**Metode Avansate de Programare – 2019-2020 SEMINAR 3**

Collections and Generics/InFileRepository

1. 1) Creaţi o clasǎ Student avand urmatorii membri: nume(String), media(float), un constructor cu parametrii care initializeaza un Student si metoda toString();

Instantiati urmatorii studenti:

Student s1= **new** Student(**"Dan"**, 4.5f);  
Student s2= **new** Student(**"Ana"**, 8.5f);  
Student s3= **new** Student(**"Dan"**, 4.5f);

1. Instantiati un obiect de tip HashSet<Student> si adaugati studentii de la punctul I.1. Ce observati?
2. Respectati *Contractul equals - hashcode:* dacă obj1.equals(obj2) atunci obj1. hashCode() == obj2.hashCode().

Atunci cand doriti sa salvati obiecte în colecții ce sunt reprezentare in memorie pe tabele de dispersie (hash table), dacă implementați equals()trebuie sa implementati și hashcode().

1. Instantiati un obiect de tipul TreeSet<Student> si adaugati studentii de la punctul I.1). Definiti un comparator care compara doi studenti dupa nume.
2. Repetati exercitiile 2-4 folosind HashMap si TreeMap – temă opțională.
   1. Scrieţi o clasǎ MyMap, ce va reprezenta un Map pentru reţinerea studenţilor dupǎ medie. Caracteristicile clasei definite sunt:
3. Cheile pot avea valori de la 0 la 10 (corespunzǎtoare mediilor posibile).
4. Valoarea asociată fiecǎrei chei va fi o listǎ (List) care va reţine toţi studenţii cu media rotunjitǎ egalǎ cu cheia. Considerǎm cǎ un student are media rotunjitǎ 8 dacǎ media sa este în intervalul [7.50, 8.49]. (Math.round)
5. Map-ul vostru va menţine cheile (mediile) ordonate descrescǎtor. Folositi o implementare potrivitǎ a interfeţei Map, care sǎ permitǎ acest lucru, şi definiti un Comparator pentru stabilirea ordinii cheilor. (clasa internă statică)
6. Definiţi în clasa MyMap metoda *add(Student),* ce va adǎuga un student în lista corespunzǎtoare mediei lui. Dacǎ, în prealabil, nu mai existǎ nici un student cu media respectivǎ (rotunjitǎ), atunci lista va fi creatǎ la cerere. Observatie: Ce se intampla cand apelam metoda put mostenita de la dictionar? „Favor Composition instead of Inheritance” – DISCUTIE CU STUDENTII....
7. Adaugati câţiva studenţi.
8. Definiti o metoda getEntries() care returneaza multimea intrarilor – Entry<Key, Value>
9. Creati un dictionar de tipul MyMap si adaugati urmatorii studenti.

**public static** List<Student> getList(){  
 List<Student> l=**new** ArrayList<Student>();  
 l.add(**new** Student(**"1"**,9.7f));  
 l.add(**new** Student(**"2"**,7.3f));  
 l.add(**new** Student(**"3"**,6f));  
 l.add(**new** Student(**"4"**,6.9f));  
 l.add(**new** Student(**"5"**,9.5f));  
 l.add(**new** Student(**"6"**,9.9f));  
 **return** l;  
}

1. Iterati multimea intrarilor – Entry<Key, Value> şi sortaţi alfabetic fiecare listǎ de studenţi.
2. 1) Consideram interfata Repository care specifica operatiile CRUD pentru un Repository cu elemente generice care au un id de un tip generic ID.

**(ex.discutat la cursul 3- curs3\_generics.zip)**

*/\*\*  
 \* 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 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);  
}

3. Adaugati la proiectul de la curs3\_generics: RepositoryFile (fara lambda) pt entitatea User.