

# Metode Avansate de Programare – 2022-2023 SEMINAR 3

## Collections and Generics

- I. 1) Creați o clasă Student având următorii membri: nume(String), media(float), un constructor cu parametrii care initializează un Student și metoda toString();

Instantiați următorii studenți:

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

- 2) Instantiați un obiect de tip HashSet<Student> și adăugați studenții de la punctul I.1. Ce observați?  
3) Respectați *Contractul equals - hashCode*: dacă obj1.equals(obj2) atunci obj1.hashCode() == obj2.hashCode().

Atunci când doriți să salvați obiecte în colecții ce sunt reprezentare în memorie pe tabele de dispersie (hash table), dacă implementați equals() trebuie să implementați și hashCode().

- 4) Instantiați un obiect de tipul TreeSet<Student> și adăugați studenții de la punctul I.1). Definiți un comparator care compară doi studenți după nume.  
5) Repetați exercitiile 2-4 folosind HashMap și TreeMap

- II. Scrieți o clasă MyMap, ce va reprezenta un Map pentru reținerea studenților după medie. Caracteristicile clasei definite sunt:

- Cheile pot avea valori de la 0 la 10 (corespunzătoare mediilor posibile).
- Valoarea asociată fiecărei chei va fi o listă (List) care va reține toți studenții cu media rotunjită egală cu cheia. Considerăm că un student are media rotunjită 8 dacă media sa este în intervalul [7.50, 8.49]. (Math.round)
- Map-ul vostru va menține cheile (mediile) ordonate descrescător. Folosiți o implementare potrivită a interfeței Map, care să permită acest lucru, și definiți un Comparator pentru stabilirea ordinii cheilor. (clasa internă statică)
- Definiți în clasa MyMap metoda add(Student), ce va adăuga un student în lista corespunzătoare mediei lui. Dacă, în prealabil, nu mai există nici un student cu media respectivă (rotunjită), atunci lista va fi creată la cerere. Observație: Ce se întâmplă când apelăm metoda put moștenită de la dicționar? „Favor Composition instead of Inheritance” – DISCUTIE CU STUDENTII....
- Adăugați câțiva studenți.
- Definiți o metoda getEntries() care returnează mulțimea intrărilor – Entry<Key, Value>
- Creați un dicționar de tipul MyMap și adăugați următorii studenți.

```
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;
}
```

- 8) Iterați mulțimea intrărilor – Entry<Key, Value> și sortați alfabetic fiecare listă de studenți.

- III. 1) Definiți o interfață care specifică operațiile CRUD pentru un Repository cu elemente generice care au un id de un tip generic ID.

```
public interface Repository<E, ID> {
    E save(E entity);
    E delete(ID id);
    E findOne(ID id);
    Iterable<E> findAll();
}

public interface HasID<ID> {
    ID getId();
    void setId(ID id);
}
```

- 2) Definiți clasa abstractă `InMemoryRepository`, `public abstract class InMemoryRepository <E extends HasId<ID>, ID> implements Repository<E, ID> {...}`, care conține un dicționar, denumit `entities`, cu elemente generice de tipul `E`, și un validator generic pentru validarea entităților de tipul `E` din repository.