Software Ingeniaritza II

LABPATTERN lab

Iván González

Beñat Arruti

Mikel Berlanga

Eneko Santos

Simple Factory

Implementazioa:

ISymptom interfazean:

```
public interface ISymptom {

String getName();

void setName(String name);

int getCovidImpact();

void setCovidImpact(int covidImpact);

int getSeverityIndex();

void setSeverityIndex(int severityIndex);

String toString();

}
```

Symptom klasean:

```
public class Symptom implements ISymptom {
```

SymptomFactory klasean:

```
public class SymptomFactory {

public SymptomFactory() {
}

public Symptom createSymptom(String symptomName) {
                                                    .
                                                    .
                                                    .
                                                    .
}
```

Interfaze bat sortu eta symptom erabiltzen dituen metodoak hor jarri beste klase sortzen bada eta metodo horiek erabili nahi badu ez egoteko problemarik.

Observer Patroia

```
public class Covid19Pacient extends Observable {

}
```

```
public void addSymptom(Symptom c, Integer w){
symptoms.put(c,w);
// notify observers that state changed
setChanged();
notifyObservers(c);
}

public ISymptom addSymptomByName(String symptom, Integer w){
Symptom s=null;
s=factory.createSymptom(symptom);
if (s!=null) {
symptoms.put(s,w);
// notify observers that state changed
setChanged();
notifyObservers(s);
}
```

```
}


public class PacientObserverGUI extends JFrame implements Observer {
}

@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);
}
```

```
p.addSymptomByName(((Symptom)symptomComboBox.getSelectedIte
m()).getName(), Integer.parseInt(weightField.getText()));
```

```
p.removeSymptomByName(((Symptom)symptomComboBox.getSelected
Item()).getName());
```

Laboratian dauden pausuak segi behar dira kodea ere hor dago eta hemen dago kode garrantzitsuena.

Adapter Patroia

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

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

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

 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!
	Set<Symptom> symptoms = pacient.getSymptoms();
	Object[] s = symptoms.toArray();
	if(col == 0) return ((Symptom)s[row]).getName();
	if(col == 1) return pacient.getWeight((Symptom)s[row]);
	return null;
	}
}
```

Aplikazio hau garatzeko, Swing paketearen JTable klase grafikoa erabili dugu. JTable klaseak TableModel objektu batean dagoen informazioa taula batean argitaratzen du. Gure kasuan, Covid19Pacient baten sintomak argitaratu nahi ditugu, beraz, klase egokitzaile bat sortu beharko dugu, Covi19Pacient objektu bat TableModel objektu batera egokitzeko, hau da, Covid19PacientTableModelAdapter. Klase horretan lau metodo daude eta metodo bakoitzak taula eraikitzen dute.

Adapter eta Iterator

Implementazioa:

InvertedIterator:

```
public class InvertedIteratorCovid19PacientAdapter implements
InvertedIterator {

List<Symptom> symptoms=new Vector<Symptom>();

int position=0;


public InvertedIteratorCovid19PacientAdapter(Set<Symptom> s) {

Iterator<Symptom> i=s.iterator();
```

```
while (i.hasNext())
symptoms.add(i.next());

}

@Override
public Object previous() {
position--;
Symptom symptom=symptoms.get(position);
return symptom;

}

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

@Override
public void goLast() {
position=symptoms.size();
}

}
```

Covid19PacientIterator-eko kodea hartu eta invertedIteratorrera egokitu dut.

severityIndexComparator:

```
public class severityIndexComparator implements Comparator<Object>{
```

```
@Override
public int compare(Object o1, Object o2) {
ISymptom s1 = (ISymptom) o1;
ISymptom s2 = (ISymptom) o2;
if (s1 == null && s2 == null) return 0;
if (s1 == null) return -1;
if (s2 == null) return 1;
return s1.getSeverityIndex() - s2.getSeverityIndex();
}

}
```

Bi sintoma hartu eta severity Index arabera konparatzen da, zenbakien konparazio moduan

SymptomNameComparator:

```
public class symptomNameComparator implements Comparator<Object>{

@Override
public int compare(Object o1, Object o2) {
ISymptom s1 = (ISymptom) o1;
ISymptom s2 = (ISymptom) o2;
if (s1 == null && s2 == null) return 0;
if (s1 == null) return -1;
if (s2 == null) return 1;
return s1.getName().compareTo(s2.getName());
}
}
```

Syptom objektuak hartu eta beren izenen arabera konparatzen da, string konparazioa erabiliz.