**@Override**

**public boolean equals(Object o) {**

**if (this == o) return true;**

**if (o == null || getClass() != o.getClass()) return false;**

**BusStation that = (BusStation) o;**

**if(!(Objects.equals(name, that.name))) return false;**

**if(!(Objects.equals(city, that.city))) return false;**

**return Objects.equals(buses, that.buses);**

equals tak jak on pisal. Pierwsze 3 linijki ok, potem 4 linijke trzeba rozbic tak o, tez w zaleznosci czy zmienne to stringi czy liczby bo dla int to jest tak:

**@Override**

**public boolean equals(Object o) {**

**if (this == o) return true;**

**if (o == null || getClass() != o.getClass()) return false;**

**InmutablePoint that = (InmutablePoint) o;**

**if (x != that.x) return false;**

**if (y != that.y) return false;**

**return z == that.z;**

**}**

**---------------------------------------------------------------------------------------------------**

**@Override**

**public int hashCode() {**

**int result = name != null ? name.hashCode() : 0;**

**result=result\*31+(city!=null? name.hashCode() : 0);**

**result=result\*31+(buses!=null? name.hashCode() : 0);**

**return result;**

**}**

Hasz jak trzeba, jest to dla stringow a dla int to:

**@Override**

**public int hashCode() {**

**int result = x;**

**result = 31 \* result + y;**

**result = 31 \* result + z;**

**return result;**

**}**

**@Override**

**public String toString() {**

**return super.toString() + ", numberOfVinyls=" + numberOfVinyls;**

**}**

**@Override**

**public boolean equals(Object o) {**

**if (!super.equals(o)) return false;**

**VinylStore that = (VinylStore) o;**

**return this.numberOfVinyls == that.numberOfVinyls;**

**}**

**@Override**

**public int hashCode() {**

**int hash = super.hashCode();**

**hash = 31 \* hash + this.numberOfVinyls;**

**return hash;**

**}**

To metody dla klasy dziedziczacej

**public ArrayList<String> getAlbums() {**

**return new ArrayList<>(albums);**

**}**

**public void setAlbums(ArrayList<String> albums) {**

**this.albums = albums!=null ? new ArrayList<>(albums) : new ArrayList<>();**

**}**

Get I set dla obiektu typu arraylist

**public MusicStore(String name, String city, ArrayList<String> albums) {**

**this.name = name;**

**this.city = city;**

**this.albums = albums!=null ? new ArrayList<>(albums) : new ArrayList<>(); (tak samo jak dla seta)**

**}**

Konstruktor gdy jest obiekt typu arraylist

**Collections.sort(lista,new AverageGradeComparator().thenComparing(new IdComparator()));**

(dla Arraylista, czyli listy tablicowej)

**Arrays.sort(songs, new DurationComparator().thenComparing(new ArtistTitleComparator()));**

(dla zwyklej tablicy [],czyli tablica obiektow)

**Arrays.*copyOf*(flightHours, flightHours.length);**

Do glebokiego kopiowania tablicy(flight hourse to double[]), trzeba to uzyc do konstruktora,gettera I setera

**clonedPilot.flightHours = Arrays.*copyOf*(this.flightHours, this.flightHours.length);**

a to w przypadku clonowania

**protected Object clone() throws CloneNotSupportedException {**

**Pilot clonedPilot = (Pilot) super.clone();**

**//** Głębokie kopiowanie ArrayList

**clonedPilot.flightHours = new ArrayList<>(this.flightHours);**

**return clonedPilot;**

**}**

**StringBuilder result = new StringBuilder();**

**for(Map.Entry<K,V> entry: map.entrySet())**

**{**

**result.append(entry.getKey())**

**.append(":")**

**.append(entry.getValue())**

**.append(", ");**

**}**

Foreach dla hashmapu, z dodaniem elementow do StringBuildera(potem w zadaniu trzeba było Stringiem zwrocic wiec toString)

**java.util.List<String> list = java.util.Arrays.*asList*("Apple", "Banana", "Orange", "Grapes");  
Iterator<String> iterator = list.iterator();**

przykladowe uzycie Iterator