eman ta zabal zazu

Universidad del País Vasco Euskal Herriko Unibertsitatea

INFORMATIKA FAKULTATEA
FACULTAD DE INFORMÁTICA

LAB PATTERNS LABORATEGIA

Software ingeniaritza

Joritz Arocena eta Mikel Carrasco

25-26 ikasturtea

AURKIBIDEA

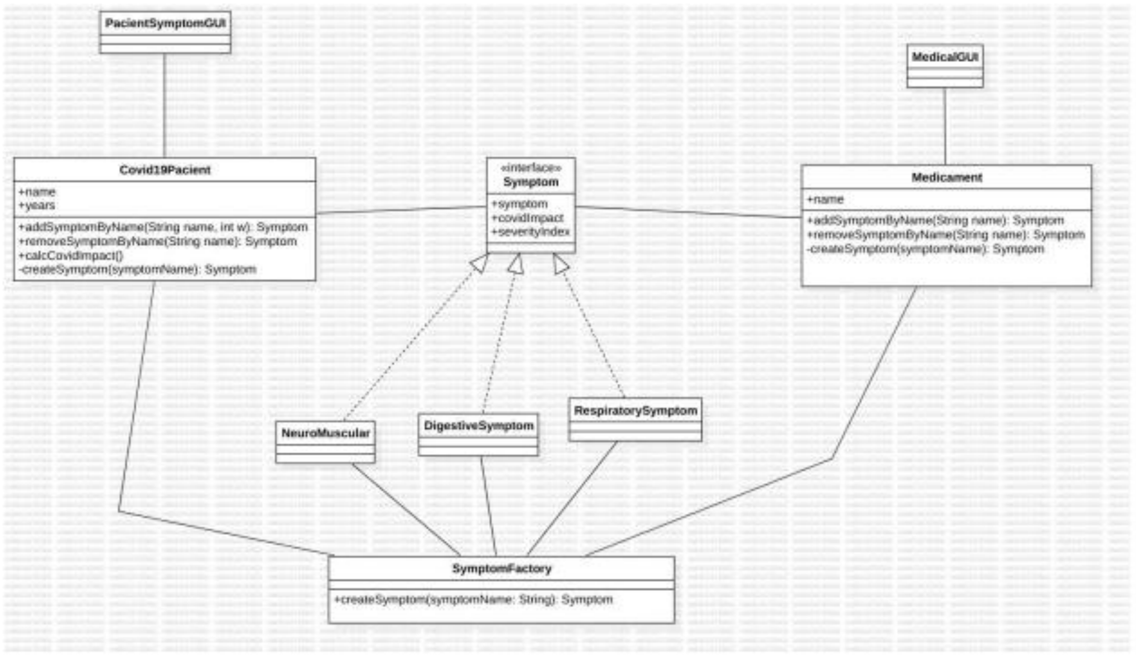
SARRERA

Patroiek buruzko laborategiari buruzko dokumentua da honako hau. Bertan sortutako galderen erantzunak eta bai kode zatiak bai uml argazkiak aurkituko dituzu berta. Garatutako kodea honako esteka honetan aurkitu dezakezu: https://github.com/jarocena013/labpatterns

Egileak: Mikel Carrasco Tajo eta Joritz Arocena Zuriarrain

SIMPLE FACTORY

Ariketa honetan uml diagrama aldatu da eta honela gelditu da:

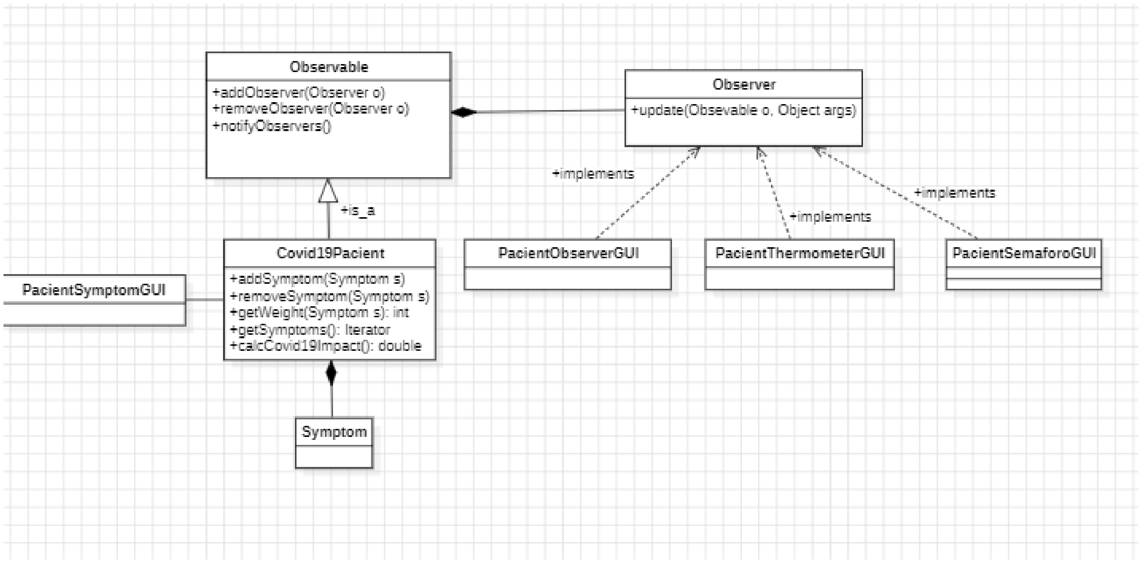

PacientSymptomGUI
MedicalGUI
Covid19Pacient
+name
+years
+addSymptomByName(String name, int w): Symptom
+removeSymptomByName(String name): Symptom
+calcCovidImpact()
-createSymptom(symptomName): Symptom
«interface»
Symptom
+symptom
+covidImpact
+severityIndex
Medicament
+name
+addSymptomByName(String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

Kasu honetan egin dugun aldaketa da createSymptom metodoa SymptomFactory klase batera pasatu dugu. Horrela klase guztiak sortu berri dugun klase hau erabiliko dugu eta aldaketaren bat egin behar bada bertan egingo ditugu banan banan gauzak aldatzen hasi beharrean.

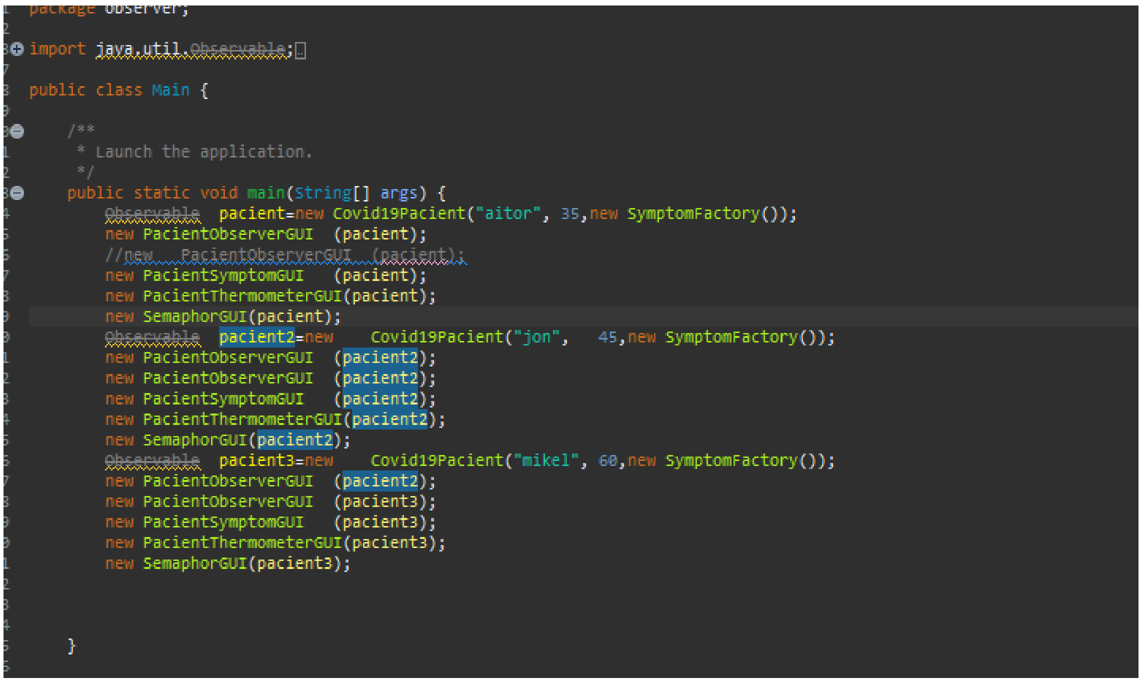
Jadanik sortuta dagoen sintoma bat berriro ere ez sortzeko aukera Nagusia, singleton patroia erabiltzea izango litzateke. Horrela sortzen den guztietan instantzia bera erabiliko litzateke eta bertan sintoma bereko mota guztiak gordeko lirateke.

OBSERVER PATROIA

Patroi hau egiteko honako UML diagrama lortu dugu:



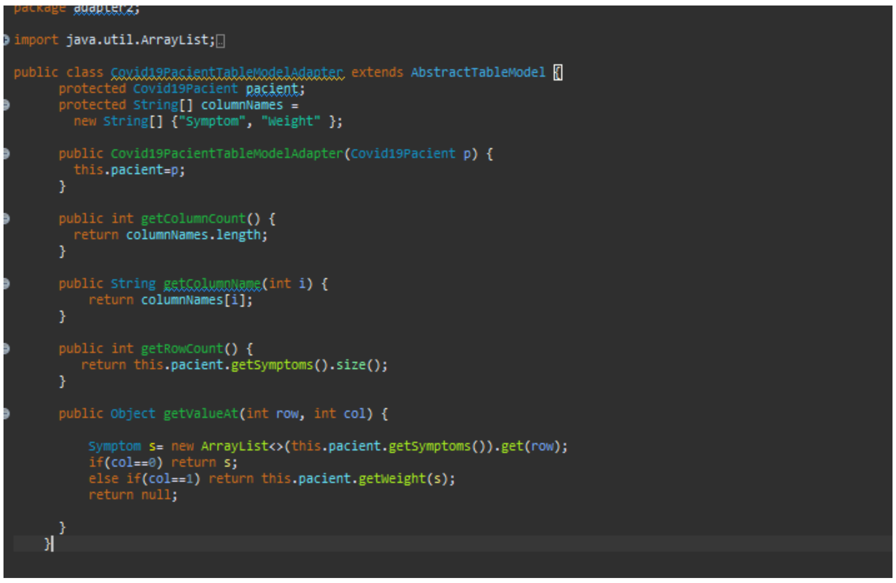
Honela gelditu da main zatiaren kodea:



```
package observer;

import java.util.Observable;

public class Main {

    /**
     * Launch the application.
     */
    public static void main(String[] args) {
        Observable  pacient=new Covid19Pacient("aitor", 35,new SymptomFactory());
        new PacientObserverGUI  (pacient);
        //new  PacientObserverGUI  (pacient);
        new PacientSymptomGUI   (pacient);
        new PacientThermometerGUI(pacient);
        new SemaphorGUI(pacient);
        Observable  pacient2=new    Covid19Pacient("jon",   45,new SymptomFactory());
        new PacientObserverGUI  (pacient2);
        new PacientObserverGUI  (pacient2);
        new PacientSymptomGUI   (pacient2);
        new PacientThermometerGUI(pacient2);
        new SemaphorGUI(pacient2);
        Observable  pacient3=new    Covid19Pacient("mikel", 60,new SymptomFactory());
        new PacientObserverGUI  (pacient2);
        new PacientObserverGUI  (pacient3);
        new PacientSymptomGUI   (pacient3);
        new PacientThermometerGUI(pacient3);
        new SemaphorGUI(pacient3);



    }
```

ADAPTER PATROIA

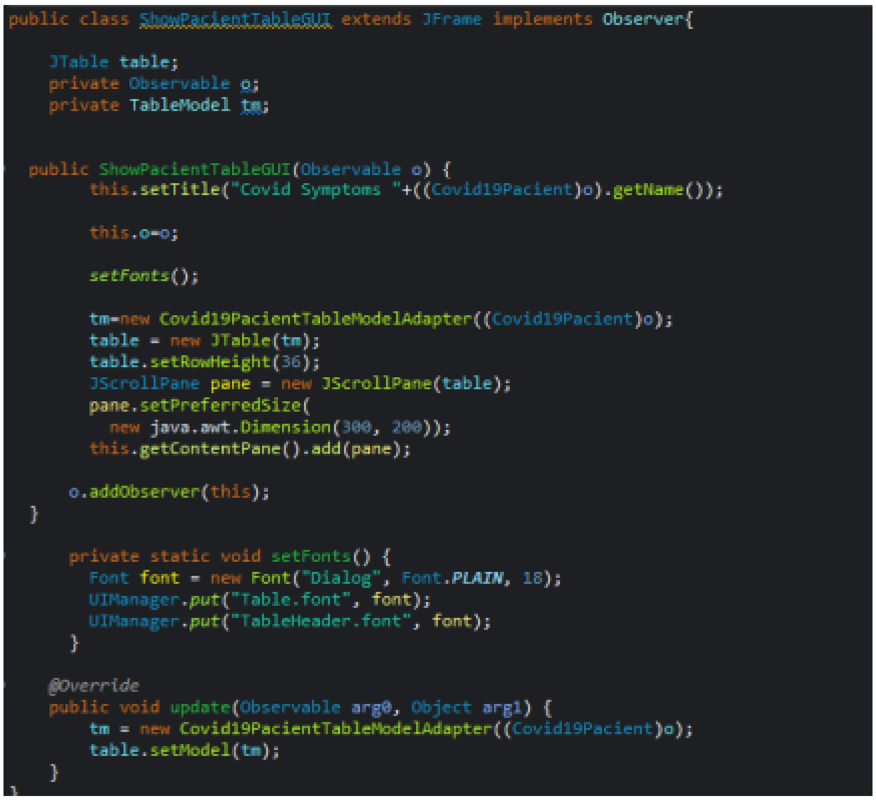
Patroi hau jarraitzeko Covid19PacientTableModelAdapter klasean zenbait aldaketa egin ditugu eta hórrela gelditu da:



```java
package adapter2;

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;
      }

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

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

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

      public Object getValueAt(int row, int col) {

          Symptom s= new ArrayList<>(this.pacient.getSymptoms()).get(row);
          if(col==0) return s;
          else if(col==1) return this.pacient.getWeight(s);
          return null;

      }
    }
```

Aurreko observer ariketan hau gehitzeko lehendabizi observer interfazea implementatu behar dugu. Ondoren Observable objektu bat sortu eraikitzailean eta azkenik observablean aldaketa bat sortzen den bakoitzen gure taulako adapter bat sortu eta taulari adapter hori pasatuko diogu.

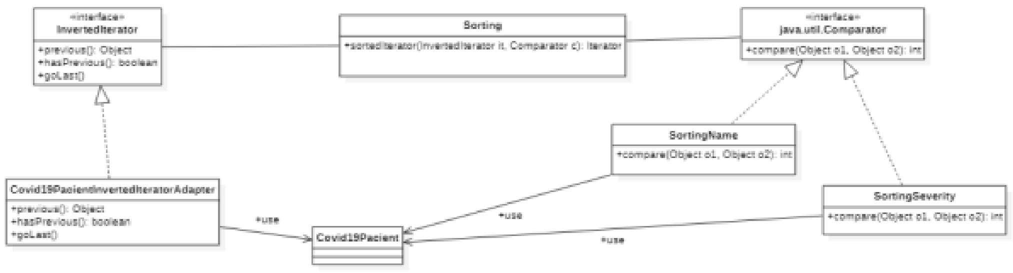
Honela geldituko litzateke kodea:



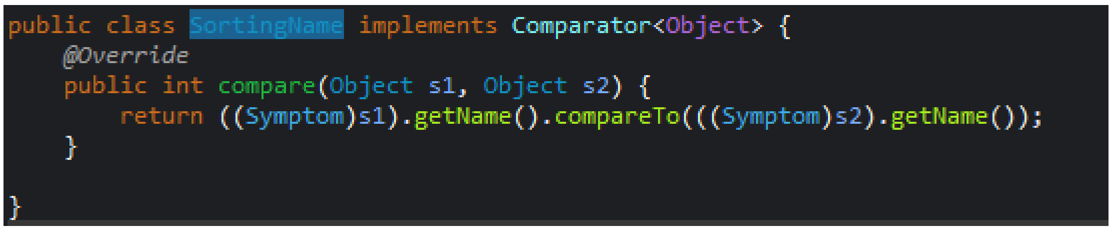
```java
public class ShowPacientTableGUI extends JFrame implements Observer{

    JTable table;
    private Observable o;
    private TableModel tm;


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

        this.o=o;

        setFonts();

        tm=new Covid19PacientTableModelAdapter((Covid19Pacient)o);
        table = new JTable(tm);
        table.setRowHeight(36);
        JScrollPane pane = new JScrollPane(table);
        pane.setPreferredSize(
          new java.awt.Dimension(300, 200));
        this.getContentPane().add(pane);

      o.addObserver(this);
  }

      private static void setFonts() {
        Font font = new Font("Dialog", Font.PLAIN, 18);
        UIManager.put("Table.font", font);
        UIManager.put("TableHeader.font", font);
      }

    @Override
    public void update(Observable arg0, Object arg1) {
        tm = new Covid19PacientTableModelAdapter((Covid19Pacient)o);
        table.setModel(tm);
    }
}
```

ADAPTER ETA ITERATOR PATROIAK

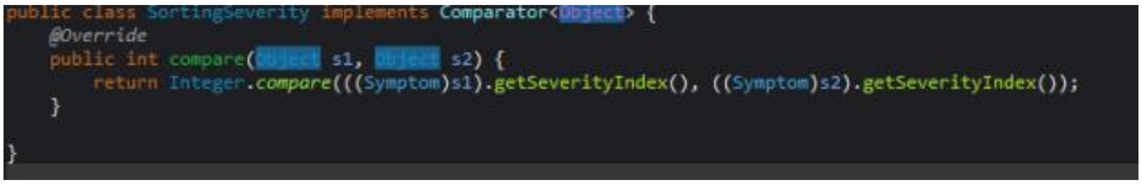
Ariketa honen UML-a honela geldituko litzateke:



Bi klase berri sortu ditugu bakoitzean behar den ordenean ordenatzeko eta bi klaseek Comparator interfazea implementatuko dute:

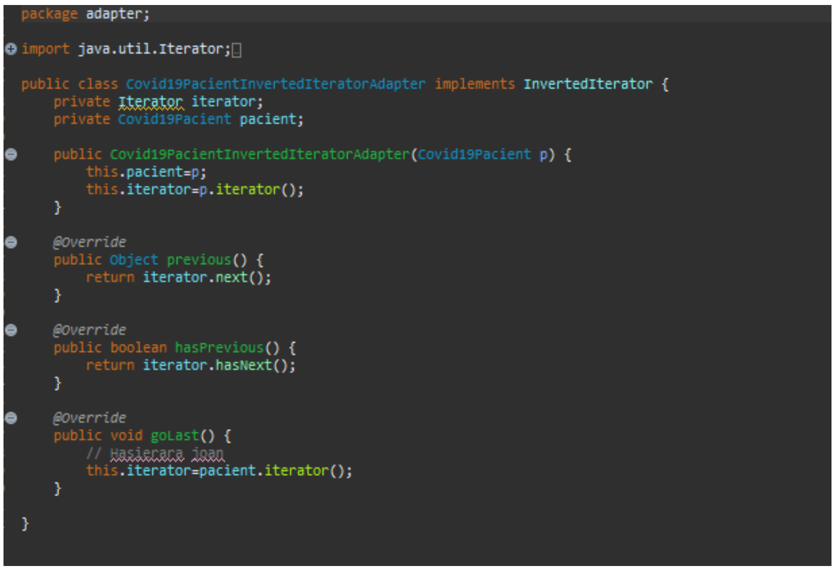


```
public class SortingName implements Comparator<Object> {
    @Override
    public int compare(Object s1, Object s2) {
        return ((Symptom)s1).getName().compareTo(((Symptom)s2).getName());
    }

}
```



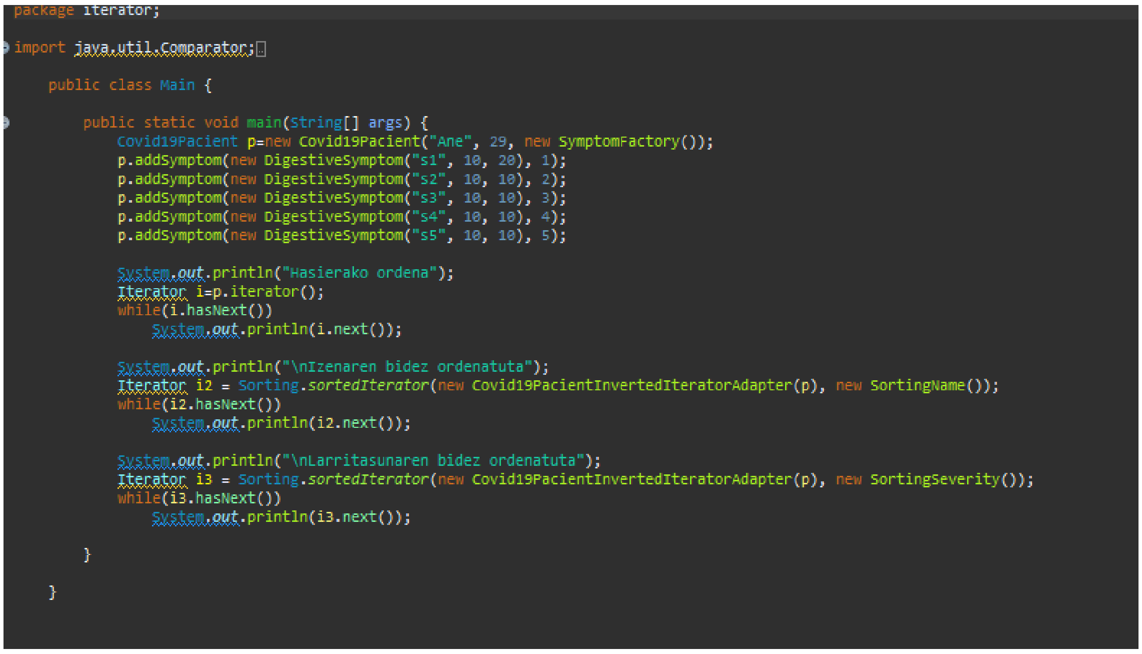
```
public class SortingSeverity implements Comparator<Object> {
    @Override
    public int compare(Object s1, Object s2) {
        return Integer.compare(((Symptom)s1).getSeverityIndex(), ((Symptom)s2).getSeverityIndex());
    }

}
```

Gainera klase berri bat sortu dugu Covid19Pacientek Inverted iterator erabili dezan. Ondorengo kodea lortu dugu:

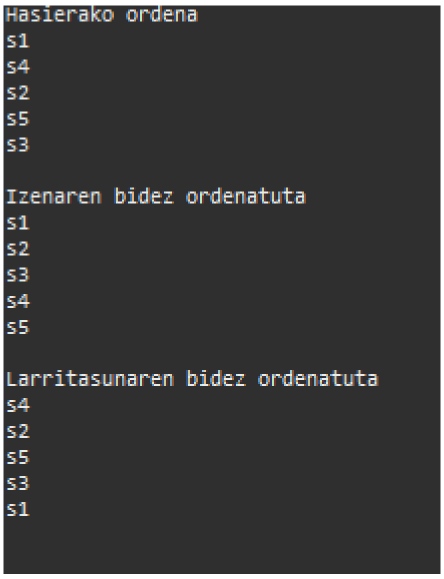


```
package adapter;

import java.util.Iterator;

public class Covid19PacientInvertedIteratorAdapter implements InvertedIterator {
    private Iterator iterator;
    private Covid19Pacient pacient;

    public Covid19PacientInvertedIteratorAdapter(Covid19Pacient p) {
        this.pacient=p;
        this.iterator=p.iterator();
    }

    @Override
    public Object previous() {
        return iterator.next();
    }

    @Override
    public boolean hasPrevious() {
        return iterator.hasNext();
    }

    @Override
    public void goLast() {
        // Hasierara joan
        this.iterator=pacient.iterator();
    }

}
```

Hona hemen programa nagusiaren kodea eta lortutako emaitzak:



```java
package iterator;

import java.util.Comparator;

    public class Main {

        public static void main(String[] args) {
            Covid19Pacient p=new Covid19Pacient("Ane", 29, new SymptomFactory());
            p.addSymptom(new DigestiveSymptom("s1", 10, 20), 1);
            p.addSymptom(new DigestiveSymptom("s2", 10, 10), 2);
            p.addSymptom(new DigestiveSymptom("s3", 10, 10), 3);
            p.addSymptom(new DigestiveSymptom("s4", 10, 10), 4);
            p.addSymptom(new DigestiveSymptom("s5", 10, 10), 5);

            System.out.println("Hasierako ordena");
            Iterator i=p.iterator();
            while(i.hasNext())
                System.out.println(i.next());

            System.out.println("\nIzenaren bidez ordenatuta");
            Iterator i2 = Sorting.sortedIterator(new Covid19PacientInvertedIteratorAdapter(p), new SortingName());
            while(i2.hasNext())
                System.out.println(i2.next());

            System.out.println("\nLarritasunaren bidez ordenatuta");
            Iterator i3 = Sorting.sortedIterator(new Covid19PacientInvertedIteratorAdapter(p), new SortingSeverity());
            while(i3.hasNext())
                System.out.println(i3.next());

        }

    }
```



```
Hasierako ordena
s1
s4
s2
s5
s3

Izenaren bidez ordenatuta
s1
s2
s3
s4
s5

Larritasunaren bidez ordenatuta
s4
s2
s5
s3
s1
```