eman ta zabal zazu

Universidad del País Vasco
Euskal Herriko Unibertsitatea

IZENBURUA

Diseinu patroiak

INFORMATIKA INGENIARITZA GRADUA

Egileak:

KEPA GAZTAÑAGA

MIRIAM RODRIGUEZ

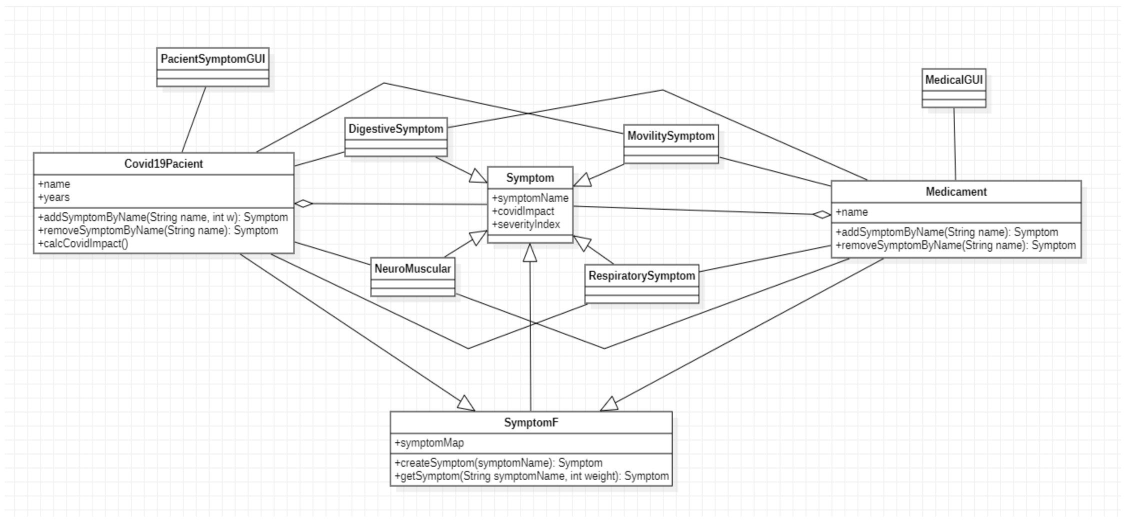
2024-ko urriak 24a

gitHubeko Repositorio esteka:
https://github.com/MiriamRodri286/labpatternsMK.git

AURKIBIDEA

1. Simple Factory

1.1. Diseinu hedatua, UML diagrama hedatua


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

CreateSymptom() metodoaren usain txarra ekiditzeko beste klase bat sortu da, SymptomF klasea. Gainera MovilitySymptom klasea ere sortu dugu 2. eskakizuna bete ahal izateko, hau da, mareo sintoma gehitzeko.

1.2. Kode azpimagarria

- SymptomF klasea:
  Esan bezala, createSymptom metodoa Symptom-etik SymptomF-ra pasatu dugu refaktorizazioaren bidez. Gainera 2. eskakizunean impact1-eko mareos sintoma gehitzea eskatzen zuenez, MovilitySymptom klasea baliatuz createSymptom metodoan gehitu ditugu.

```
public class SymptomF {
        private static Map<String, Symptom> symptomMap = new HashMap<>();

        public static Symptom getSymptom(String symptomName, int weight) {
      Symptom s = symptomMap.get(symptomName);
      if (s == null) {
         s = createSymptom(symptomName, weight);
         if (s != null) {
            symptomMap.put(symptomName, s);
         }
      }
      return s;
   }
   private static Symptom createSymptom(String symptomName, int weight) {
      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("mareos","nauseas", "vomitos",
"congestion nasal","diarrea","hemoptisis","congestion conjuntival");
```

```
        List<Double> index1 = Arrays.asList(5.0, 4.8, 3.7, 0.9, 0.8);

        List<String> digestiveSymptom=Arrays.asList("nauseas", "vomitos","diarrea");
        List<String> neuroMuscularSymptom=Arrays.asList("fiebre", "astenia",
"cefalea", "mialgia","escalofrios");
        List<String> respiratorySymptom=Arrays.asList("tos
seca","expectoracion","disnea","dolor de garganta", "congestion
nasal","hemoptisis","congestion conjuntival");
        List<String> movilitySymptom=Arrays.asList("mareos");
        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);
            }
            if (movilitySymptom.contains(symptomName)) {
                return new RespiratorySymptom(symptomName, (int)index, impact);
            }
        }
        return null;
    }
```

- MovilitySymptom klasea:

```
public class MovilitySymptom extends Symptom{
    public MovilitySymptom(String name, int covidImpact, int severityIndex) {
            super(name, covidImpact, severityIndex);
    }
}
```

1.3. Galderak

1.3.1. Aplikazio ondo diseinatuta egongo legoke, Symptom klaseko objektuak Covid19Pacient objektuetan soilik erabiliko balira, baina zer gertatzen da beste klase batzuk Symptom klaseko objektuak sortu nahi badituzte?

Symptom objektuak beste klase batzuetan zuzenean sortzen badira, errepikapenak eta inkoherentziak sor daitezke. Hori ekiditeko, fabrika eredua erabiltzea gomendatzen da

1.3.2. Zer gertatzen da, sintoma mota berri bat agertzen bada (adb: MovilitySymptom)?

GUI berri bat sortu behar da, honela ez da OCP printzipioa austen.

1.3.3. Nola sortu daiteke sintoma berri bat orain arte dauden klaseak aldatu gabe (OCP printzipioa)?

SymptomF klasea sortu da errefaktorizazioa erabiliz createSympton metodoa SymptomF klasean egoteko. Horrela eginez, bakarrik SymptomF aldatu behar da eta beste guztia ez da ikutu behar.

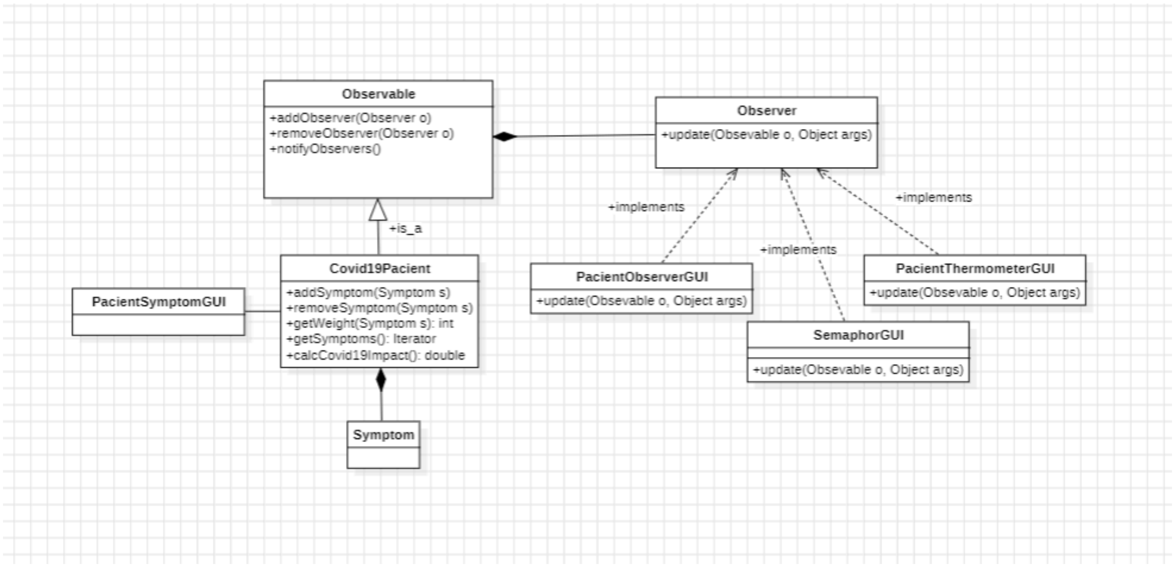
1.3.4. Zenbat erresponsabilitate dauzkate Covid19Pacient eta Medicament klaseak (SRP printzipioa)?

SymptomF aldatu baina lehenago kalkulatu ditugu:

- Covid19Pacient: 4
    - Sintomen kudeaketa (gehitu(add), kendu(remove) edo bilatu (get))
    - CovidImpact kalkulatu
    - Sintomen iterazioa
    - CreateSymptom
- Medicament: 2
    - CreateSymptom
    - Sintomen kudeaketa (gehitu(add), kendu(remove) edo bilatu (get))

2. Observer Patroia

2.1. Diseinu hedatua, UML diagrama hedatua



Observable
+addObserver(Observer o)
+removeObserver(Observer o)
+notifyObservers()
Observer
+update(Obsevable o, Object args)
+is_a
+implements
+implements
+implements
Covid19Pacient
+addSymptom(Symptom s)
+removeSymptom(Symptom s)
+getWeight(Symptom s): int
+getSymptoms(): Iterator
+calcCovid19Impact(): double
PacientSymptomGUI
Symptom
PacientObserverGUI
+update(Obsevable o, Object args)
PacientThermometerGUI
+update(Obsevable o, Object args)
SemaphorGUI
+update(Obsevable o, Object args)

UML-an termometroaren eta semaforoaren GUI-a gehitu ditugu. Eta ondoren GUI bakoitzari updatea jarri zaie.

2.2. Kode azpimagarria

- Covid19Pacient klasea:
  extend-a gehitzen da Obsever-ari notifikatzeko.

```
public class Covid19Pacient extends Observable {
…
}
```

- PacientObserverGUI klasea:
  Lehenik eta behin observer patroia implementatzeko aldaketak egin dira, CovidPacient-aren notifikazioak PacientObserver-ean jaso ahal izateko.

```
public class PacientObserverGUI extends JFrame implements Observer{

        private JPanel contentPane;
        private final JLabel symptomLabel = new JLabel("");
        public PacientObserverGUI(Observable obs) {
                obs.addObserver(this);
                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);
        }
        @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);
        }
}
```

- PacientSymptomGUI klasea:

  Ondorengo aldaketekin pazienteari sintomak gehitu edo kentzeko aukera ematen da, egoera automatikoki eguneratuz, eta horrela observer-ari notifikatuz.

```
public class PacientSymptomGUI extends JFrame {
	private JPanel contentPane;
	private JTextField weightField;
	JComboBox<Symptom> symptomComboBox;
	private JButton btnRemoveSymptom;
	private JLabel errorLabel;
	private JLabel lblPacient;
	private JLabel labelPacient;
	private Covid19Pacient p;
	/**
	 * Create the frame.
	 */
	public PacientSymptomGUI(Covid19Pacient p) {
		this.p=p;
		setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
		setBounds(200, 100, 450, 300);
		…
					//addSymptomByName ...
			p.addSymptomByName(((Symptom)
symptomComboBox.getSelectedItem()).getName()
,Integer.parseInt(weightField.getText()));
			} else errorLabel.setText("ERROR, Weight between [1..3]");
		    }
		});
		…
		System.out.println("Symptom removed : " + (Symptom)
symptomComboBox.getSelectedItem());

	p.removeSymptomByName( ((Symptom)
symptomComboBox.getSelectedItem() ).getName());
			}
		});
		btnRemoveSymptom.setBounds(255, 202, 147, 29);
		contentPane.add(btnRemoveSymptom);
		errorLabel = new JLabel("");
		errorLabel.setBounds(117, 146, 238, 16);
		contentPane.add(errorLabel);
		lblPacient = new JLabel("Pacient:");
		lblPacient.setBounds(210, 17, 61, 16);
		contentPane.add(lblPacient);
		labelPacient = new JLabel("New label");
		labelPacient.setFont(new Font("Lucida Grande", Font.BOLD, 13));
		labelPacient.setBounds(271, 17, 131, 16);
		labelPacient.setText(p.getName());
		contentPane.add(labelPacient);
		this.setVisible(true);
	}
}
```

- Main-a:

```
…
```

```
public static void main(String[] args) {
        Covid19Pacient pacient1 = new Covid19Pacient("aitor", 35);
         new PacientObserverGUI(pacient1);
         new PacientSymptomGUI(pacient1);
}
```

PacientSymptomGUI klaseak pazienteei sintomak gehitu eta kentzeko balio du, Covid19Pacient-ak Obvserver-ari notifikatzen dio eta azken honek datuak bistaratuko ditu. Ondoren termometroa inplementatu dugu.

- <u>PacientThermometerGUI klasea:</u>

PacientThermometerGUI-k Observer interfaz-a inplementatzen du, beraz, Observable bat behatzaile gisa funtzionatzen du, pazientearen egoerako aldaketak iragartzeko, Honetarako update funtzioa daukagu, eta honek observable motako aldagai bat jasotzen du GUI a automatiko eguneratzeko.

```
public class PacientThermometerGUI extends Frame implements Observer{
        private TemperatureCanvas gauges;
        /**
         * @wbp.nonvisual location=119,71
         */
        private final JLabel label = new JLabel("New label");

        public PacientThermometerGUI(Observable obs){

                super("Temperature Gauge");
                Panel Top = new Panel();
                . . .
                setVisible(true);
                obs.addObserver(this);
        }
        class TemperatureCanvas extends Canvas {
                public void set(int level) { current = level; }
                public int get(){return current;}
                public int getMax(){return Max;}
                public int getMin(){return Min;}
                private int Max, Min, current;
                public TemperatureCanvas(int min, int max){ Min = min; Max = max; }
                public void paint(Graphics g){
                  Color c;
                   . . .
                }

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

```
                    }
}
```

Azkenik linea hau gehitu da main ean termometroa agertzeko,

```
...
new PacientThermometerGUI(pacient1);
...
```

Azkenik semaforoa gehitu dugu. Honek ere Observer interfaz-a inplementatzen du.
Semaforoak funtzionatzeko covidImpact-aren arabera clorea aldatuko duen funtzionalitatea gehitu dugu updatean,honek termometroaren kasuan bezala observer bat jasotzen du GUI-a automatikoki eguneratzeko eta ondoren main-ean GUI-a sortu dugu.

- SemaphorGUI klasea:

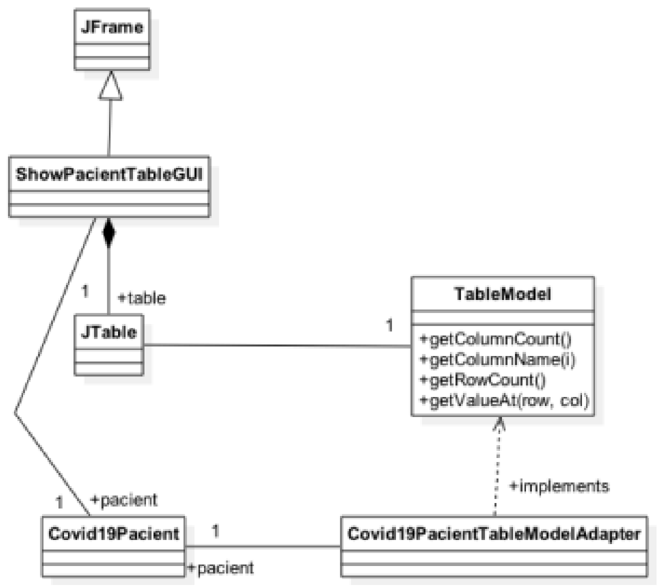
```
public class SemaphorGUI extends JFrame implements Observer {
          /** stores the associated ConcreteSubject */
          public SemaphorGUI (Observable obs) {
                    setSize(100, 100);
                    setLocation(350,10);
                    Color c=Color.green;
          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

          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();
          }
}
```

- Main:

```
...
new SemaphorGUI(pacient1);
...
```

3. Adapter Patroia

3.1. Diseinu hedatua, UML diagrama hedatua



3.2. Kode azpimagarria

- Covid19PacientTableModelAdapter klasea:
  getColumnCount(): zenbat zutabe dauden bidaltzen du, getRowCount(): zenbat lerro dauden, getColumnName(int i): zutabearen izena bidaltzen du, getValueAt(int row, int col): zutabe eta lerro horren balioa bidaltzen du, metodoak osatu ditugu.

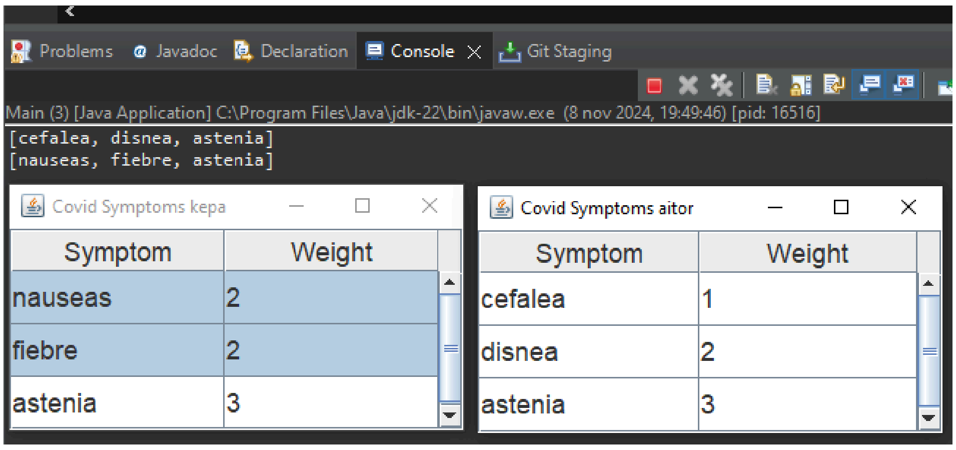
```
public class Covid19PacientTableModelAdapter extends AbstractTableModel {
        protected Covid19Pacient pacient;
        protected String[] columnNames = new String[] {"Symptom", "Weight" };

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

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

        public String getColumnName(int i) {
          return columnNames[i];
        }
        public int getRowCount() {
           return pacient.getSymptoms().size();
        }
        public Object getValueAt(int row, int col) {
           Iterator <Symptom> it= pacient.getSymptoms().iterator();
           Set<Symptom> symptomsSet = pacient.getSymptoms();
           Symptom symptom= null;
           for(int i= 0; i <= row && it.hasNext(); i++) {
                 symptom = it.next();
            }
```

```
                if (symptom != null) {
                        if (col == 0) {
                              return symptom.getName();
                        } else {
                              return pacient.getWeight(symptom);
                        }
                 }
                 return null;
           }
}
```

- Main klasea:

```
public class Main {

        public static void main(String[] args) {
                Covid19Pacient pacient1=new Covid19Pacient("aitor", 35);
                pacient1.addSymptomByName("disnea", 2);
                pacient1.addSymptomByName("cefalea", 1);
                pacient1.addSymptomByName("astenia", 3);
                System.out.println(pacient1.getSymptoms());
                ShowPacientTableGUI gui1=new ShowPacientTableGUI(pacient1);
                gui1.setPreferredSize(new java.awt.Dimension(300, 200));
                gui1.setVisible(true);

                Covid19Pacient pacient2=new Covid19Pacient("kepa",20);
                pacient2.addSymptomByName("fiebre", 2);
                pacient2.addSymptomByName("nauseas", 2);
                pacient2.addSymptomByName("astenia", 3);
                System.out.println(pacient2.getSymptoms());
                ShowPacientTableGUI gui2=new ShowPacientTableGUI(pacient2);
                gui2.setPreferredSize(new java.awt.Dimension(300, 200));
                gui2.setVisible(true);
        }
}
```

3.3. Emaitza probatzen



```
Main (3) [Java Application] C:\Program Files\Java\jdk-22\bin\javaw.exe (8 nov 2024, 19:49:46) [pid: 16516]
[cefalea, disnea, astenia]
[nauseas, fiebre, astenia]
```

Covid Symptoms kepa

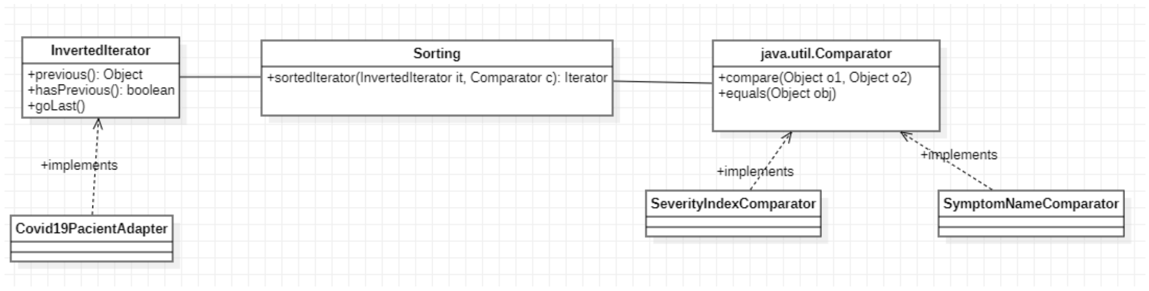
Symptom	Weight
nauseas	2
fiebre	2
astenia	3

Covid Symptoms aitor

Symptom	Weight
cefalea	1
disnea	2
astenia	3

4. Iterator Patroia

4.1. Diseinu hedatua, UML diagrama hedatua



Comparator erabiltzen duten SymptomNameComparator eta SeveretyIndexComparator klaseak sortu ditugu sorting-a erabili ahal izateko bi era ezberdinetara. Bestalde, Covid19PacientAdapter adaptadorea sortu dugu InvertedIterator implemetatu ahal izateko Iterator erabilita.

4.2. Kode azpimagarria

- SymptomNameComparator klasea:
  Sintomaren izenarekiko ordenatzen ditu comparator erabiliz.

```
public class SymptomNameComparator implements Comparator<Object> {
        @Override
        public int compare(Object o1, Object o2) {
                Symptom s1 = (Symptom) o1;
                Symptom s2 = (Symptom) o2;
                return s1.getName().compareTo(s2.getName());
        }
}
```

- SeveretyIndexComparator klasea:
  Severity Index-arekiko ordenatzen ditu comparator erabiliz.

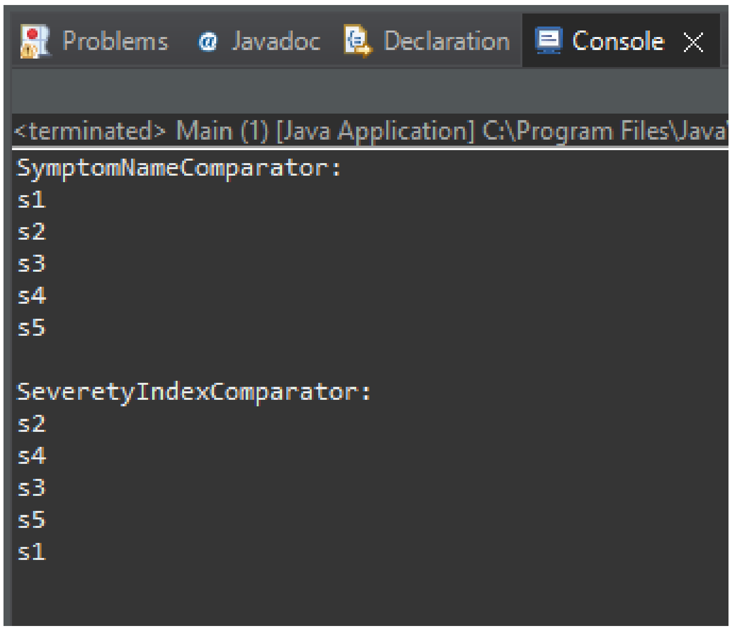
```
public class SeveretyIndexComparator implements Comparator<Object>{
        @Override
        public int compare(Object o1, Object o2) {
                Symptom s1 = (Symptom) o1;
                Symptom s2 = (Symptom) o2;
                return Integer.compare(s1.getSeverityIndex(), s2.getSeverityIndex());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

- Covid19PacientAdapter klasea:
  Klase honetan adpater erabiltzen da InvertedIterator interfazea erabiltzeko Covid19Pacient-ek Iterator erabiltzen duenean, hau da, Covid19Pacient InvertedIterator nola funtzionatzen duen jakin gabe Covid19PacientAdapter erabiliz aldaketa hori egiten da.

```
public class Covid19PacientAdapter implements InvertedIterator {
   private List<Symptom> symptoms;
   private int position;
```

```
    public Covid19PacientAdapter(Set<Symptom> symptomSet) {
        this.symptoms = new Vector<>(symptomSet);
        this.position = symptoms.size();
    }

    @Override
    public Symptom previous() {
        if (!hasPrevious()) {
            throw new IndexOutOfBoundsException("No hay elementos anteriores");
        }
        position--;
        return symptoms.get(position);
    }

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

    @Override
    public void goLast() {
        this.position = symptoms.size();
    }
    public boolean hasNext() {
        return position < symptoms.size();
    }
    public Symptom next() {
        if (!hasNext()) {
            throw new IndexOutOfBoundsException("No hay elementos siguientes");
        }
        return symptoms.get(position++);
    }
}
```

4.3. Emaitza probatzen



```
<terminated> Main (1) [Java Application] C:\Program Files\Java
SymptomNameComparator:
s1
s2
s3
s4
s5

SeveretyIndexComparator:
s2
s4
s3
s5
s1
```