

Metode Avansate de – 2018-2019 - LABORATOR 3

DEADLINE: Săptămâna 5

A. Cerințe funcționale:

- Se vor implementa funcționalitățile din Iterația 1 – vezi fișierul TemaProiectLaborator_MAP_2018-2019.pdf

B. Cerințe non-funcționale:

- Arhitectură Stratificată – pt Laboratorul 3 se cere doar layerul de Persistence (pt fiecare clasa din Domain, care are legatura cu Iteratia 1, veti avea un repository de date) si definirea entitatilor necesare Iteratiei 1;
- Toate funcțiile din Repository si Domain vor fi documentate și testate (unit teste);
- Documentația - Java doc (Tools->Generate Java Doc in IntelliJ Idea)
- *Persistența datelor: în memorie;*
- Folosiți interfața generică CrudRepository<ID,E> definită mai jos, fără a modifica această interfață;
- Entitățile din domeniu vor implementa interfața generică HasID<ID>;
- Validarea datelor - folosiți interfața Validator ([Strategy Pattern](#)) din exemplul de mai jos;
- Definiți propriile clase de excepții pentru tratarea situațiilor speciale;
- Nu se cere o interfață cu utilizatorul de tip consolă, sunt suficiente testele unitare.

```
/**
 * CRUD operations repository interface
 * @param <ID> - type E must have an attribute of type ID
 * @param <E> - type of entities saved in repository
 */

public interface CrudRepository<ID, E> {

    /**
     *
     * @param id -the id of the entity to be returned
     *            id must not be null
     * @return the entity with the specified id
     *         or null - if there is no entity with the given id
     * @throws IllegalArgumentException
     *         if id is null.
     */
    E findOne(ID id);

    /**
     *
     * @return all entities
     */
    Iterable<E> findAll();

    /**
     *
     * @param entity
     *            entity must be not null
     * @return null- if the given entity is saved
     *         otherwise returns the entity (id already exists)
     * @throws ValidationException
     *         if the entity is not valid
     * @throws IllegalArgumentException
     *         if the given entity is null.
     */
    E save(E entity) throws ValidationException;
```

```

    /**
     * removes the entity with the specified id
     * @param id
     *      id must be not null
     * @return the removed entity or null if there is no entity with the given id
     * @throws IllegalArgumentException
     *      if the given id is null.
     */
    E delete(ID id);

    /**
     *
     * @param entity
     *      entity must not be null
     * @return null - if the entity is updated,
     *      otherwise returns the entity - (e.g id does not exist).
     * @throws IllegalArgumentException
     *      if the given entity is null.
     * @throws ValidationException
     *      if the entity is not valid.
     */
    E update(E entity);
}

```

```

public interface Validator<E> {
    void validate(E entity) throws ValidationException;
}

```

```

public interface HasID<ID> {
    ID getID();
    void setID(ID id);
}

```