Design patterns

Design patterns used in the project

Singleton:

- **Description:** Only one instance of a class is used everywhere.
- **In this project:** Beans created by Spring framework which is used in this project

Factory:

- **Description:** Get objects of different classes from same method by passing different parameter.
- **In this project:** java.lang.Class.forName(JDBC_DRIVER);

Iterator pattern

- **Description:** Access the elements of a collection object in sequential manner without any need to know its underlying representation.
- **In this project:** For-in loops, for example: “for (SubjectAttributeType type : subjectAttributeTypeList)”

Observer

- **Description:** If one object is modified, its depenedent objects are to be notified automatically.
- **In this project:** AngularJS, which is used in this project, provides built-in observer pattern for data-binding between view and model.

Composite (not composite view)

- **Description:** Composite pattern is used where we need to treat a group of objects in similar way as a single object.
- **In this project:** Wrapper classes (EmployeeListWrapper, EntPerRelationTypeListWrapper etc)

Adapter

- **Description:** Adapter pattern works as a bridge between two incompatible interfaces.
- **In this project:** CurrentUser class between UserAccount and UserDetails.