Notes:

1. I have coded the DAO using basic spring JDBC template and not used any of the ORM frameworks since the example association can be easily managed,
2. The same can be achieved using hibernate, I believe hibernate is an over kill for small applications with fewer domain and fewer dependencies between them. However, for complex domain models involving associations hibernate is helpful
3. Once the mapping is done right it is easy to save object and its dependencies using simple

method like template.save(student) saves the entire object graph(meaning student and all associations like courses).

1. Hibernates provides caching for faster retrievals. Session is the first level cache. It optimizes all the changes to an object once so that the updates to the state of an object are synchronized effectively
2. It supports versioning which is not possible using JDBC API
3. Hibernate supports hibernate query language which operates on objects and their properties. HQL is intuitive from application domain model
4. Hibernate also scales well with application