HEALTH TRACKER

CSCI 5448 : GROUP #21

PRITHVI MANIKONDA

PRIYANKA PASHTE

SHREYA JOSHI

PROJECT SUMMARY

- A Health Monitoring Web Application that allows the users to monitor their health parameters.
- Allows two types of users – Doctors and Patients
- Allows Patients to select/deselect an Advising Doctor
- Book, Reschedule or Cancel appointment with Doctor

USE CASES # 1 : SELECT DOCTOR

- Allows the Patient to select Advising Doctor
- Patient can search doctors based on multiple dropdown lists namely Specialization and Location.
- The Patient can select the sort preference for the results to be displayed
- The Patient can then select one as the Advising Doctor
- The Doctor is successfully associated with the Patient.

USE CASES # 2 : VIEW PATIENT'S PROFILE

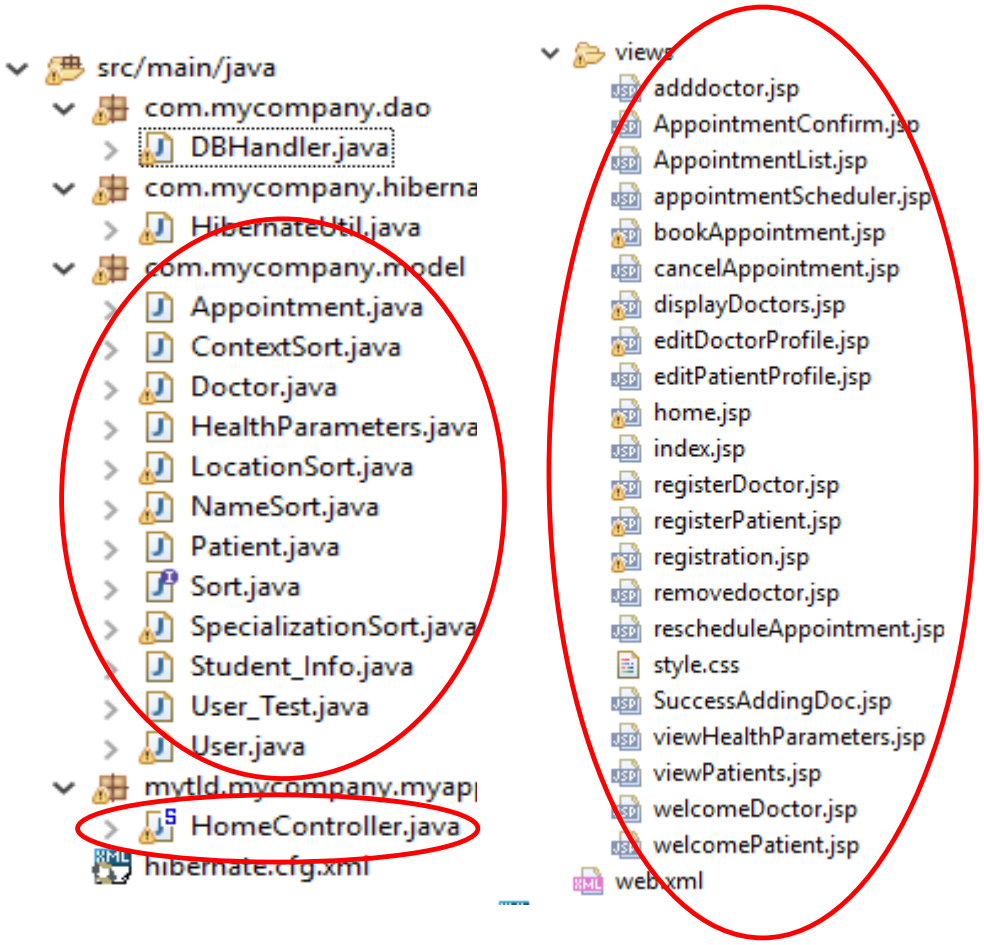
- Allows the Doctor to view his Advising Patients
- Doctor can view his Advising Patients.
- The Doctor can select a patient to see the Patient's profile.
- The Patient's basic information is displayed.

USE CASES DEMO

- http://screencast.com/t/sJqNn5fnmI9n

SPRING MVC

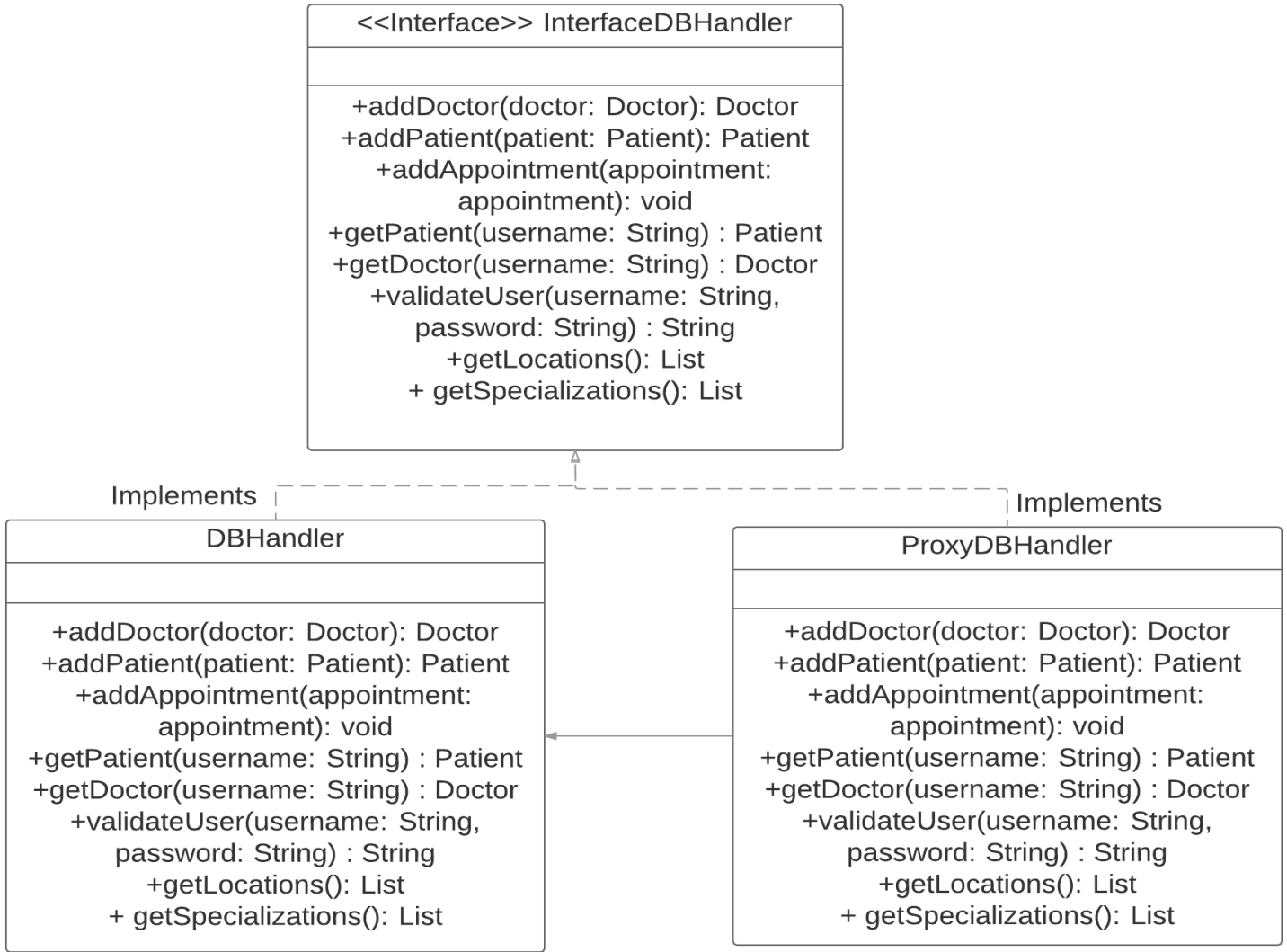
- Separation of concerns between controllers, JavaBean models, and views.
- **Controller**
  - Responsible for preparing a model Map with data and selecting a view name
- **Model**
  - Encapsulates the application data and they will consi of POJO's
- **View**
  - Data binding through powerful JSP tag library known the Spring tag library



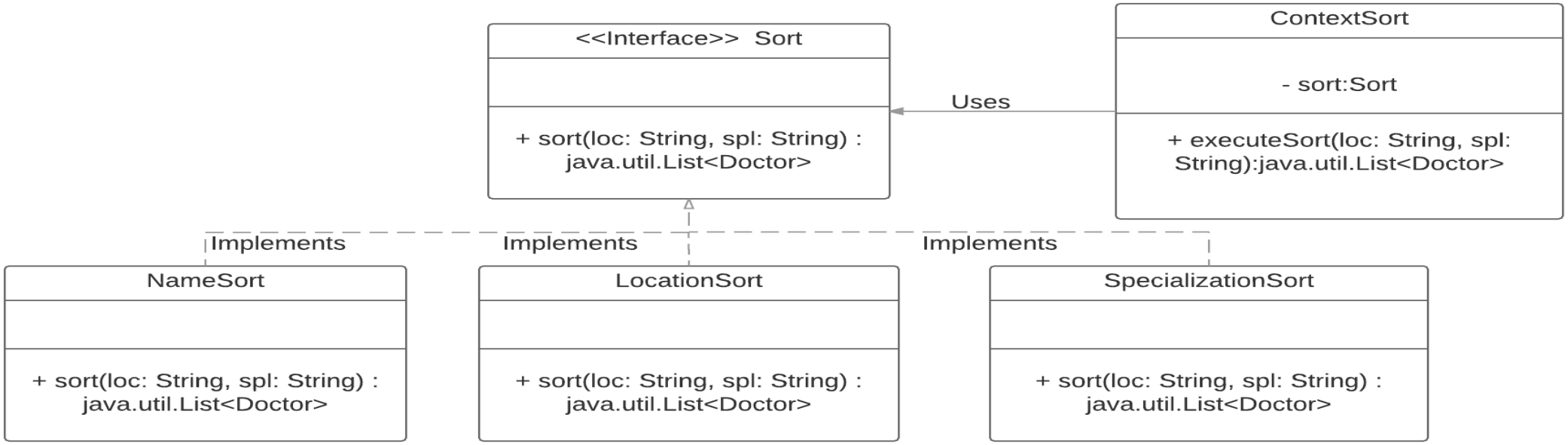
HIBERNATE MAPPINGS & ANNOTATIONS

- **Annotations:**
  - Newest way to define mappings without a use of XML file
  - All the metadata is clubbed into the POJO java file along with the code
  - **Annotations Used:**
    - @Id, @Generated Value, @Entity, @MapperSuperclass
    - @Inheritance(strategy=InheritanceType.***TABLE_PER_CLASS***)
- **Association Mappings:**
  - Mapping of associations between entity classes and the relationships between tables
  - @OnetoOne, @ManytoMany, @OnetoMany, @ManytoOne

PROXY DESIGN PATTERN



STRATEGY DESIGN PATTERN



REFACTORING

```
public Patient getPatient(String username){
    SessionFactory sessionFactory = HibernateUtil.getSessionFactory();
    Session session = sessionFactory.openSession();
    session.beginTransaction();
    String queried="from Patient P where P.userName = :username" ;
    Query query=  session.createQuery(queried);
    query.setParameter("username",username);
    java.util.List<Patient> patient = query.list();
    return patient.get(0);
}
public Doctor getDoctor(String username){
    SessionFactory sessionFactory = HibernateUtil.getSessionFactory();
    Session session = sessionFactory.openSession();
    session.beginTransaction();
    String queried="from Doctor D where D.userName = :username" ;
    Query query=  session.createQuery(queried);
    query.setParameter("username",username);
    java.util.List<Doctor> doctor = query.list();
    return doctor.get(0);
}
```

```
public Session getDBsession(){
    SessionFactory sessionFactory = HibernateUtil.getSessionFactory();
    Session session = sessionFactory.openSession();
    session.beginTransaction();
    return session;
}
public Patient getPatient(String username){
    Session session = getDBsession();
    String queried="from Patient P where P.userName = :username" ;
    Query query=  session.createQuery(queried);
    query.setParameter("username",username);
    java.util.List<Patient> patient = query.list();
    return patient.get(0);
}
```

FUTURE DEVELOPMENT

- Iterator Design Pattern
- Further Refactoring

FULL DEMO

- https://github.com/priyankapashte/CSCI5448_HealthTracker/blob/master/HealthTracker_video.mp4

THANK YOU