Metode Avansate de Programare

LABORATOR 4

I. Cerință non-funcțională: **Deadline Săptămâna 5**

II. Persistenta datelo in baza de date: **Deadline Săptămâna 6**

I. Cerință non-funcțională: „Refactorizați ” codul de la laboratorul 3 astfel:
Interfata Repository va fi inlocuita cu urmatoarea:

```
/**
 * 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 Repository<ID, E extends Entity<ID>> {

    /**
     *
     * @param id -the id of the entity to be returned
     *           id must not be null
     * @return an {@code Optional} encapsulating the entity with the given id
     * @throws IllegalArgumentException
     *                  if id is null.
     */
    Optional<E> findOne(ID id);

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

    /**
     *
     * @param entity
     *         entity must be not null
     * @return an {@code Optional} - null if the entity was saved,
     *                             - the entity (id already exists)
     * @throws ValidationException
     *            if the entity is not valid
     * @throws IllegalArgumentException
     *             if the given entity is null.     *
     */
    Optional<E> save(E entity);


    /**
     *  removes the entity with the specified id
     * @param id
     *      id must be not null
     * @return an {@code Optional}
     *            - null if there is no entity with the given id,
     *            - the removed entity, otherwise
     * @throws IllegalArgumentException
     *                   if the given id is null.
     */
    Optional<E> delete(ID id);

    /**
     *
```

```
             * @param entity
             *          entity must not be null
             * @return  an {@code Optional}
             *              - null if the entity was updated
             *              - otherwise (e.g. id does not exist) returns the entity.
             * @throws IllegalArgumentException
             *              if the given entity is null.
             * @throws ValidationException
             *              if the entity is not valid.
             */
            Optional<E> update(E entity);

        }
```

Schimbarea interfetei cu cea de la punctul I are ca scop folosirea tipului Optional la tipul returnat de metodele update, delete, save si findOne. Rezolvați problemele care se propagă în cod în urma acestei refactorizări în toate layere aplicației.

Folosiți Java 8 features in implementarile metodelor de până acum în concordanță cu ce s-a discutat la cursul 4. (Interfețe funcționale, funcții lambda, referințe la metode, structurile repetitive folosite pt iterarea colecțiilor se vor înlocui cu foreach etc.)

De exemplu, o implementare pt metode save si delete din InMemoryRepository are putea fi următoarea:

```
@Override
public Optional<E> save(E entity)  {
    if (entity == null) {
        throw new IllegalArgumentException("id must not be null");
    }
    validator.validate(entity);
    return Optional.ofNullable(entities.putIfAbsent(entity.getId(), entity));
}

@Override
public Optional<E> delete(ID id) {
    if (id == null) {
        throw new IllegalArgumentException("id must not be null");
    }
    return Optional.ofNullable(entities.remove(id));
}
```