Todo IT – part 3

Topics:

- **Collections**
- **Interfaces**
- **DAO pattern**
- **Sequencers**
- Stream API - optional
- Singleton pattern – optional
- Generics – optional
- Serialization to JSON - optional

Overview:

In this project you need to implement **central storage** of your model objects. (**AppUser**, **Person**, **TodoItem** and **TodoItemTask**) You are going to do this by applying the **DAO pattern** with help of **Collections**. You are also going to make **sequencers** in order to be able to pass unique id's to each new object.

Requirements:

- Sequencers created according to specific requirements
- Interfaces **AppUserDAO.class**, **PersonDAO.class**, **TodoItemDAO.class** and **TodoItemTaskDAO.class** implemented according to each specific requirement

If you have time:

- Unit testing

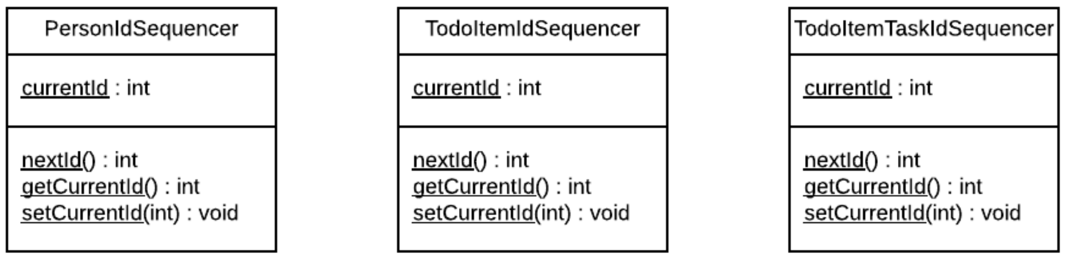
Optional:

- Use singleton pattern for DAO implementing classes and sequencers
- Use Generics to simplify DAO interfaces
- Use the Stream API in implementing DAO classes
- Read / Write all objects from / to JSON file(s) when starting / ending application.
- Read / Write all sequencervalues from / to a properties file when starting / ending application.

Good Luck!

Sequencers:

Your sequencers know which id is the next available id for each new object you create. Here we are making the sequencers static. If you don’t want your sequencers to be static it would be best to follow the singleton pattern. (Opens up many possibilities like using inheritance and polymorphism)
Put your sequencers in a new package.



Create the following three classes. All underlined fields and methods are static.

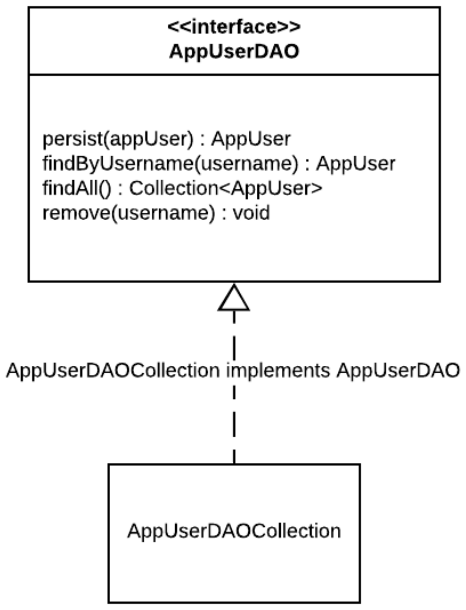
- **currentId**: is an integer that acts as a counter.
- **nextId()** : increment currentId and then return currentId.
- **getCurrentId()** : returns currentId.
- **setCurrentId()** : sets currentId.

Data Access Objects (DAOs):

AppUserDAO implementation:

State: Any collection of type AppUser
Constructor(s): Up to you
Methods:

- **persist**: add new *AppUser.class* object to collection
- **findByUsername**: returns single *AppUser.class* object
- **findAll**: returns all *AppUser.class* objects
- **remove**: remove one *AppUser.class* object from collection

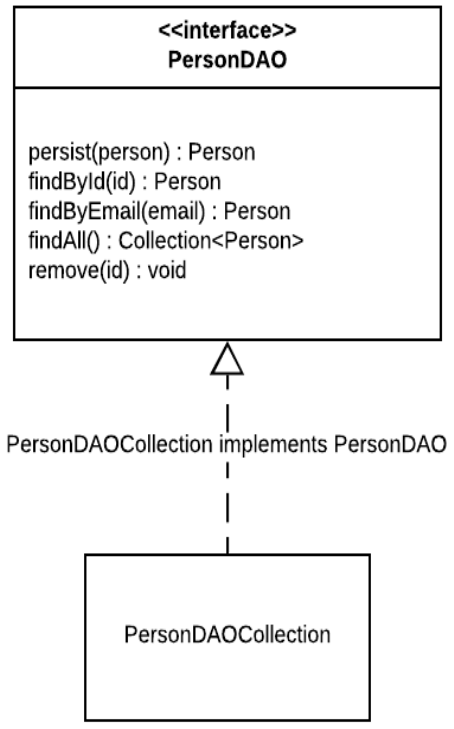


<<interface>>
AppUserDAO
persist(appUser) : AppUser
findByUsername(username) : AppUser
findAll() : Collection<AppUser>
remove(username) : void
AppUserDAOCollection implements AppUserDAO
AppUserDAOCollection

PersonDAO implementation:

State: Any collection of type Person
Constructor(s): Up to you
Methods:

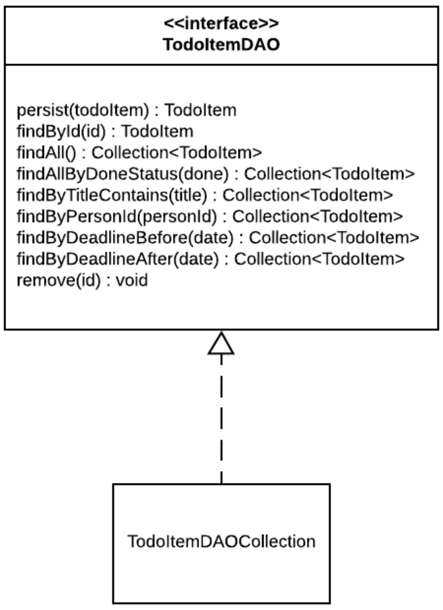
- **persist**: add new *Person.class* object to collection
- **findById**: returns single *Person.class* object
- **findByEmail**: returns single *Person.class* object
- **findAll**: returns all *Person.class* objects
- **remove**: remove one *Person.class* object from collection



TodoItemDAO implementation:

State: Any collection of type TodoItem
Constructor(s): Up to you
Methods:

- **persist**: add new *TodoItem.class* object to collection
- **findById**: returns single *TodoItem.class* object
- **findAll**: returns all *TodoItem.class* objects
- **findAllByDoneStatus**: returns many *TodoItem.class* objects where status match
- **findByTitleContains**: returns many *TodoItem.class* objects where title match.
- **findByPersonId**: returns many *TodoItem.class* objects where personId match todoItem.creator.id
- **findByDeadlineBefore**: returns many *TodoItem.class* objects where date is before deadline
- **findByDeadlineAfter**: returns many *TodoItem.class* objects where date is after deadline
- **remove**: remove one *TodoItem.class* object from collection

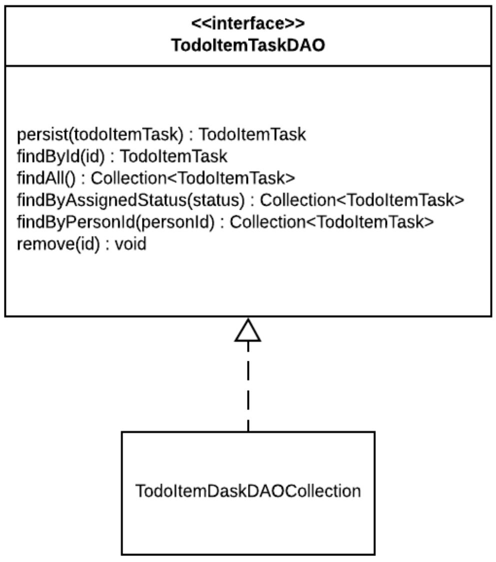


<<interface>>
TodoItemDAO
persist(todoItem) : TodoItem
findById(id) : TodoItem
findAll() : Collection<TodoItem>
findAllByDoneStatus(done) : Collection<TodoItem>
findByTitleContains(title) : Collection<TodoItem>
findByPersonId(personId) : Collection<TodoItem>
findByDeadlineBefore(date) : Collection<TodoItem>
findByDeadlineAfter(date) : Collection<TodoItem>
remove(id) : void
TodoItemDAOCollection

TodoItemTaskDAO implementation:

State: Any collection of type TodoItemTask
Constructor(s): Up to you
Methods:

- **persist**: add new *TodoItemTask.class* object to collection
- **findById**: returns single *TodoItemTask.class* object
- **findAll**: return all *TodoItemTask.class* objects
- **findByAssignedStatus**: return all *TodoItemTask.class* objects where assigned matches status
- **findByPersonId**: return all *TodoItemTask.class* objects where todoItemTask.assignee.id matches personId
- **remove**: removes one *TodoItemTask.class* object from collection



<<interface>>
TodoItemTaskDAO
persist(todoItemTask) : TodoItemTask
findById(id) : TodoItemTask
findAll() : Collection<TodoItemTask>
findByAssignedStatus(status) : Collection<TodoItemTask>
findByPersonId(personId) : Collection<TodoItemTask>
remove(id) : void
TodoItemDaskDAOCollection