Metode Avansate de Programare, 2019–2020, LABORATOR 3

DEADLINE: Săptămânile 4 si 5

A. Cerințe funționale:

- Se vor implementa funcționalitățile din Iterația 1 – vezi fișierul **TemaProiectLaborator_DRIVE.docx**

B. Cerințe non-funționale:

- Arhitectură Stratificată – pt Laboratorul 3 se cere doar layerul de Persistance (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 InteliJ Idea)
- *Persistența datelor: în memorie;*
- Folosiți interfața generică `CrudRepository<ID,E>` definită mai jos, fără a modifica această interfață;
- Entitățile din domeniu extinde clasa `Entity<ID>;`
- Validarea datelor - folosiți interfața Validator (Strategy Pattern) din exemplul de mai jos;
- Definiți propriile clase de excepții pentru tratarea situațiilor speciale;
- interfața cu utilizatorul, de tip consolă, poate sa fie minimalista, nu e necesar un meniu

```
/**
 * 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 extends Entity<ID>> {

    /**
     *
     * @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 class Entity<ID> {
    private ID id;
    public ID getId() {
        return id;
    }
    public void setId(ID id) {
        this.id = id;
    }
}
```