi beharra dugu, honela egin dugu:

```
    @Override
    public void update(Observable o, Object arg) {
        Covid19Pacient p=(Covid19Pacient)o;
        String s="<html> Pacient: <b>"+p.getName()+"</b> <br>";
        s=s+"Covid impact: <b>"+p.covidImpact()+"</b><br><br>";
        s=s+" _____ <br> Symptoms: <br>";
        Iterator<Symptom> i=p.getSymptoms().iterator();
        Symptom p2;
        while (i.hasNext()) {
        p2=i.next();
        s=s+ " - " + p2.toString()+", "+p.getWeight(p2)+"<br>";
        }
        s=s+"</html>";
        symptomLabel.setText(s);
    }
```

3. Pausoa: PacientSymptomGUI klasea:
Observerra eginda, orain informazioa erakutsiko duen GUI-a eguneratu beharko dugu. Honetarako PacientSymptomGUI klasean aldaketa hauek egin ditugu:

Eraikitzailean pacient motako aldagai bat sartuko dugu:

```
     */
    public PacientSymptomGUI(Covid19Pacient p) {
```

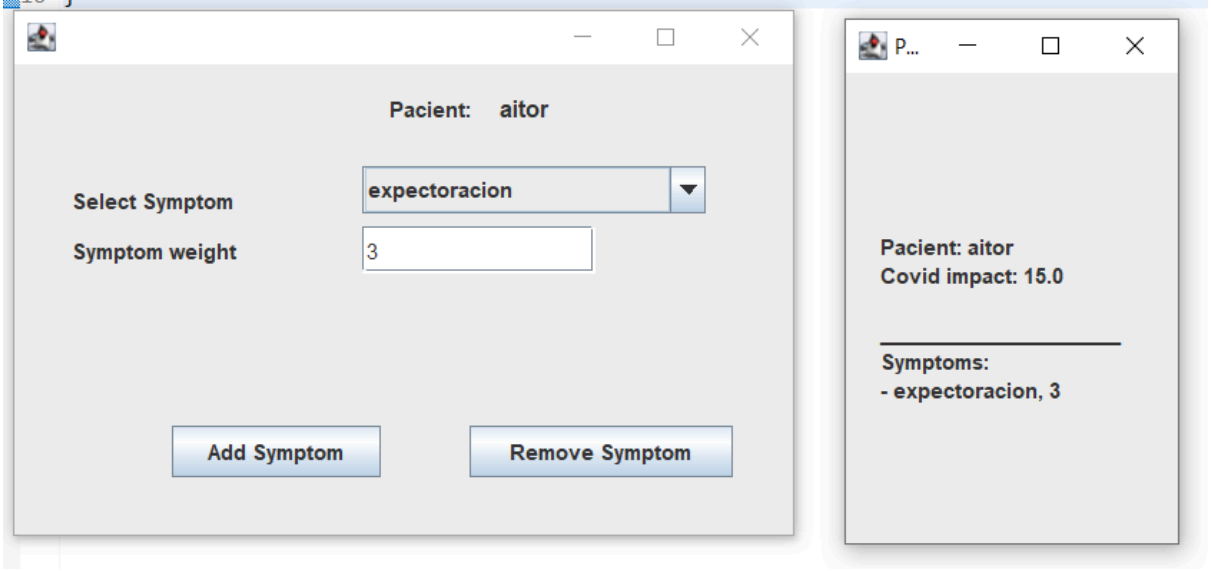
Aldaketa bat egiten denean (sartzeko edo kentzeko botoi bat sakatzean), lehenago inplementatu ditugun funtzioei deitzea action listenerretatik:

```
JButton btnNewButton = new JButton("Add Symptom");
btnNewButton.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        errorLabel.setText(" ");
        if (new Integer(weightField.getText())<=3) {
        System.out.println("Symptom added :"+(Symptom)symptomComboBox.getSelectedItem());

        p.addSymptomByName(((Symptom)symptomComboBox.getSelectedItem()).getName(), Integer.parseInt(weightFiel

    } else errorLabel.setText("ERROR, Weight between [1..3]");

    }


btnRemoveSymptom = new JButton("Remove Symptom");
btnRemoveSymptom.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        errorLabel.setText(" ");

        System.out.println("Symptom removed :"+(Symptom)symptomComboBox.getSelectedItem());

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

    }
});
```

4. Pausoa: Programa nagusia sortu.

Covid19Pacient.java | PacientObserverGUI.java | PacientSymptomGUI.java | Main2.java | Main1.java

```
1  package observer;
2
3  import java.util.Observable;
8
9  public class Main1 {
10
11     public static void main(String  args[]){
12         Covid19Pacient  pacient=new Covid19Pacient("aitor", 35);
13         new PacientObserverGUI(pacient);
14         new PacientSymptomGUI (pacient);
15     }
16 }
```



Programa nagusiak horrela funtzionatzen du. Arazoren bat dauka sintomak gehitzean lehenengo aldiz, baina hau ez da guk inplementatukoaren arazoa, uste dugu.

Bigarren Prototipoa:

Hasteko, PacientThermometerGUI klaseak Observer klasea implementatzea egin dugu, jarraian eraikitzaileari Observable objetu bat pasatzeko eskatu addObserver() metodoa erabiltzeko eta azkenik update metodoa inplementatu diogu:

```
public class PacientThermometerGUI extends Frame implements Observer{
```

```java
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);
    obs.addObserver(this);
}
```

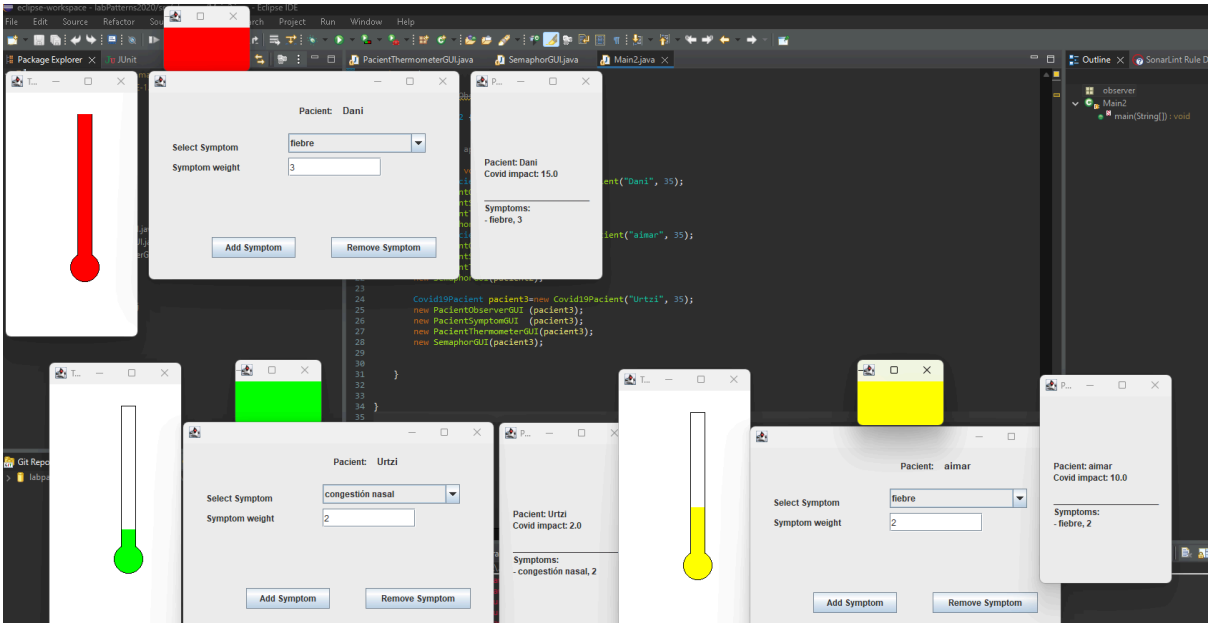
```java
@Override
public void update(Observable o, Object arg) {
    Covid19Pacient  p=(Covid19Pacient)  o;
    //      Obtain  the     current covidImpact to  paint
    int farenheit = (int)   p.covidImpact();
    //      temperature gauge   update
    gauges.set(farenheit);
    gauges.repaint();

}
```

Ondoren, SemaphorGUI klasea ere moldatu dugu Observer bat izateko eta main programan ondo agertzeko:

```java
package observer;

import java.awt.Color;


public class SemaphorGUI extends JFrame implements Observer{
    /** stores the associated ConcreteSubject */
    public SemaphorGUI (Observable obs) {

        setSize(100, 100);
        setLocation(350,10);
        Color c=Color.green;
        getContentPane().setBackground(c);
        repaint();
        setVisible(true);
        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();

    }
}
```

Hemen dago erabilitako main2 programa, eta bere emaitza:

```java
public static void main(String[] args) {
    Covid19Pacient pacient=new Covid19Pacient("Dani", 35);
    new PacientObserverGUI (pacient);
    new PacientSymptomGUI  (pacient);
    new PacientThermometerGUI(pacient);
    new SemaphorGUI(pacient);
    Covid19Pacient pacient2=new Covid19Pacient("aimar", 35);
    new PacientObserverGUI (pacient2);
    new PacientSymptomGUI  (pacient2);
    new PacientThermometerGUI(pacient2);
    new SemaphorGUI(pacient2);

    Covid19Pacient pacient3=new Covid19Pacient("Urtzi", 35);
    new PacientObserverGUI (pacient3);
    new PacientSymptomGUI  (pacient3);
    new PacientThermometerGUI(pacient3);
    new SemaphorGUI(pacient3);


}
```



Adapter Patroia

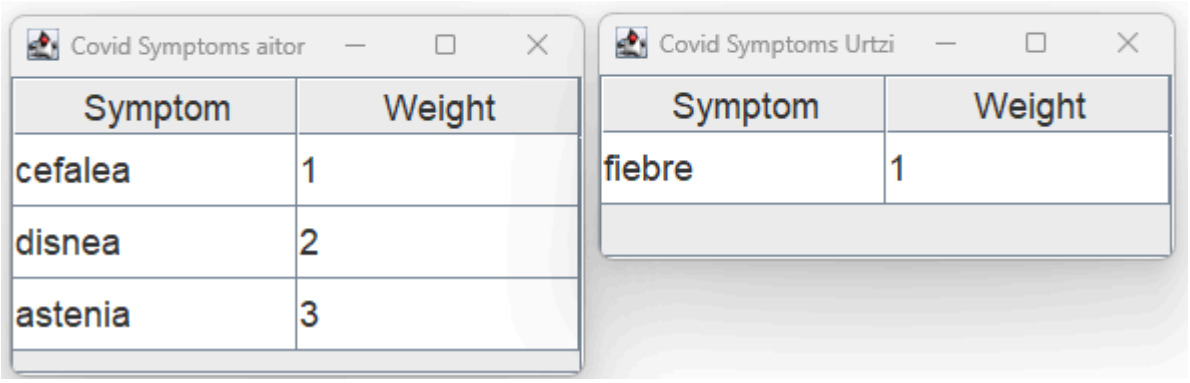
Covid19PacientTableModelAdapter klasea osatu dugu, behar ziren metodoak betez:

```
import java.util.Iterator;

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

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

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

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

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

      public Object getValueAt(int row, int col) {
        // Challenge!

          Iterator<Symptom> it=pacient.getSymptoms().iterator();
          int i=0;
          Symptom s=null;
          while(i <=row && it.hasNext()) {
              s= (Symptom)it.next();
              i++;
          }
          if(col==0) {
              return s;
          }
          return pacient.getWeight(s);
      }
    }
```

Main-ean beste paziente bat sartu dugu eta exekutatu:

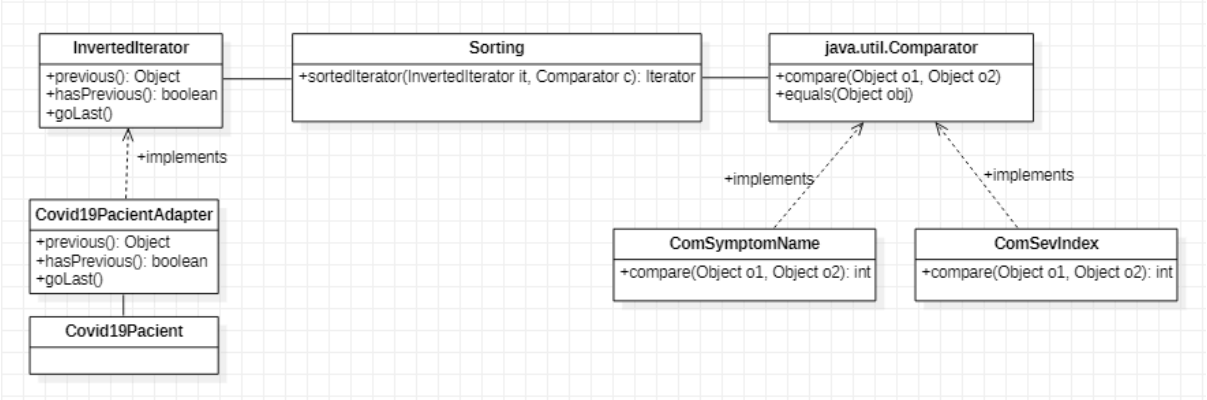
```
public class Main {

    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("Urtzi", 6);

        pacient2.addSymptomByName("fiebre", 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(500, 200));
        gui2.setVisible(true);


    }

}
```

Horrela agertzen da exekutatu ondoren:



Adapter eta Iterator Patroiak

UML hedatua:



Inplementazioa:

Comparator ezberdinak inplementatu, Adapter bat sortu Covid19Pacient InvertedIteratorrekin erabiltzeko eta Main osatu behar izan dugu:

ComSymptomName:

```
package iterator;

import java.util.Comparator;

public class ComSymptomName implements Comparator<Object>{

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

}
```

ComSevIndex:

```
package iterator;

import java.util.Comparator;

public class ComSevIndex  implements Comparator<Object>{

    @Override
    public int compare(Object o1, Object o2) {
        return ((Symptom) o1).getSeverityIndex()- ((Symptom)o2).getSeverityIndex();
    }

}
```

Covide19PacientAdapter:

```java
package iterator;

import java.util.*;

public class Covid19PacientAdapter implements InvertedIterator{
    private  Covid19Pacient pacient;
    private int index;
    private List<Symptom> lista;

    public Covid19PacientAdapter(String name,int years) {
        pacient=new Covid19Pacient(name,years);
        index=0;
        lista= new ArrayList<Symptom>(pacient.getSymptoms());
        Collections.sort(lista,new ComSymptomName());
    }
    public Covid19PacientAdapter (Covid19Pacient pacient) {
        this.pacient=pacient;
        index=0;
        lista= new ArrayList<Symptom>(pacient.getSymptoms());
        Collections.sort(lista,new ComSevIndex());
    }

    @Override
    public Object previous() {
        index--;
        return lista.get(index) ;
    }

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

    @Override
    public void goLast() {
        index=lista.size();

    }

}
```

Main:

```java
package iterator;

import adapter.InvertedIterator;

    public class Main {

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

            InvertedIterator i=new Covid19PacientAdapter(p);
            i.goLast();
            while(i.hasPrevious())
                System.out.println(i.previous());

        }

    }
```