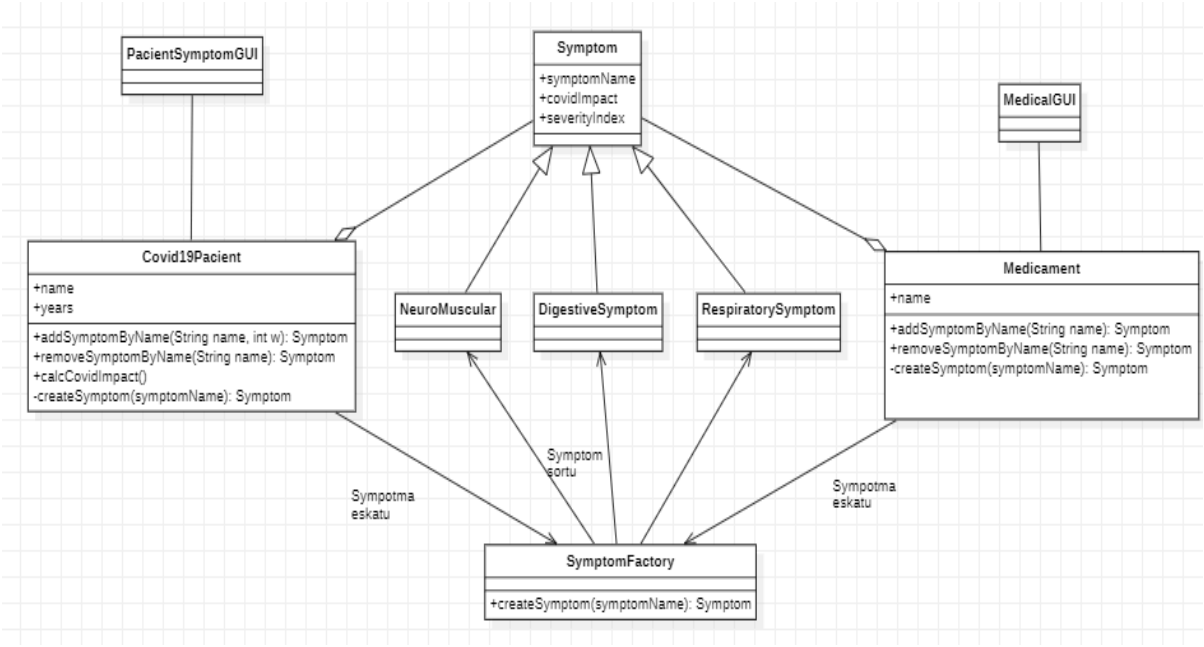
Diseinu Patroiak

Github: https://github.com/cianaranburu/labpatterns

1. Factory

1.1. SymptomFactory klasea sortu. Sintomak zituzten loturak medikamendukin eta pazientekin kendu ta SymptomFactory klasera lotu. Medikamentua eta Pazienteak Fabrika erabiliko dute sintoma berriak sortzeko.



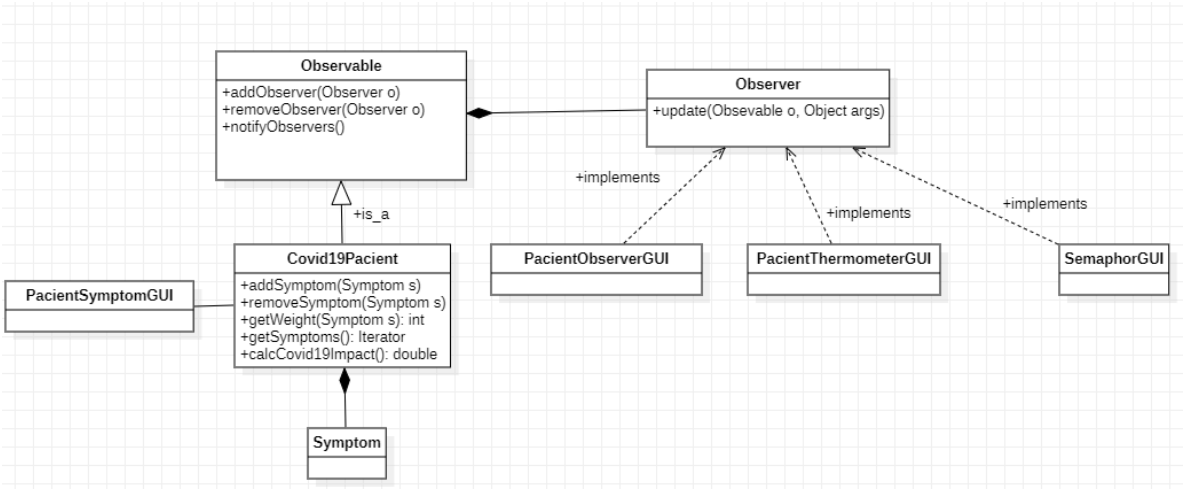
1.2 Mareo sintoma impact1 listara eta neuroMuscular listan gehitzen dugu.

1.3 SymptomFactory klasean aldaketa bat egin dugu sisteman sintoma bakoitzaren objektu bakarra egon dadin. Hau lortzeko, Map<String, Symptom> motako aldagaia gehitu dugu, non gako bezala sintomaren izena eta balio bezala bere objektua gordetzen diren.

createSymptom metodoa deitzen den bakoitzean, lehenik begiratzen du ea sintoma hori Map horretan dagoen. Aurretik sortua badago, objektua zuzenean itzultzen da. Ez badago, sortu egiten da dagokion klasearekin (DigestiveSymptom, NeuroMuscularSymptom edo RespiratorySymptom), Map-ean gordetzen da eta gero itzultzen da.

2. Obsever

2.1. UML diagraman egindako aldaketak



3. Adapter

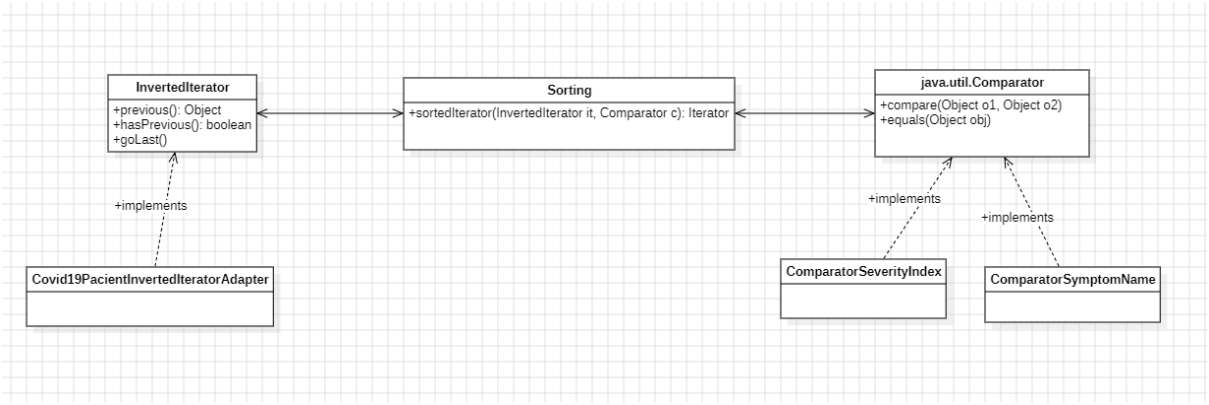
getColumnCount, getColumnName eta getRowCownt getter sinple batzuk dira, ez da logika berririk sortu behar izan. getValueAt metodorako ordea bai.

```
public Object getValueAt(int row, int col) {
    Covid19PacientIterator it = (Covid19PacientIterator) pacient.iterator();
    ArrayList<Symptom> symptomList= new ArrayList<Symptom>();
    while (it.hasNext()) {
        symptomList.add((Symptom) it.next());

    }

    if (col == 0) {
        return symptomList.get(row).getName();
    } else if (col==1) {
        return pacient.getWeight(symptomList.get(row));
    }
    return null;
}
```

getValueAt funtzioan Covid19PacientIterator erabili da HashMapa iteragarria den ArrayList batean bihurtzeko eta gero aukeratutako zutabearen arabera lerro horri dagokion sintoma edo pisua erakusten da.

4. Adapter Iterator

Sortutako UML diagrama:



Covid19PacientInvertedIteratorAdapter klasea sortu dugu sorting metodoan objetu bezala pasatzeko Covid19Pacientren sintomekin, eta ComparatorSeverityIndex eta ComparatorSymptomName klaseak sortu dugu sintomen listak ordenatzeko, metodoen implementazioa:

```
public class ComparatorSymptomName implements Comparator {

    @Override
    public int compare(Object arg0, Object arg1) {
        return ((Symptom) arg0).getName().compareTo(((Symptom) arg1).getName());
    }



}
```

metodoa bi symtom objetu jasota beraien izenaren arabera konparatzen ditu compareTo metodoari deituz, eta zenbaki positibo negatibo edo zero itzultzen du symtomaren izena alfabetikoki handiagoa, txikiagoa edo berdina bada.

```
public class ComparatorSeverityIndex implements Comparator {

    @Override
    public int compare(Object o1, Object o2) {

        return ((Symptom) o1).getSeverityIndex()-(((Symptom) o2).getSeverityIndex());
    }


}
```

metodoa bi symtom objetu jasota beraien SeverityIndexaren arabera zenbaki positibo negatibo edo zero itzultzen du bere SeverityIndex balioen arteko kenketa eginez.