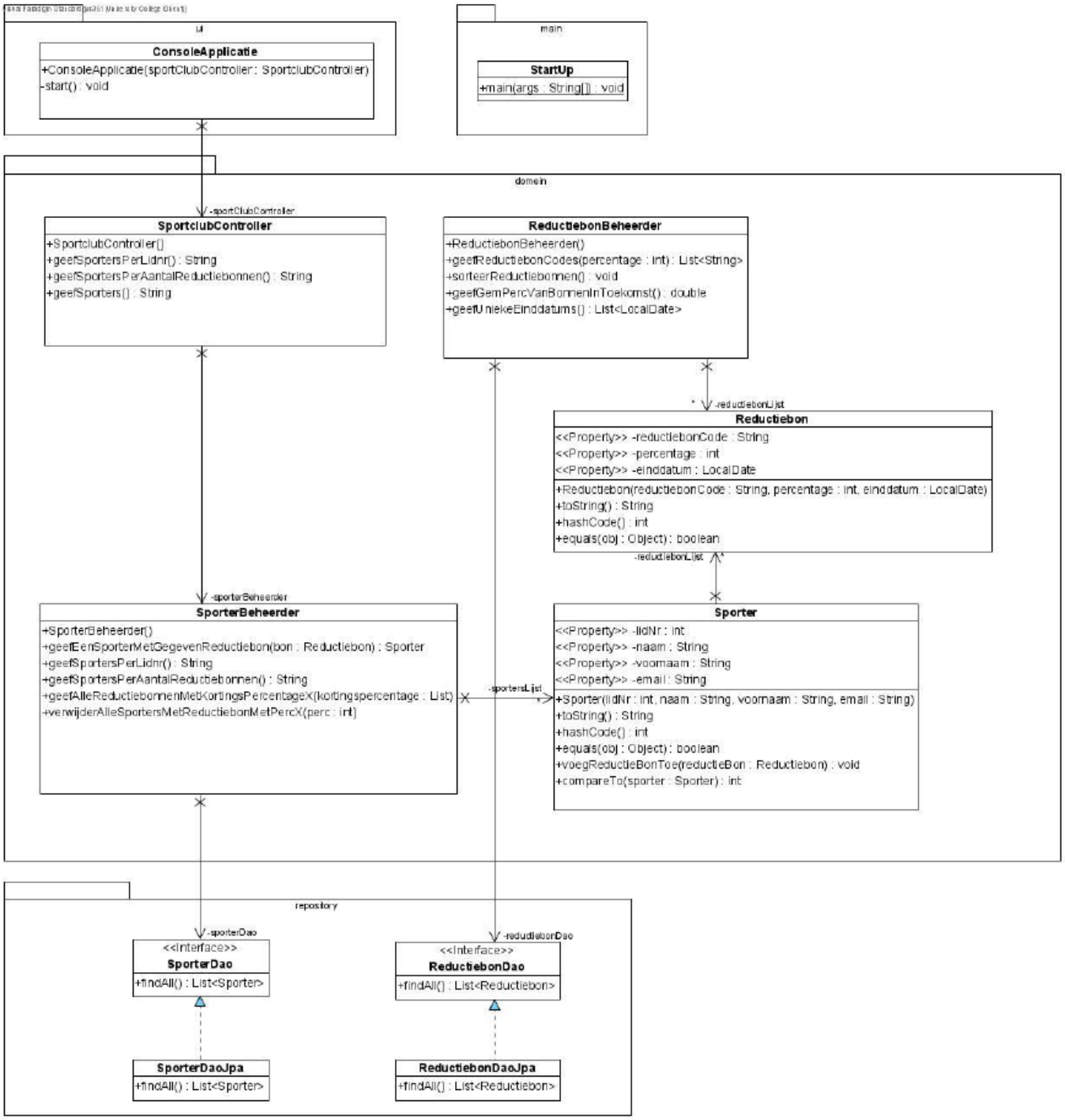
JAVA

Collections & streams

Reductiebonnen



Reductiebonbeheerder

Gemakzuchtige statische fabrieksmethoden op de List, Set en Map interfaces laten je eenvoudig niet wijzigbare lijsten, sets en maps maken. Een verzameling wordt als niet wijzigbaar beschouwd als elementen

niet kunnen worden toegevoegd, verwijderd of vervangen.
Omzetten naar een niet-wijzigbare lijst

```
public List<Reductiebon> getReductiebonLijst() {
    //return reductiebonLijst;
    return Collections.unmodifiableList(reductiebonLijst);
}
```

Vraag 1:

Methode geefReductiebonCodes: een lijst van reductiebonCodes wordt teruggegeven waarvan de percentage hoger ligt dan het meegegeven percentage.

```
public List<String> geefReductiebonCodes(int percentage) {
    return reductiebonLijst.stream()
        .filter(bon -> bon.getPercentage() > percentage)
        .map(Reductiebon::getReductiebonCode)
        .collect(Collectors.toList());
}
```

Vraag 2:

Methode sorteerReductiebonnen: sorteer de lijst met reductiebonnen volgens oplopende percentage (van laag naar hoog), en bij gelijke percentage op reductiebonCode – alfabetisch omgekeerde volgorde. (De originele lijst van reductiebonnen is gewijzigd.)

```
public void sorteerReductiebonnen() {
        reductiebonLijst.sort(Comparator.comparing(Reductiebon::getPercentage)

.thenComparing(Comparator.comparing(Reductiebon::getReductiebonCode)
                .reversed()));
}
```

Als er gevraagd werd, maak een nieuwe gesorteerde lijst dan moet er .stream().sorted() staan.

Vraag 3:

Methode geefGemPercVanBonnenInToekomst: geef het gemiddelde percentage terug van alle reductiebonnen die in de toekomst liggen (ter info: huidige datum: LocalDate.now()).

```
public double geefGemPercVanBonnenInToekomst() {
    return reductiebonLijst.stream() //Stream van reductiebonnen
                .filter(bon -> bon.getEinddatum().isAfter(LocalDate.now()))
                .mapToDouble(Reductiebon::getPercentage)
```

```
                .average().getAsDouble();
}
```

Vraag 4:

Methode geefUniekeEinddatums: geef alle unieke einddatums terug (m.a.w. geen dubbels), gesorteerd in stijgende volgorde.

```
public List<LocalDate> geefUniekeEinddatums() {
    return reductiebonLijst.stream()
            .map(Reductiebon::getEinddatum)
            .sorted().distinct()
            .collect(Collectors.toList());
}
```

Sporterbeheerder

```
#### Vraag 5:
Bekijk deze klasse en pas aan. Er is nog een encapsulatie lek.
```java
public Collection<Sporter> getSportersLijst() {
    return Collections.unmodifiableCollection(sportersLijst);
}
```

Vraag 6:

Methode geefEenSporterMetGegevenReductiebon: geeft een willekeurige sporter terug die de gegeven reductiebon bevat. Indien geen sporter aanwezig, dan wordt null teruggegeven.

```
public Sporter geefEenSporterMetGegevenReductiebon(Reductiebon bon) {
    return sportersLijst.stream()
            .filter(sporter -> sporter.getReductiebonLijst().contains(bon))
//Stream<Sporters>
            .findAny() //Optional<Sporter>
            .orElse(null);
}
```

Extra vraag 1

Methode geefAlleReductiebonnenMetKortingsPercentageX: geeft een lijst van alle Reductiebonnen met die één van de meegegeven kortingspercentages hebben.
```

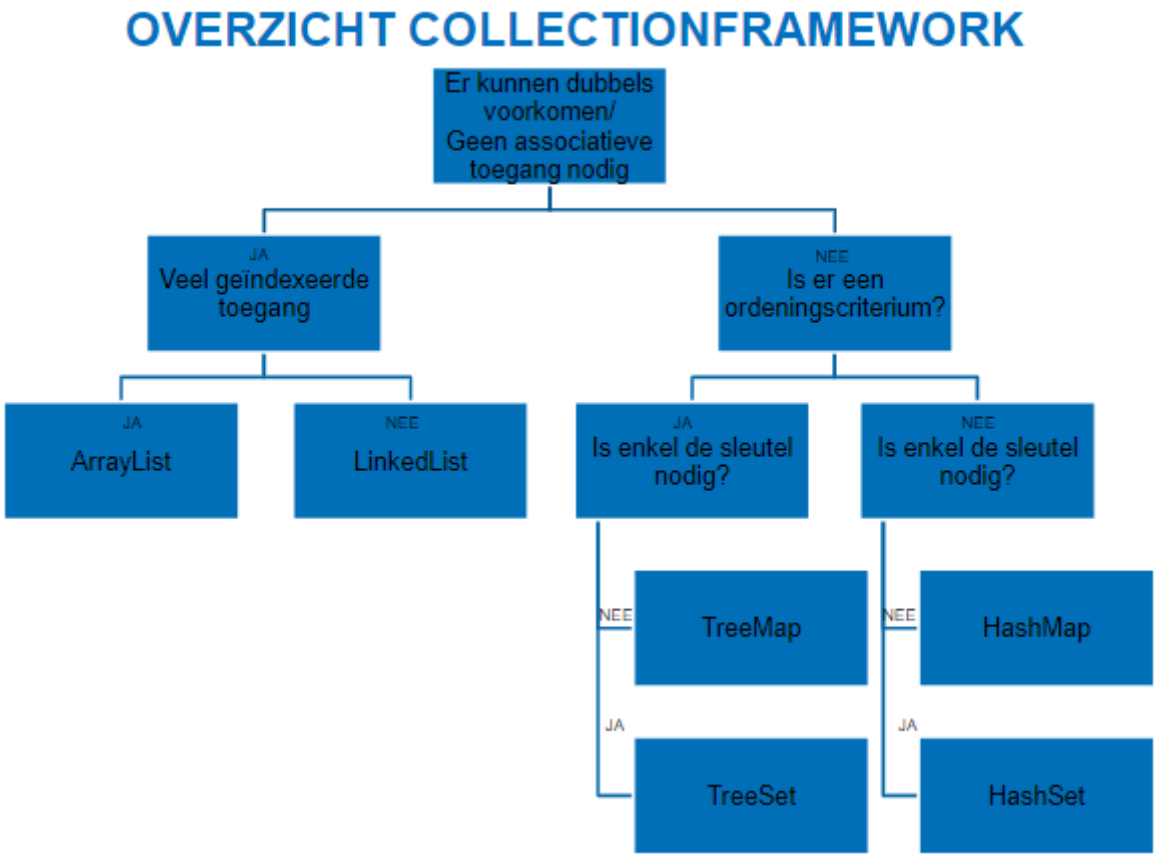
```
public List<Reductiebon>
geefAlleReductiebonnenMetKortingsPercentageX(List<Integer> kortingspercentage) {
    return sportersLijst.stream()
            .map(Sporter::getReductiebonLijst)
            .flatMap(Collection::stream)
            .filter(bon -> kortingspercentage.contains(bon.getPercentage()))
            .collect(Collectors.toList());
}
```

Extra vraag 2

Methode verwijderAlleSportersMetReductiebonMetPercX : zal alle sporters uit de originele lijst verwijderen die als korting het meegegeven percentage hebben. Opgelet: Testmethode uncomment

```
public void verwijderAlleSportersMetReductiebonMetPercX(int perc) {
    sportersLijst.removeIf(sporter -> sporter.getReductiebonLijst().stream()
            .anyMatch(bon -> bon.getPercentage()== perc));
}
```

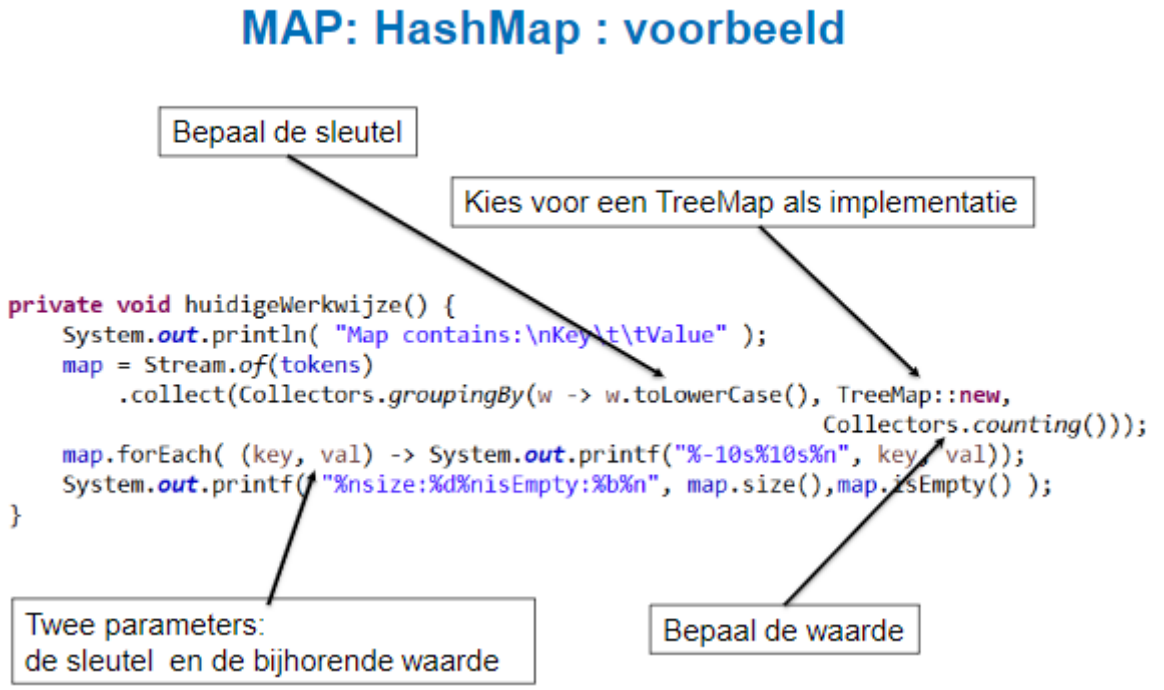
MAP

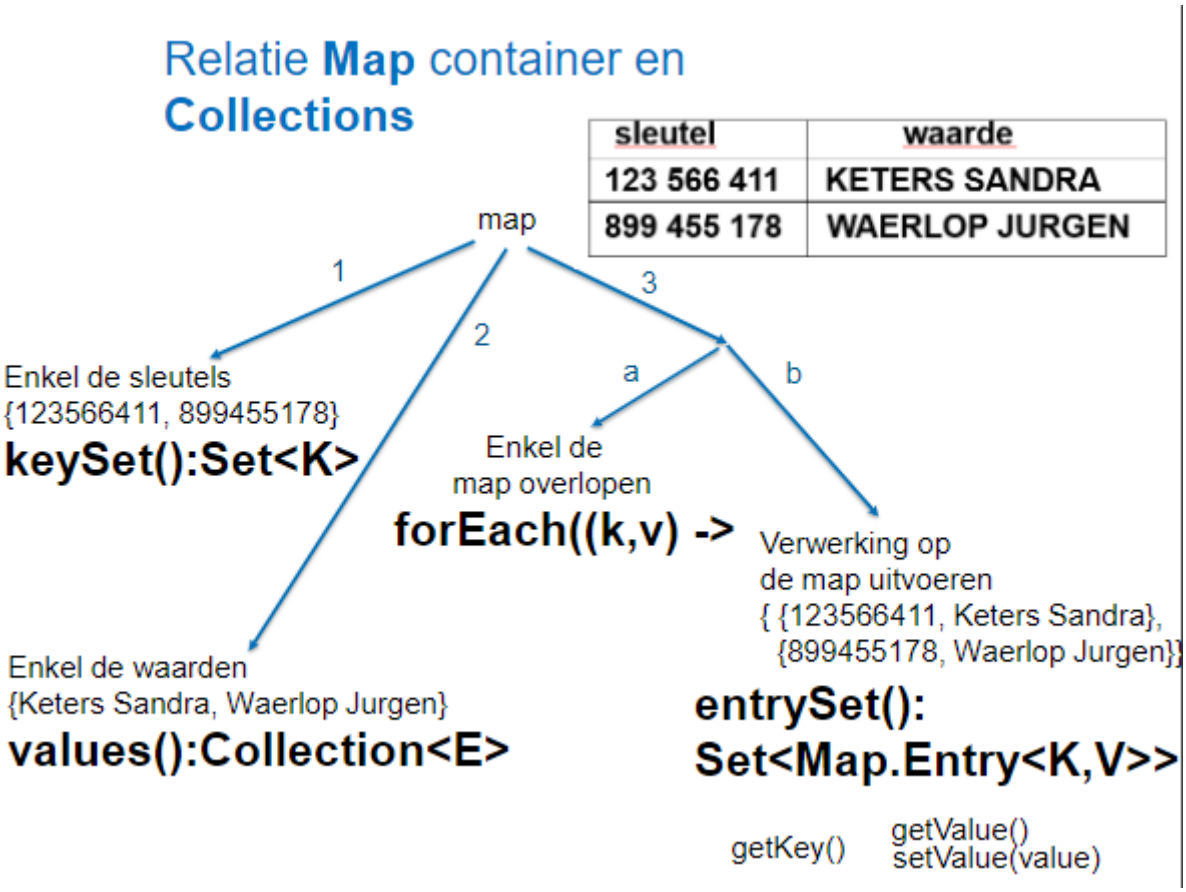


HashMap<K,V> en Hashtable<K,V> zijn implentatie-klassen van Map.
TreeMap<K,V> is een implementatie-klasse van SortedMap.
De klasse properties is een subklasse van HashTable<K,V>

Implementaties van interface Map<K,V>:

- HashMap<K,V>
  - Elementen opgeslagen in een hash-tabel
- Hashtable<K,V>
  - Zoals HashMap maar verouderde versie.
  - Werpt een NullPointerException indien de sleutel of value null is.
- TreeMap<K,V>
  - Gesorteerd
  - Elementen opgeslagen in boomstructuur
  - Implementatie van SortedMap subinterface van Map. Gebruik de natuurlijke volgorde of een comparator.





Java_Map_Oefeningen_start

```
class Auteur {

    private String naam, voornaam;

    public Auteur(String naam, String voornaam) {
        setNaam(naam);
```

```
        setVoornaam(voornaam);
    }

    public String getNaam() {
        return naam;
    }

    public String getVoornaam() {
        return voornaam;
    }

    public void setNaam(String naam) {
        this.naam = naam;
    }

    public void setVoornaam(String voornaam) {
        this.voornaam = voornaam;
    }

    @Override
    public String toString() {
        return String.format("%s %s", naam, voornaam);
    }
}

public class OefMap_opgave {
    public OefMap_opgave() {
        // we zullen een hashmap gebruiken waarbij auteursid de sleutel is en
        // de waarde is naam en voornaam van Auteur.
        //Creëer de lege hashMap "auteursMap"; de sleutel is van type Integer, de
waarde van type Auteur
        //------------------------------------------------------------------------
----------
        Map<Integer, Auteur> auteursMap = new HashMap<>();

        //Voeg toe aan de hashmap: auteursID = 9876, naam = Gosling, voornaam =
James
        //Voeg toe aan de hashmap: auteursID = 5648, naam = Chapman, voornaam =
Steve
        //------------------------------------------------------------------------
-------
        auteursMap.put(9876, new Auteur("Gosling", "James"));
        auteursMap.put(5648, new Auteur("Chapman", "Steve"));

        //Wijzig de voornaam van Chapman: John ipv Steve
        //-------------------------------------------
        auteursMap.get(5648).setVoornaam("John");

        //Komt de auteursID 1234 voor in de hashmap
        //-----------------------------------------
        if (auteursMap.containsKey(1234))
        System.out.println("auteursID 1234 komt voor\n");
        else
        System.out.println("auteursID 1234 komt niet voor\n");
```

```
        //Toon de naam en voornaam van auteursID 5648
        //-----------------------------------------

        Auteur auteur = auteursMap.get(5648);
        if (auteur != null)
            System.out.println(auteur);

        toonAlleAuteurs(auteursMap);

        //Alle auteursID's worden in stijgende volgorde weergegeven.
        //  1) de hashMap kopiëren naar een treeMap (= 1 instructie)
        //  2) roep de methode toonAlleSleutels op.
        //-----------------------------------------------------------
        Map<Integer, Auteur> treeMap = new TreeMap<>();
        toonAlleAuteurs(auteursMap);

    }

    public void toonAlleSleutels(Map<Integer, Auteur> map) {
        //Alle sleutels van de map worden op het scherm weergegeven.
        //-----------------------------------------------------------
        map.keySet().forEach(System.out::println);
        System.out.println();
    }

    public void toonAlleAuteurs(Map<Integer, Auteur> map) {
        /*Alle gegevens van de map worden op het scherm weergegeven.
        Per lijn wordt een auteursnr, naam en voornaam weergegeven.*/
        //-----------------------------------------------------------
        //NIET ZO -> OVERKILL
        //map.entrySet().stream().forEach(entry -> System.out.printf("%d %s%n",
entry.getKey(), entry.getValue()));
        map.forEach((auteursId, auteur) -> System.out.printf("%d %s%n",auteursId,
auteur));
        System.out.println();
    }

    public static void main(String args[]) {
        new OefMap_opgave();
    }
}
```

```
class CollectionOperaties {

    //methode verwijderOpLetter
    //-------------------------
    public static boolean verwijderOpLetter(List<String> list, char c) {
        return list.removeIf(elem -> elem.charAt(0)==c);
    }
```

```
    //methode verwijderSequence
    //-------------------------
    public static boolean verwijderSequence(List<String> list, String grens) {
        int first = list.indexOf(grens);
        if (first==-1) {
            return false;
        }
        int last = list.lastIndexOf(grens);
        list.subList(first, last).clear();
        return true;

    }

    //uitbreiding opgave Fruit   addOrdered
    //-------------------------------------
    public static boolean addOrdered(List<String> list, String fruit) {
        int index = Collections.binarySearch(list, fruit);
        if(index>=0) {
            return false;
        }
        list.add(index*-1,fruit);
        return true;
    }
}

public class OefFruit_opgave {

    public static void main(String args[]) {
        String kist[][] = {{"appel", "peer", "citroen", "kiwi", "perzik"},
        {"banaan", "mango", "citroen", "kiwi", "zespri", "pruim"},
        {"peche", "lichi", "kriek", "kers", "papaya"}};

        List<String> list =
Stream.of(kist).flatMap(Arrays::stream).collect(Collectors.toList());
        String mand[];

        //Toon de inhoud van de array "kist"
        //----------------------------------
        System.out.println(Arrays.deepToString(kist));
        //Arrays.toString werkt voor 1 dim array


        //Voeg de verschillende kisten samen in een ArrayList list.
        //---------------------------------------------------------


        CollectionOperaties.verwijderOpLetter(list, 'p');
        System.out.println("na verwijder letter ('p') :  " + list + "\n");

        CollectionOperaties.verwijderSequence(list, "kiwi");
        System.out.println("na verwijder sequence (kiwi) : " + list + "\n");

        //UITBREIDING
        list.sort(null);
```

```
        CollectionOperaties.addOrdered(list, "sapodilla");

        //Plaats het resultaat terug in een array mand en sorteer die oplopend.
        //---------------------------------------------------------------------
        mand = list.toArray(new String[0]);
        Arrays.sort(mand);

        //Toon de inhoud van de array "mand"
        //----------------------------------
        System.out.println(Arrays.toString(mand));
    }
}
```

```
public class OefFruitMap_opgave {

    public static void main(String args[]) {
        String kist[][] = {{"appel", "peer", "citroen", "kiwi", "perzik"},
        {"banaan", "mango", "citroen", "kiwi", "zespri", "pruim"},
        {"peche", "lichi", "kriek", "kers", "papaya"}};

        List<String> list =
Stream.of(kist).flatMap(Arrays::stream).collect(Collectors.toList());
        Scanner in = new Scanner(System.in);

        //declaratie + creatie map
        //------------------------------
        Map<String, Double> fruitMap = new TreeMap<>();

        /*Berg de fruit list van vorige oefeningen in een boom
        op zodat dubbels ge limineerd worden.
        Er moet ook de mogelijkheid zijn de bijhorende prijs
        (decimale waarde) bij te houden.*/
        //-------------------------------------------------------------
        list.forEach(fruit -> fruitMap.put(fruit, null));

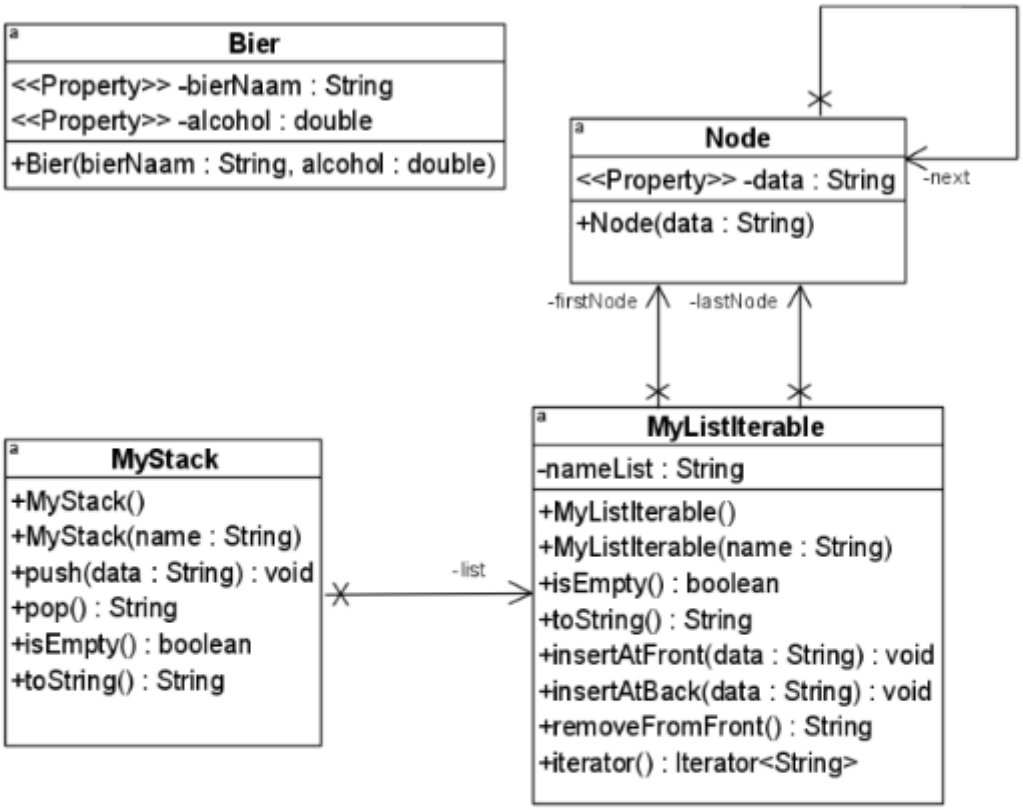
        /*Doorloop de boom in lexicaal oplopende volgorde en vraag
        telkens de bijhorende prijs, die je mee in de boom opbergt.*/
        //-------------------------------------------------------------
        //fruitMap.forEach(...);
        fruitMap.entrySet().stream().forEach(entry -> {
            System.out.printf("Prijs van %s : ", entry.getKey());
            double prijs = in.nextDouble();
            entry.setValue(prijs);
        });

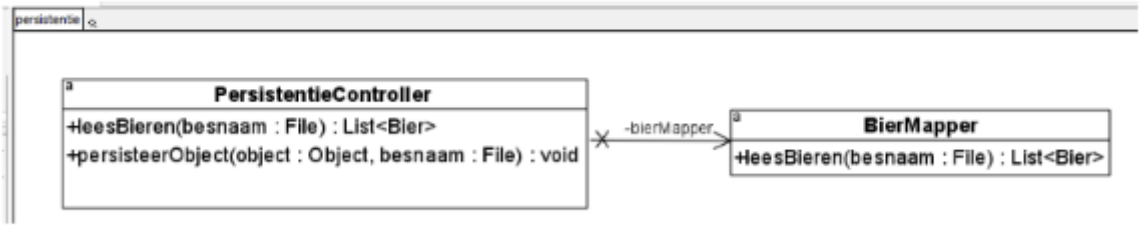

        /*Druk vervolgens de volledige lijst in twee
        kolommen (naam : prijs) in lexicaal oplopende volgorde af
        op het scherm.*/
        //-------------------------------------------------------------
        fruitMap.forEach((fruit, prijs) -> System.out.printf("%s\t%.2f%n", fruit,
```

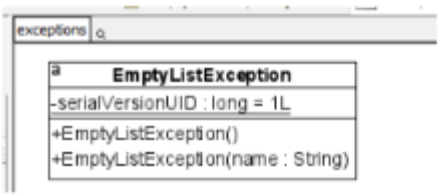
```
  prijs));
      }
  }
```

Java_Oefeningen2_start

Opgave







De gelinkte lijst is ontworpen voor String objecten als data. We willen de gelinkte lijst kunnen gebruiken voor alle mogelijke objecten.

Stap 1

Herwerk de MyListIterable-klasse: maak ze generiek. Zo kan je een lijst met eender welk object aanmaken. Maak ook MyStack generiek.

Stap2

Herschrijf de testmethoden van TestStackGeneriek in een generieke versie, zie TODO in de code

Stap3
Maak nu een lijst van Bier-objecten aan. De bierklasse bevat enkel de naam van het bier en het alcoholgehalte. De nodige gegevens om de objecten aan te maken lees je uit het tekstbestand "bieren.txt". De bierlijst persisteer je dan als één MyListIterable<Bier>-object onder de naam "bierenListObj.dat".
Je voorziet een methode **void persisteerMyList(..)**, met als argumenten een MyListIterable van objecten (vb. Bier-objecten) en een File voor de bestandsnaam. De hele lijst wordt als één object weggeschreven.
Zorg ervoor dat niet-serialiseerbare objecten in MyListIterable niet kunnen opgenomen worden, zodat we daardoor runtime exceptions kunnen vermijden. Bij een poging om dat wel te doen zal de compiler dit verhinderen.

Code

```
public class BierWinkel {

    private final List<Bier> bieren;
    private PersistentieController pc = new PersistentieController();

    public BierWinkel() {
        bieren = pc.inlezenBieren("bieren.txt");
    }

    //OEF
    public Map<String, Bier> opzettenOverzichtBierPerNaam() {
        return bieren.stream().collect(Collectors.toMap(Bier::getNaam,
Function.identity()));
    }

    //OEF
    public Map<String, List<Bier>> opzettenOverzichtBierenPerSoort() {
        //return bieren.stream().collect(Collectors.groupingBy(Bier::getSoort));
        return bieren.stream().collect(Collectors.groupingBy(Bier::getSoort,
TreeMap::new, Collectors.toList()));
    }

    //OEF
    public Map<String, Long> opzettenAantalBierenPerSoort() {
        return bieren.stream().collect(Collectors.groupingBy(Bier::getSoort,
TreeMap::new, Collectors.counting()));
    }
}
```

```
public class DomeinController {

    private final BierWinkel bierWinkel;

    public DomeinController() {
```

```
        bierWinkel = new BierWinkel();
    }

    public String opzettenBierPerNaam() {
        /*
         * return bierWinkel.opzettenOverzichtBierPerNaam().entrySet().stream()
         * .map(entry -> String.format("%s = %s", entry.getKey(),
entry.getValue()))
         * .collect(Collectors.joining("\n"));
         */
        return overzichtToString(bierWinkel.opzettenOverzichtBierenPerSoort());
    }

    public String opzettenAantalBierenPerSoort() {
        /*
         * return bierWinkel.opzettenAantalBierenPerSoort().entrySet().stream()
         * .map(entry -> String.format("%s = %s", entry.getKey(),
entry.getValue()))
         * .collect(Collectors.joining("\n"));
         */
        return overzichtToString(bierWinkel.opzettenAantalBierenPerSoort());
    }

    public String opzettenOverzichtBierenPerSoort() {
        /*
         * return bierWinkel.opzettenOverzichtBierenPerSoort().entrySet().stream()
         * .map(entry -> String.format("%s = %s", entry.getKey(),
entry.getValue()))
         * .collect(Collectors.joining("\n"));
         */
        return overzichtToString(bierWinkel.opzettenOverzichtBierenPerSoort());
    }

    // TODO na hoofdstuk generics
    // --> generieke oplossing "overzichtToString" methode
    //
    private <K, V> String overzichtToString(Map<K, V> map) {
        return map.entrySet().stream().map(entry -> String.format("%s = %s",
entry.getKey(), entry.getValue()))
                .collect(Collectors.joining("\n"));
    }

}
```

toMap

```
public class Sporter {

    private int lidNr;
    private String naam;
    private String voornaam;
```

```
    public Sporter(int lidNr, String naam, String voornaam) {
        setLidNr(lidNr);
        setNaam(naam);
        setVoornaam(voornaam);
    }

    public int getLidNr() {
        return lidNr;
    }

    private void setLidNr(int lidNr) {
        this.lidNr = lidNr;
    }

    public String getNaam() {
        return naam;
    }

    private void setNaam(String naam) {
        this.naam = naam;
    }

    public String getVoornaam() {
        return voornaam;
    }

    private void setVoornaam(String voornaam) {
        this.voornaam = voornaam;
    }

    @Override
    public String toString()
    {
        return String.format("%s %s", naam, voornaam);
    }
}
```

```
public class VoorbeeldToMap {

    public static void main(String args[])
    {
        new VoorbeeldToMap().voorbeeld();
    }

    public void voorbeeld()
    {
        List<Sporter> sportersLijst =
            List.of(new Sporter(789, "Keters", "Jan"),
                    new Sporter(123, "Blondee","Ann"),
                    new Sporter(456, "Keters", "Ann"),
```

```
                    new Sporter(147, "Vandriessche","Els"));

        System.out.printf("vertrekpunt lijst%n%s%n%n", sportersLijst);
        //van lijst naar map (per lidnummer - lidnummer is uniek)
        Map<Integer, Sporter> mapPerLidnummer = sportersLijst.stream().
                collect(Collectors.toMap(
                        Sporter::getLidNr, Function.identity()));

        System.out.printf(String.format("per lidnummer%n%s%n%n",
                geefMapAlsString(mapPerLidnummer)));


        try
        {
            //Poging van lijst naar map (per voornaam, waarde 1 sporter)
            Map<String, Sporter> mapPerVoornaam = sportersLijst.stream().
                    collect(Collectors.toMap(
                            Sporter::getVoornaam, Function.identity()));
        }catch(IllegalStateException e)
        {
            System.out.printf("poging per voornaam, waarde 1
sporter%n%s%n%n",e.getMessage());
        }

        //van lijst naar map (per voornaam, waarde 1 Sporter).
        Map<String, Sporter> mapPerVoornaam = sportersLijst.stream().
                collect(Collectors.toMap(
                        Sporter::getVoornaam, Function.identity(),
                        (voornaam1, voornaam2) -> voornaam1));

        System.out.printf(String.format("per voornaam, waarde 1 sporter%n%s%n%n",
                geefMapAlsString(mapPerVoornaam)));

        //van lijst naar map (per naam, waarde 1 Sporter), gesorteerd per sleutel
        Map<String, Sporter> mapPerNaam = sportersLijst.stream().
                collect(Collectors.toMap(
                        Sporter::getNaam, Function.identity(),
                        (naam1, naam2) -> naam1, TreeMap::new));

        System.out.printf(String.format("per naam, waarde 1 sporter, gesorteerd op
sleutel%n%s%n%n",
                geefMapAlsString(mapPerNaam)));

        //van lijst naar map (per lidnummer - lidnummer is uniek), gesorteerd per
sleutel
        Map<Integer, Sporter> mapPerLidnummer2 = sportersLijst.stream().
                collect(Collectors.toMap(
                        Sporter::getLidNr, Function.identity(),
                        (key1, key2) -> key1, TreeMap::new));

        System.out.printf(String.format("per lidnummer, gesorteerd op
sleutel%n%s%n%n",
                geefMapAlsString(mapPerLidnummer2)));
    }
```

```
    public <K,V> String geefMapAlsString(Map<K,V> eenMap)
    {
        return eenMap.entrySet().stream().
            map(entry -> String.format("sleutel: %-12s waarde: %-15s",
entry.getKey(), entry.getValue())).
                    collect(Collectors.joining("\n"));
    }
}
```

```
public class OefFruitMap_opgave {

    public static void main(String args[]) {
        String kist[][] = {{"appel", "peer", "citroen", "kiwi", "perzik"},
        {"banaan", "mango", "citroen", "kiwi", "zespri", "pruim"},
        {"peche", "lichi", "kriek", "kers", "papaya"}};

        List<String> list =
Stream.of(kist).flatMap(Arrays::stream).collect(Collectors.toList());
        Scanner in = new Scanner(System.in);

        //declaratie + creatie map
        //-----------------------------
        //Map<String, Double> fruitMap = new TreeMap<>();

        /*Berg de fruit list van vorige oefeningen in een boom
 op zodat dubbels ge limineerd worden.
 Er moet ook de mogelijkheid zijn de bijhorende prijs
 (decimale waarde) bij te houden.*/
        //-------------------------------------------------------------
        //list.forEach(fruit -> fruitMap.put(fruit, null));
        Map<String, Double> fruitMap = list.stream().collect(Collectors.toMap(f ->
f, f -> 0.0, (fruit1, fruit2) -> fruit1, TreeMap::new));

        /*Doorloop de boom in lexicaal oplopende volgorde en vraag
 telkens de bijhorende prijs, die je mee in de boom opbergt.*/
        //-------------------------------------------------------------
        //fruitMap.forEach(...);
        fruitMap.entrySet().stream().forEach(entry -> {
            System.out.printf("Prijs van %s : ", entry.getKey());
            double prijs = in.nextDouble();
            entry.setValue(prijs);
        });


        /*Druk vervolgens de volledige lijst in twee
 kolommen (naam : prijs) in lexicaal oplopende volgorde af
 op het scherm.*/
        //-------------------------------------------------------------
        fruitMap.forEach((fruit, prijs) -> System.out.printf("%s\t%.2f%n", fruit,
prijs));
```

```
    }
}
```

Generics

- Alle generieke methoden hebben een **"type paramter section"**, dat tussen < en > staat.
- Elke **"type paramter section"** bevat één of meer **"(formal) type parameters"**, gescheiden door komma's
- Een **"type parameter"** kan enkel referenties bevatten, geen primitieve datatypes!

```
public static < E >void printArray( E[]inputArray){
    Stream.of(inputArray).forEach(
      element -> System.out.printf("%s ", element));System.out.println();
}
```

- De generieke methode 'printArray' heeft als **"type parameter section"**
- ...

Oefeningen

Generieke methoden

```
public class Generieke_methoden_opgave {

    public static void main(String args[]) {
        new Generieke_methoden_opgave();
    }

    public Generieke_methoden_opgave() {
        Double[] decGetallen = {3.0, 7.8, 9.0, 10.6};
        List<Double> doubleLijst = new ArrayList<>(Arrays.asList(decGetallen));

        Integer[] integer = {5, 8, 9, -6, 0, 7};
        List<Integer> integerLijst = new ArrayList<>(Arrays.asList(integer));

        String[] woorden = {"aaa", "bbb", "ccc", "ddd", "eee", "fff", "ggg",
"hhh", "a"};
        List<String> stringLijst = new ArrayList<>(Arrays.asList(woorden));

//methode minimum oproepen
        Integer minGetal1 = minimum(integerLijst);
        Double minGetal2 = minimum(doubleLijst);

        System.out.printf("%s%d\n\n", "oplossing is -6, jouw oplossing = ",
minGetal1);
        System.out.printf("%s%.1f\n\n", "oplossing is 3.0, jouw oplossing = ",
minGetal2);
```

```
//methode geefAlleElementenKleinerDan oproepen
        List<Integer> lijstInteger = geefAlleElementenKleinerDan(integerLijst, 8);
        List<String> lijstString = geefAlleElementenKleinerDan(stringLijst,
"ddd");

        weergevenLijst("oplossing : 5 -6 0 7", lijstInteger);
        weergevenLijst("oplossing : aaa bbb ccc a", lijstString);

    }

    public <E> void weergevenLijst(String oplossing, Collection<E> lijst) {

        System.out.printf("%s\n%s", oplossing, "            ");
        lijst.forEach(element -> System.out.printf("%s ", element));
        System.out.println("\n");
    }

    //-------------------------------------------------------------------------
    //Schrijf de generieke methode "minimum" die het kleinste getal van een
Collection
    //van type E teruggeeft.
    //-------------------------------------------------------------------------
    public <E extends Comparable<E>> E minimum(Collection<E> lijst) {
        return lijst.stream().min(E::compareTo).get();
    }


    //-------------------------------------------------------------------------
---------
    //Schrijf de generieke methode geefAlleElementenKleinerDan
    //De methode heeft twee argumenten: een Collection van type E "lijst" en een
element van type E "grens"
    //De methode geeft een List van type E terug; alle elementen van "lijst" die
kleiner
    //zijn dan "grens" worden in de lijst bewaard.
    //-------------------------------------------------------------------------
---------
    public <E extends Comparable<E>> List<E>
geefAlleElementenKleinerDan(Collection<E> lijst, E grens){
        return lijst.stream().filter(element -> element.compareTo(grens)
<0).collect(Collectors.toList());
    }
}
```

Generieke klasse

```
//Gegeven:

abstract class Dier {
}
```

```
interface Huisdier {
}

class Hond extends Dier implements Huisdier {
}

class Kat extends Dier implements Huisdier {
}

class Leeuw extends Dier {
}

//Gevraagd:
//schrijf een klasse die een set van Kat, Hond of Huisdier kan bevatten
class VerzamelingHuisdier<E extends Huisdier>
{
    //attribuut "huisdieren" = set van Kat, Hond of Huisdier
    //-------------------------------------------------------------------------
    private Set<E> huisdieren = new HashSet<>();

    //getHuisdieren
    //-------------------
    public Set<E> getHuisdieren(){
        return Collections.unmodifiableSet(huisdieren);
    }

    //methode add: een dier toevoegen in de set
    //-----------------------------------------------------------
    public void add(E dier) {
        huisdieren.add(dier);
    }
}

class Tools {
    //methode bevatHuisdier met twee argumenten: een huisdier en een set van Kat,
Hond of Huisdier.
    //Geeft true terug indien het huisdier in de set voorkomt, anders false.
    //schrijf de static methode bevatHuisdier d.m.v. wildcards
    //-------------------------------------------------------------------------
-----------
    public static boolean bevatHuisdier(Huisdier huisdier, Set<? extends Huisdier>
setHuisdieren) {
        return setHuisdieren.contains(huisdier);
    }

    //schrijf de static methode bevatHuisdier2 d.m.v. generieke methode
    //-------------------------------------------------------------------------
-----------
    public static <T extends Huisdier> boolean bevatHuisdier2(T huisdier, Set<?
extends Huisdier> setHuisdieren ) {
        return setHuisdieren.contains(huisdier);
    }
```

```
    //schrijf de static methode geefAantal. 1 argument: Een set van Kat, Dier,
Huisdier of Object.
    //Aantal elementen van de set wordt teruggegeven.
    //-------------------------------------------------------------------------
----------
    public static int geefAantal(Set<? super Kat> dier) {
        return dier.size();
    }
}

public class Generieke_klasse_opgave {

    public static void main(String args[]) {
        Kat kat = new Kat();
        Hond hond = new Hond();
        Huisdier huisdier = new Hond();

        VerzamelingHuisdier<Kat> katten = new VerzamelingHuisdier<>();
        katten.add(kat);
        VerzamelingHuisdier<Hond> honden = new VerzamelingHuisdier<>();

        VerzamelingHuisdier<Huisdier> huisdieren = new VerzamelingHuisdier<>();
        huisdieren.add(huisdier);

        boolean komtVoor = Tools.bevatHuisdier(kat, katten.getHuisdieren());
        System.out.printf("correct = true;  %s%n", komtVoor);
        komtVoor = Tools.bevatHuisdier(hond, honden.getHuisdieren());
        System.out.printf("correct = false;  %s%n", komtVoor);
        komtVoor = Tools.bevatHuisdier(huisdier, huisdieren.getHuisdieren());
        System.out.printf("correct = true;  %s%n", komtVoor);
        //compileerfout: komtVoor = Tools.bevatHuisdier(kat, new HashSet<Dier>());

        komtVoor = Tools.bevatHuisdier2(kat, katten.getHuisdieren());
        System.out.printf("correct = true;  %s%n", komtVoor);
        komtVoor = Tools.bevatHuisdier2(hond, honden.getHuisdieren());
        System.out.printf("correct = false;  %s%n", komtVoor);
        komtVoor = Tools.bevatHuisdier2(huisdier, huisdieren.getHuisdieren());
        System.out.printf("correct = true;  %s%n", komtVoor);

        int aantalKatten = Tools.geefAantal(katten.getHuisdieren());
        int aantalHuisdieren = Tools.geefAantal(huisdieren.getHuisdieren());
       //compileerfout: int aantalHonden =
Tools.geefAantal(honden.getHuisdieren());

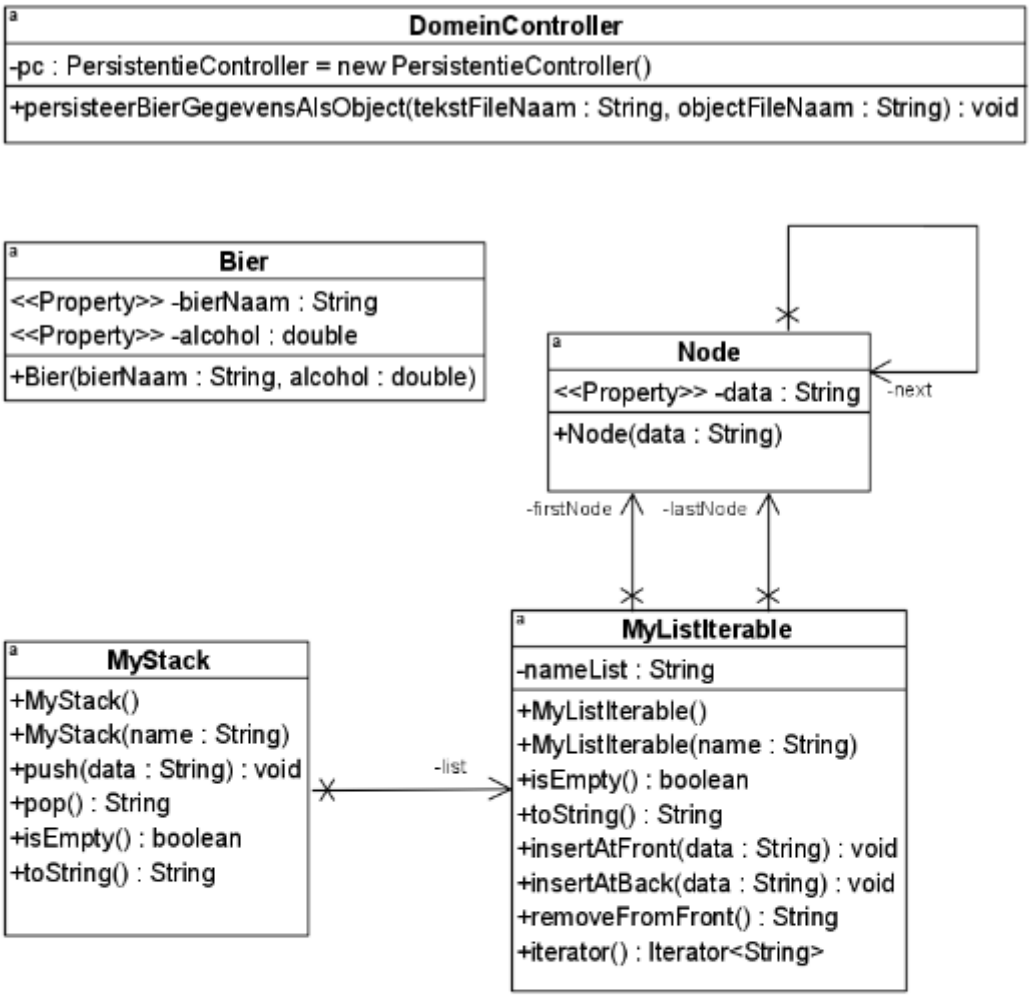
        Set<Dier> dieren = Set.of(new Leeuw(), new Hond());
        int aantalDieren = Tools.geefAantal(dieren);

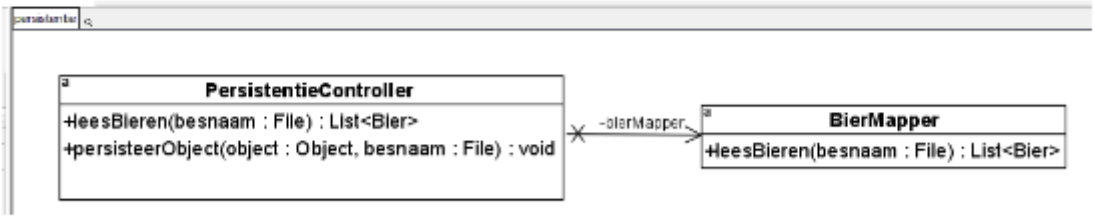
        int aantalObjecten = Tools.geefAantal(new HashSet<Object>());

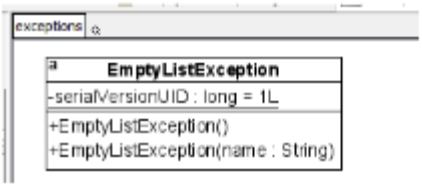
        System.out.printf("correct = 1;  %d%ncorrect = 1;  %d%ncorrect = 2;
%d%ncorrect = 0;  %d%n", aantalKatten,
                aantalHuisdieren, aantalDieren, aantalObjecten);
     }
```

```
}
```

Oefeningen 2







De gelinkte lijst is ontworpen voor String objecten als data. We willen de gelinkte lijst kunnen gebruiken voor alle mogelijke objecten.

Stap 1

Herwerk de MyListIterable-klasse: maak ze generiek. Zo kan je een lijst met eender welk object aanmaken. Maak ook MyStack generiek.

Stap2

Herschrijf de testmethoden van TestStackGeneriek in een generieke versie, zie TODO in de code

Stap3
Maak nu een lijst van Bier-objecten aan. De bierklasse bevat enkel de naam van het bier en het alcoholgehalte. De nodige gegevens om de objecten aan te maken lees je uit het tekstbestand "bieren.txt". De bierlijst persisteer je dan als één MyListIterable<Bier>-object onder de naam "bierenListObj.dat".
Je voorziet een methode **void persisteerMyList(..)**, met als argumenten een MyListIterable van objecten (vb. Bier-objecten) en een File voor de bestandsnaam. De hele lijst wordt als één object weggeschreven.
Zorg ervoor dat niet-serialiseerbare objecten in MyListIterable niet kunnen opgenomen worden, zodat we daardoor runtime exceptions kunnen vermijden. Bij een poging om dat wel te doen zal de compiler dit verhinderen.

Implements Serializable plaatsen

```
public class Bier implements Serializable{

    private String bierNaam;
    private double alcohol;

    public Bier(String bierNaam, double alcohol) {
        this.bierNaam = bierNaam;
        this.alcohol = alcohol;
    }

    public String getBierNaam() {
        return bierNaam;
    }

    public void setBierNaam(String bierNaam) {
        this.bierNaam = bierNaam;
    }

    public double getAlcohol() {
        return alcohol;
    }

    public void setAlcohol(double alcohol) {
        this.alcohol = alcohol;
    }
}
```

```
public class Node<T extends Serializable> implements Serializable{

    private final T data;
    private Node<T> next;

    public Node(T data) {
        this.data = data;
```

```
    }

    public T getData() {
        return data;
    }

    public void setNext(Node<T> next) {
        this.next = next;
    }

    public Node<T> getNext() {
        return next;
    }
}
```

```
public class MyListIterable<T extends Serializable> implements Iterable<T>,
Serializable {

    private Node<T> firstNode;
    private Node<T> lastNode;
    private String nameList;

    public MyListIterable() {
        this("List");
    }

    public MyListIterable (String name) {
        nameList = name;
    }

    public boolean isEmpty() {
        return firstNode == null;
    }

    @Override
    public String toString() {
        if (isEmpty()) {
            return nameList + " is empty";
        }

        StringBuilder buffer = new StringBuilder();
        buffer.append("The ").append(nameList).append(" is: ");

        this.forEach(elem -> buffer.append(elem).append(" "));
        return buffer.toString();
    }

    public void insertAtFront(T data) {
        Node<T> aNode = new Node<>(data);
        if (isEmpty()) {
            firstNode = lastNode = aNode;
```

```
            } else {
                aNode.setNext(firstNode);
                firstNode = aNode;
            }
        }

        public void insertAtBack(T data) {
            Node<T> aNode = new Node<>(data);
            if (isEmpty()) {
                firstNode = lastNode = aNode;
            } else {
                lastNode.setNext(aNode);
                lastNode = lastNode.getNext();
            }
        }

        public T removeFromFront() throws EmptyListException {
            if (isEmpty()) {
                throw new EmptyListException(nameList);
            }

            T removedItem = firstNode.getData();
            if (firstNode == lastNode) {
                firstNode = lastNode = null;
            } else {
                firstNode = firstNode.getNext();
            }

            return removedItem;
        }

        @Override
        public Iterator<T> iterator() {
            return new MyIterator();
        }

        private class MyIterator implements Iterator<T> {

            private Node<T> nextToFetch = firstNode;

            @Override
            public boolean hasNext() {
                return nextToFetch != null;
            }

            @Override
            public T next() {
                T data = nextToFetch.getData();
                nextToFetch = nextToFetch.getNext();
                return data;
            }
        }
    }
```

```java
public class MyStack<T extends Serializable> implements Serializable {

    /**
     *
     */
    private static final long serialVersionUID = 1L; //Wordt automatisch gegeneerd
    private MyListIterable<T> list;

    public MyStack() {
        this("Stack");
    }

    public MyStack(String name) {
        list = new MyListIterable<>(name);
    }

    public void push(T data) {
        list.insertAtFront(data);
    }

    public T pop() {
        return list.removeFromFront();
    }

    public boolean isEmpty() {
        return list.isEmpty();
    }

    @Override
    public String toString() {
        return list.toString();
    }

}
```

```java
public class TestStackGeneriek {

    private MyStack<String> woordenStack;
    private MyStack<Double> decGetallenStack;

    @BeforeEach
    public void before() {
        woordenStack = new MyStack<>("woordenStack");
        decGetallenStack = new MyStack<>("decimaleGetallenStack");
    }

    ...

    @Test
```

```
    public void toevoegenVerwijderenTesten() {
        String[] woorden = {"lekker", "zijn", "wafels"};
        Double[] decGetallen = {12.5, 24.3, 30.5, 40.5};
        itemsToevoegenVerwijderen(woordenStack, woorden);
        itemsToevoegenVerwijderen(decGetallenStack, decGetallen);
    }

    //public void itemsToevoegenVerwijderen(MyStack myStack,     items){

    //-------------------------------------------------------------------------
    public <E extends Serializable> void itemsToevoegenVerwijderen(MyStack<E>
myStack, E[] items) {
        Stream.of(items).forEach(myStack::push);

        List<E> listExpected = Arrays.asList(items);
        Collections.reverse(listExpected);

        listExpected.stream().limit(listExpected.size()-1).forEach(expected -> {
            Assertions.assertEquals(expected, myStack.pop());
            Assertions.assertFalse(myStack.isEmpty());
        });

        Assertions.assertEquals(listExpected.get(listExpected.size() -1),
myStack.pop());
        Assertions.assertTrue(myStack.isEmpty());
    }

    ...
```

```
public class PersistentieController {

   private BierMapper bierMapper;

    public List<Bier> leesBieren(File besnaam) {
        if (bierMapper == null) {
            bierMapper = new BierMapper();
        }
        return bierMapper.leesBieren(besnaam);
    }

    //TODO stap 3, maak de methode generiek
    public <T extends Serializable> void persisteerObject(T object, File besnaam)
{
        try (ObjectOutputStream out = new
ObjectOutputStream(Files.newOutputStream(besnaam.toPath()))) {
            out.writeObject(object);
        } catch (IOException ex) {
            Logger.getLogger(BierMapper.class.getName()).log(Level.SEVERE, null,
ex);
        }
```

```
    }
}
```

```
public class DomeinController {

    private PersistentieController pc = new PersistentieController();

    public void persisteerBierGegevensAlsObject(String tekstFileNaam, String
objectFileNaam){
        //TODO zie stap3
        List<Bier> lijstBier = pc.leesBieren(new File(tekstFileNaam));
        MyListIterable<Bier> myList = new MyListIterable<>();
        lijstBier.forEach(myList::insertAtBack);
        pc.persisteerObject(myList, new File(objectFileNaam));
    }

}
```

Netwerken

Versturen van string (client)

```
private Socket socket;
private Scanner sockInput;
private Formatter sockOutput;

    public FileTransfer(String host) {
        try {
            //Maak verbinding met server, init attributen
            //TODO
            socket = new Socket(host, 44444);
            sockInput = new Scanner(socket.getInputStream());
            sockOutput = new Formatter(socket.getOutputStream());
        } catch (IOException ex) {
            System.out.println("Probleem " + ex.getMessage());
        }
    }
```

Ontvangen van string (server)

```
private Socket socket;
private Scanner sockInput;
private Formatter sockOuput;

    public void run() {
        //initialiseer server
```

```
        //TODO
        try (ServerSocket ss = new ServerSocket(44444, 10)) {
            System.out.println("Fileserver up");
            while (true) {
                try {
                    System.out.println("Fileserver waiting...");
                    socket = ss.accept();
                    processClient();
                } catch (IOException ex) {
                    System.out.println("Problemen : " + ex.getMessage());
                }
            }
        } catch (IOException ex) {
            System.out.println("Problemen met server connectie : " +
                                                    ex.getMessage());
        }
    }
```

Oefening

Client

```
public class FileTransfer {
    //Attributen netwerkverbinding/streams
    //TODO
    private Socket socket;
    private Scanner sockInput;
    private Formatter sockOutput;

    public FileTransfer(String host) {
        try {
            //Maak verbinding met server, init attributen
            //TODO
            socket = new Socket(host, 44444);
            sockInput = new Scanner(socket.getInputStream());
            sockOutput = new Formatter(socket.getOutputStream());
        } catch (IOException ex) {
            System.out.println("Probleem " + ex.getMessage());
        }
    }

    public String readFile(String fileNaam) {
        //verzoek server om bestand  'fileNaam' door te sturen
        //lees het bestand in als de server het doorstuurt
        //TODO
        sockOutput.format("%s%n", "READ");
        sockOutput.format("%s%n", fileNaam);
        sockOutput.flush();
        String reactie = sockInput.nextLine();
        if(reactie.equals("FOUND")) {
            String line;
```

```
            StringBuilder fileContent = new StringBuilder();
            while(!(line = sockInput.nextLine()).equals("*E*O*F*")) {
                //verwerking
                fileContent.append(line).append(System.lineSeparator());
            }
            return fileContent.toString();
        }

        return "BESTAND NIET GEVONDEN";
    }

    public void updateFile(String fileContents, String fileNaam) {
        //meld de server dat je het gewijzigde bestand gaat doorsturen
        //geef de eventueel gewijzigde bestandsnaam mee door
        //bij onveranderde bestandsnaam zal de server het originele bestand
overschrijven
        //stuur het bestand door naar de server
        //TODO
        sockOutput.format("%s%n", "REWRITE");
        sockOutput.format("%s%n", fileNaam);
        sockOutput.flush();
        sockOutput.format(fixEOL(fileContents));
        sockOutput.format("%s%n", "*E*O*F*");
        sockOutput.flush();
    }

    public void closeConnection() {
        try {
        //TODO
        socket.close();
        } catch (IOException ex) {
            System.out.println("Probleem " + ex.getMessage());
        }
    }

    //De laatste regel moet eindigen met de systeem einde regel.
    //Enkel \n of \r is niet goed.
    public String fixEOL(String text) {
        if (!text.endsWith(System.lineSeparator())) {
            int count = 0;
            int lastChar = text.length() - 1;
            if (lastChar >= 0 && (text.charAt(lastChar) == '\n' ||
                    text.charAt(lastChar) == '\r')) {
                count++;
            }
            if (lastChar > 0 && (text.charAt(lastChar - 1) == '\r')) {
                count++;
            }
            if (count > 0) {
                text = text.substring(0, text.length() - count);
            }
            return text + System.lineSeparator();
        }
        return text;
```

```
    }
}
```

Server

```
public class FileServer {
    //attributen voor netwerkconnectie en streams
    //TODO
    private Socket socket;
    private Scanner sockInput;
    private Formatter sockOutput;

    public void run() {
        //initialiseer server
        //TODO
        try (ServerSocket ss = new ServerSocket(44444, 10)) {
            System.out.println("Fileserver up");
            //wacht tot een client verbindig maakt
            //verwerk al de verzoeken van een client tot deze afsluit
            //          delegeer naar hulpmethode processClient
            //wacht opnieuw op een client, blijft dit doen
            //TODO
            while (true) {
                try {
                    System.out.println("Fileserver waiting...");
                    socket = ss.accept();
                    processClient();
                } catch (IOException ex) {
                    System.out.println("Problemen : " + ex.getMessage());
                }
            }
        } catch (IOException ex) {
            System.out.println("Problemen met server connectie : " +
                                                ex.getMessage());
        }
    }

    private void processClient() {
        //verwerk al de verzoeken van een client volgens het afgesproken protocol
        //tot deze afsluit
        //sluit dan ook de connectie met deze client
        //maak gebruik van de 3 onderstaande hulpmethoden
        try {
            sockInput = new Scanner(socket.getInputStream());
            sockOutput = new Formatter(socket.getOutputStream());
            File file;
            while(sockInput.hasNextLine()) {
                String actie = sockInput.nextLine();
                switch(actie) {
                    case "READ" -> {
                        file = new File(sockInput.nextLine());
```

```
                        System.out.println("READ" + file.getName());
                        if(file.exists())
                            sendFile(file);
                        else
                            sendNoFile();
                    }
                    case "REWRITE" -> {
                        file = new File(sockInput.nextLine());
                        System.out.println("REWRITE" + file.getName());
                        readAndSaveUpdateFile(file);
                    }
                }
            }
        } catch (IOException ex) {
            System.out.println("Problemen met client connectie : " +
                    ex.getMessage());
        }
    }

    private void sendFile(File file) throws IOException {
        //TODO
        sockOutput.format("%s%n", "FOUND");
        sockOutput.flush();
        try(Scanner diskFile = new Scanner(file)){
            while(diskFile.hasNext()) {
                sockOutput.format("%s%n", diskFile.nextLine());
                sockOutput.flush();
            }
        }
        sockOutput.format("%s%n", "*E*O*F*");
        sockOutput.flush();
    }

    private void sendNoFile() {
        //TODO
        sockOutput.format("%s%n", "NOTFOUND");
        sockOutput.flush();
    }

    private void readAndSaveUpdateFile(File file) throws IOException {
        //TODO
        try(Formatter naarSchijf = new Formatter(file)){
            String line;
            while(!(line = sockInput.nextLine()).equals("*E*O*F*")) {
                naarSchijf.format("%s%n", file);
            }
        }
    }
}
```

Herhalingsoefeningen

CollectionsOpstart0_start1

Collections Vervolg 1 (Map)

1. In **SporterBeheerder**:

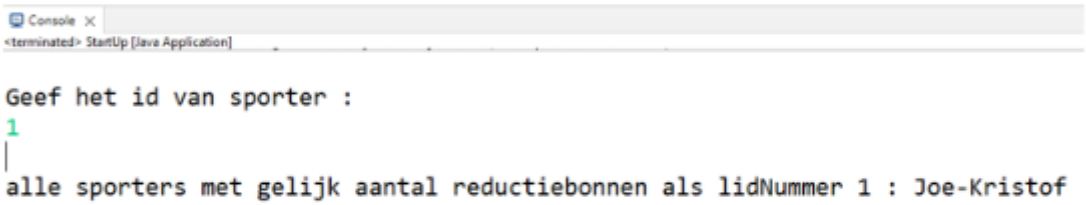
Methode **maakOverzichten**: maak de volgende overzichten:
- een overzicht van sporters per lidnummer (sportersPerLidnummer).
- een overzicht van sporters per aantal reductiebonnen (sportersPerAantalReductiebonnen).

Vul de 2 attributen in de klasse verder aan

2. In **ConsoleApplicatie/SportclubController/SportBeheerder.**

De applicatie vraagt een lidnummer op en geeft dan als resultaat alle sporters terug die evenveel reductiebonnen hebben als de ingegeven sporter.

Output: 1



LidNummer 1 (dat is Joe) heeft 2 reductiebonnen. Enkel Kristof heeft ook 2 reductiebonnen.

Na het dubbelpunt geven we de namen gescheiden met een – teken.

Output: 2

```
Geef het id van sporter :
0
alle sporters met gelijk aantal reductiebonnen als lidNummer 0 : Ongekend lidnummer
```

Resultaat bij niet bestaand lidnummer.

Uitwerking:

→ ConsoleApplicatie
opvragen sportLidnummer en afdrukken aanroep methode in SportclubController

→ SporclubController

String **geefSportersMetEvenveelReductiebonnen**(int lidnr)

→ SportBeheerder

```
public Sporter geefSporter(int sporterLidNr)

public List<Sporter> geefSportersMetEvenveelReductiebonnen(Sporter
                                                           sporter)
```

```
public class SporterBeheerder {

    private SporterDao sporterDao;
    private Collection<Sporter> sportersLijst;

    // TODO OEF MAP extra attributen
    private Map<Integer, Sporter> sportersPerLidnummer;
    private Map<Integer, List<Sporter>> sportersPerAantalReductiebonnen;
```

```
    public SporterBeheerder() {
        sporterDao = new SporterDaoJpa();
        sportersLijst = sporterDao.findAll();
        // TODO OEF MAP
        maakOverzichten();
    }

    public Collection<Sporter> getSportersLijst() {
        return Collections.unmodifiableCollection(sportersLijst);
    }

    // TODO OEF MAP extra methoden
    public void maakOverzichten() {
        sportersPerLidnummer =
sportersLijst.stream().collect(Collectors.toMap(Sporter::getLidNr,
Function.identity()));
        sportersPerAantalReductiebonnen =
sportersLijst.stream().collect(Collectors.groupingBy(sporter ->
sporter.getReductiebonLijst().size()));
    }

    // VRAAG 6
    public Sporter geefEenSporterMetGegevenReductiebon(Reductiebon bon) {
        return sportersLijst.stream().filter(sporter ->
sporter.getReductiebonLijst().contains(bon)).findAny()
                .orElse(null);
    }

    // EXTRA vraag1
    public List<Reductiebon>
geefAlleReductiebonnenMetKortingsPercentageX(List<Integer> kortingspercentage) {
        return
sportersLijst.stream().map(Sporter::getReductiebonLijst).flatMap(Collection::strea
m)
                .filter(bon ->
kortingspercentage.contains(bon.getPercentage())).collect(Collectors.toList());
    }

    // EXTRA vraag2
    public void verwijderAlleSportersMetReductiebonMetPercX(int perc) {
        sportersLijst.removeIf(
                s -> s.getReductiebonLijst().stream().filter(bon ->
bon.getPercentage() == perc).count() != 0);
    }

    public String geefSportersPerLidnr() {
        throw new UnsupportedOperationException();
    }

    public String geefSportersPerAantalReductiebonnen() {
        throw new UnsupportedOperationException();
    }
```

```
        // OEF GENERICS
        // methode geefAlleSleutelsWaarden
    }
```

SporterBeheerder en SportClubController toMap

Collections Vervolg 1 (Map)

1. In **SporterBeheerder**:

Methode **maakOverzichten**: maak de volgende overzichten:
- een overzicht van sporters per lidnummer (sportersPerLidnummer).
- een overzicht van sporters per aantal reductiebonnen (sportersPerAantalReductiebonnen).

Vul de 2 attributen in de klasse verder aan

2. In **ConsoleApplicatie/SportclubController/SportBeheerder.**

De applicatie vraagt een lidnummer op en geeft dan als resultaat alle sporters terug die evenveel reductiebonnen hebben als de ingegeven sporter.

Output: 1


```
Geef het id van sporter :
1
alle sporters met gelijk aantal reductiebonnen als lidNummer 1 : Joe-Kristof
```

LidNummer 1 (dat is Joe) heeft 2 reductiebonnen. Enkel Kristof heeft ook 2 reductiebonnen.

Na het dubbelpunt geven we de namen gescheiden met een – teken.

Output: 2

```
Geef het id van sporter :
0
alle sporters met gelijk aantal reductiebonnen als lidNummer 0 : Ongekend lidnummer
```

Resultaat bij niet bestaand lidnummer.

Uitwerking:

→ ConsoleApplicatie
opvragen sportLidnummer en afdrukken aanroep methode in SportclubController

→ SporclubController

String **geefSportersMetEvenveelReductiebonnen**(int lidnr)

→ SportBeheerder

```
public Sporter geefSporter(int sporterLidNr)

public List<Sporter> geefSportersMetEvenveelReductiebonnen(Sporter
                                                           sporter)
```

```
public class SporterBeheerder {

    private SporterDao sporterDao;
```

```
    private Collection<Sporter> sportersLijst;

    // TODO OEF MAP extra attributen
    private Map<Integer, Sporter> sportersPerLidnummer;
    private Map<Integer, List<Sporter>> sportersPerAantalReductiebonnen;

    public SporterBeheerder() {
        sporterDao = new SporterDaoJpa();
        sportersLijst = sporterDao.findAll();
        // TODO OEF MAP
        maakOverzichten();
    }

    public Collection<Sporter> getSportersLijst() {
        return Collections.unmodifiableCollection(sportersLijst);
    }

    // TODO OEF MAP extra methoden
    public void maakOverzichten() {
        sportersPerLidnummer =
sportersLijst.stream().collect(Collectors.toMap(Sporter::getLidNr,
Function.identity()));
        sportersPerAantalReductiebonnen =
sportersLijst.stream().collect(Collectors.groupingBy(sporter ->
sporter.getReductiebonLijst().size()));
    }

    // VRAAG 6
    public Sporter geefEenSporterMetGegevenReductiebon(Reductiebon bon) {
        return sportersLijst.stream().filter(sporter ->
sporter.getReductiebonLijst().contains(bon)).findAny()
                .orElse(null);
    }

    // EXTRA vraag1
    public List<Reductiebon>
geefAlleReductiebonnenMetKortingsPercentageX(List<Integer> kortingspercentage) {
        return
sportersLijst.stream().map(Sporter::getReductiebonLijst).flatMap(Collection::strea
m)
                .filter(bon ->
kortingspercentage.contains(bon.getPercentage())).collect(Collectors.toList());
    }

    // EXTRA vraag2
    public void verwijderAlleSportersMetReductiebonMetPercX(int perc) {
        sportersLijst.removeIf(
                s -> s.getReductiebonLijst().stream().filter(bon ->
bon.getPercentage() == perc).count() != 0);
    }

    public String geefSportersPerLidnr() {
        throw new UnsupportedOperationException();
    }
```

```
    public String geefSportersPerAantalReductiebonnen() {
        throw new UnsupportedOperationException();
    }

    public Sporter geefSporter(int sporterLidNr) {
        return sportersPerLidnummer.get(sporterLidNr);
    }

    public List<Sporter> geefSportersMetEvenveelReductiebonnen(Sporter sporter){
        return
sportersPerAantalReductiebonnen.get(sporter.getReductiebonLijst().size());
    }
    // OEF GENERICS
    // methode geefAlleSleutelsWaarden

}
```

```
public class SportclubController {

    private SporterBeheerder sporterBeheerder;

    public SportclubController() {
        sporterBeheerder = new SporterBeheerder();
    }

//TODO uncomment OEF GENERICS
//  public String geefSportersPerLidnr() {
//      return sporterBeheerder.geefSportersPerLidnr();
//  }
//
//  public String geefSportersPerAantalReductiebonnen() {
//      return sporterBeheerder.geefSportersPerAantalReductiebonnen();
//  }

    public String geefSporters() {
        throw new UnsupportedOperationException();
    }

    public String geefSportersMetEvenveelReductiebonnen(int sporterLidNr) {
        Sporter sporter = sporterBeheerder.geefSporter(sporterLidNr);
        if(sporter==null) {
            return "Ongekend lidnummer";
        }
        List<Sporter> sporters =
sporterBeheerder.geefSportersMetEvenveelReductiebonnen(sporter);

        if(sporters.size() == 1) {
            return "Geen sporters met gelijk aantal kortingsbonnen";
        }
```

```
        return
sporters.stream().map(Sporter::getNaam).collect(Collectors.joining("-"));
    }

    //TODO OEF GENERICS...
}
```

SporterBeheerder en SportClubController generics

Collections Vervolg 2 (Generics)

```
Overzicht sporters per lidnummer:
1: sporter 1, Joe Blondeel heeft 2 reductiebon(nen)
2: sporter 2, Kristof Roose heeft 2 reductiebon(nen)
3: sporter 3, Ann Vanderstene heeft 0 reductiebon(nen)
4: sporter 4, Christel Hereman heeft 1 reductiebon(nen)
5: sporter 5, Emiel Beulens heeft 0 reductiebon(nen)
6: sporter 6, Ilias Keters heeft 1 reductiebon(nen)

Overzicht sporters per aantal reductiebonnen
0: [sporter 3, Ann Vanderstene heeft 0 reductiebon(nen), sporter 5, Emiel Beulens heeft 0 reductiebon(nen)]
1: [sporter 4, Christel Hereman heeft 1 reductiebon(nen), sporter 6, Ilias Keters heeft 1 reductiebon(nen)]
2: [sporter 1, Joe Blondeel heeft 2 reductiebon(nen), sporter 2, Kristof Roose heeft 2 reductiebon(nen)]

Overzicht sporters:
sporter 1, Joe Blondeel heeft 2 reductiebon(nen)
sporter 2, Kristof Roose heeft 2 reductiebon(nen)
sporter 3, Ann Vanderstene heeft 0 reductiebon(nen)
sporter 4, Christel Hereman heeft 1 reductiebon(nen)
sporter 5, Emiel Beulens heeft 0 reductiebon(nen)
sporter 6, Ilias Keters heeft 1 reductiebon(nen)
```

In **SportBeheerder**:

1. Implementeer de generieke methode **geefAlleSleutelsWaarden** die voor een willekeurige Map de verzameling van sleutels en waarden als ***1 string*** teruggeeft. Zowel voor sleutel als voor waarde mag de toString() worden gebruikt. De sleutels zijn in stijgende volgorde ***gesorteerd***. Tussen sleutel en waarde staat een ***dubbelpunt***. De volgende sleutel start terug op de volgende regel.
   Bv.
   sleutel1: waarde1
   sleutel2: waarde2
   etc
   zie bovenstaande uitvoer, de eerste twee overzichten.

In **SportclubController**:

2. Schrijf een generieke methode **geefGesorteerdeCollectie** die een willekeurige collectie volgens zijn natuurlijke volgorde sorteert. De collectie wordt meegegeven als parameter.
   Het bekomen resultaat wordt teruggegeven door de methode en omgevormd naar een String (scheidingsteken = newline). Hiervoor wordt de toString-methode van het object gebruikt.
   Definieer de methode op een zodanige **strenge manier** dat ze altijd zal werken!

3. Implementeer de methode **geefSporters** a.h.v. de bovenstaande methode geefGesorteerdeCollectie. De sportersLijst van sporterBeheerder wordt aan de methode geefGesorteerdeCollectie doorgegeven.
   Vul eventueel de nodige code aan in een andere klasse om dit te laten werken.
   zie bovenstaande uitvoer, derde overzicht.

4. Voeg een nieuwe generieke klasse **MyCollection** toe in het domein en een subklasse **Actiebon** van klasse **Reductiebon** (zie verder).

Nieuwe klasse **BonController** toevoegen in domein.

```java
public class BonController {
    private ReductiebonBeheerder reductiebonBeheerder;

    public BonController() {
        reductiebonBeheerder = new ReductiebonBeheerder();
    }

    public void  init() {
        //versie A dient te werken
        /*reductiebonBeheerder.storeBon(new Reductiebon("kerstbon", 50, LocalDate.of(2023, 12,10)));
        reductiebonBeheerder.storeBon(new Actiebon("Kerstdagen", "kerst", 15, LocalDate.of(2023, 12, 31)));
        */

        //versie B dient te werken
        /*List<Reductiebon> bonnen = List.of(new Reductiebon("kerstbon", 50, LocalDate.of(2023, 12,10)),
                new Actiebon("Kerstdagen", "kerst", 15, LocalDate.of(2023, 12, 31)));
        reductiebonBeheerder.addReeks(bonnen);*/


        //versie C dient te werken
        List<Actiebon> bonnen = List.of(new Actiebon("Paasdagen", "paas", 50, LocalDate.of(2023, 4,1)),
                new Actiebon("Kerstdagen", "kerst", 15, LocalDate.of(2023, 12, 31)));
        reductiebonBeheerder.addReeks(bonnen);
```

```
        Reductiebon bon = reductiebonBeheerder.retrieveBon();
        System.out.println(bon);
        Reductiebon bon2 = reductiebonBeheerder.retrieveBon();
        System.out.println(bon2);
    }
}
```

Startup aanpassen – aanmaak bc en oproep init().

```
public class StartUp {
    public static void main(String[] args) {
        //new ConsoleApplicatie(new SportclubController());
        BonController bc = new BonController();
        bc.init();
    }
}
```

Aanpassingen in **ReductiebonBeheerder**: extra attribuut en aanpassing constructor

```
//Extra deel generics
private MyCollection<Reductiebon> myCollection;
//end

public ReductiebonBeheerder() {
    reductiebonDao = new ReductiebonDaoJpa();
    reductiebonLijst = reductiebonDao.findAll();
    //extra Deel generics
    myCollection = new MyCollection<>();
    //end
}
```

En toevoegen van volgende methoden

```
//EXTRA vraag generics
public void storeBon(Reductiebon bon) {
    myCollection.addElement(bon);
}
public Reductiebon retrieveBon() {
    return myCollection.getElement();
}
//public void addReeks AANVULLEN...
//
//
//}
//end
```

De extra klasse Actiebon in domein toevoegen:

```
public class Actiebon extends Reductiebon {
    private String beschrijving;
    public Actiebon(String beschrijving, String reductiebonCode, int percentage, LocalDate einddatum) {
        super(reductiebonCode, percentage, einddatum);
    }
    public String getBeschrijving() {
        return beschrijving;
    }
    public void setBeschrijving(String beschrijving) {
        this.beschrijving = beschrijving;
    }
}
```

Werk de generieke klasse MyCollection uit. Het dient bonnen (Reductiebon, Actiebon) in een wachtrij bij te houden. We voorzien ook de mogelijkheid om het gemiddelde percentage van de bonnen in de lijst op te vragen. De werking kan je nagaan door de in de init() methode telkens één van de versies te uncommenten.

Klasse **MyCollection** bevat methoden **addElement**, **getElement**, **addReeks**(een lijst van bonnen) en **gemiddeldePercentage**().

```
public class SporterBeheerder {

    private SporterDao sporterDao;
    private Collection<Sporter> sportersLijst;

    // TODO OEF MAP extra attributen
    private Map<Integer, Sporter> sportersPerLidnummer;
    private Map<Integer, List<Sporter>> sportersPerAantalReductiebonnen;

    public SporterBeheerder() {
        sporterDao = new SporterDaoJpa();
        sportersLijst = sporterDao.findAll();
        // TODO OEF MAP
        maakOverzichten();
    }

    public Collection<Sporter> getSportersLijst() {
        return Collections.unmodifiableCollection(sportersLijst);
    }

    // TODO OEF MAP extra methoden
```

```
    public void maakOverzichten() {
        sportersPerLidnummer =
sportersLijst.stream().collect(Collectors.toMap(Sporter::getLidNr,
Function.identity()));
        sportersPerAantalReductiebonnen = sportersLijst.stream()
                .collect(Collectors.groupingBy(sporter ->
sporter.getReductiebonLijst().size()));
    }

    // VRAAG 6
    public Sporter geefEenSporterMetGegevenReductiebon(Reductiebon bon) {
        return sportersLijst.stream().filter(sporter ->
sporter.getReductiebonLijst().contains(bon)).findAny()
                .orElse(null);
    }

    // EXTRA vraag1
    public List<Reductiebon>
geefAlleReductiebonnenMetKortingsPercentageX(List<Integer> kortingspercentage) {
        return
sportersLijst.stream().map(Sporter::getReductiebonLijst).flatMap(Collection::strea
m)
                .filter(bon ->
kortingspercentage.contains(bon.getPercentage())).collect(Collectors.toList());
    }

    // EXTRA vraag2
    public void verwijderAlleSportersMetReductiebonMetPercX(int perc) {
        sportersLijst.removeIf(
                s -> s.getReductiebonLijst().stream().filter(bon ->
bon.getPercentage() == perc).count() != 0);
    }
    //TODO
    public String geefSportersPerLidnr() {
        return geefAlleSleutelsWaarden(sportersPerLidnummer);
    }
    //TODO
    public String geefSportersPerAantalReductiebonnen() {
        return geefAlleSleutelsWaarden(sportersPerAantalReductiebonnen);
    }

    public Sporter geefSporter(int sporterLidNr) {
        return sportersPerLidnummer.get(sporterLidNr);
    }

    public List<Sporter> geefSportersMetEvenveelReductiebonnen(Sporter sporter) {
        return
sportersPerAantalReductiebonnen.get(sporter.getReductiebonLijst().size());
    }

    // OEF GENERICS
    public <K, V> String geefAlleSleutelsWaarden(Map<K, V> eenMap) {
        Map<K, V> gesorteerdeMap = new TreeMap<>(eenMap);
        return gesorteerdeMap.entrySet().stream()
```

```
                .map(entry -> String.format("%s: %s", entry.getKey(),
entry.getValue()))
                .collect(Collectors.joining("\n"));
    }
}
```

Sporter comparable maken

```
public class Sporter implements Comparable<Sporter> {

    private int lidNr;
    ...
    //GENERICS
    @Override
    public int compareTo(Sporter sporter) {
        return lidNr - sporter.lidNr;
    }
}
```

```
public class SportclubController {

    private SporterBeheerder sporterBeheerder;

    public SportclubController() {
        sporterBeheerder = new SporterBeheerder();
    }

//TODO uncomment OEF GENERICS
    public String geefSportersPerLidnr() {
        return sporterBeheerder.geefSportersPerLidnr();
    }

    public String geefSportersPerAantalReductiebonnen() {
        return sporterBeheerder.geefSportersPerAantalReductiebonnen();
    }

    //TODO OEF GENERICS...
    public String geefSporters() {
        return geefGesorteerdeCollectie(sporterBeheerder.getSportersLijst());
    }

    public String geefSportersMetEvenveelReductiebonnen(int sporterLidNr) {
        Sporter sporter = sporterBeheerder.geefSporter(sporterLidNr);
        if(sporter==null) {
            return "Ongekend lidnummer";
        }
        List<Sporter> sporters =
sporterBeheerder.geefSportersMetEvenveelReductiebonnen(sporter);
```

```
        if(sporters.size() == 1) {
            return "Geen sporters met gelijk aantal kortingsbonnen";
        }

        return
sporters.stream().map(Sporter::getNaam).collect(Collectors.joining("-"));
    }

    //TODO OEF GENERICS...
    public <T extends Comparable<T>> String geefGesorteerdeCollectie(Collection<T>
collectie) {
        return
collectie.stream().sorted().map(T::toString).collect(Collectors.joining("\n"));
    }
}
```

```
public class MyCollection<T extends Reductiebon> {
    private Queue<T> deLijst;

    public MyCollection() {
        deLijst = new ArrayDeque<>();
    }

    public void addElement(T element) {
        deLijst.offer(element);
    }

    public T getElement() {
        return deLijst.poll();
    }

    public void addReeks(List<? extends T> bonnen) {
        deLijst.addAll(bonnen);
    }

    public double gemiddeldPercentage() {
        return
deLijst.stream().mapToDouble(T::getPercentage).average().getAsDouble();
    }
}
```

```
public class ReductiebonBeheerder {

    private ReductieBonDao reductiebonDao;
    private List<Reductiebon> reductiebonLijst;
    //EXTRA vraag generics
    private MyCollection<Reductiebon> myCollection;

    public ReductiebonBeheerder() {
```

```
        reductiebonDao = new ReductieBonDaoJpa();
        reductiebonLijst = reductiebonDao.findAll();
        myCollection = new MyCollection<>();
    }

    public ReductiebonBeheerder(List<Reductiebon> reductiebonLijst) {
        reductiebonDao = new ReductieBonDaoJpa();
        this.reductiebonLijst = reductiebonLijst;
        myCollection = new MyCollection<>();
    }

    //EXTRA vraag generics

        public void storeBon(Reductiebon bon) {
            myCollection.addElement(bon);
        }
        public Reductiebon retrieveBon() {
            return myCollection.getElement();
        }
        //TODO
        public void addReeks(List<? extends Reductiebon> bonnen) {
            myCollection.addReeks(bonnen);
        }


    public List<Reductiebon> getReductiebonLijst() {
        return Collections.unmodifiableList(reductiebonLijst);
    }

    // VRAAG1
    public List<String> geefReductiebonCodes(int percentage) {
        return reductiebonLijst.stream().filter(bon -> bon.getPercentage() >
percentage)

.map(Reductiebon::getReductiebonCode).collect(Collectors.toList());
    }

    // VRAAG2
    public void sorteerReductiebonnen() {
        reductiebonLijst.sort(Comparator.comparing(Reductiebon::getPercentage)

.thenComparing(Comparator.comparing(Reductiebon::getReductiebonCode).reversed()));
    }

    // VRAAG3
    public double geefGemPercVanBonnenInToekomst()
    {
        return reductiebonLijst.stream().filter(bon ->
bon.getEinddatum().isAfter(LocalDate.now())).
                mapToDouble(Reductiebon::getPercentage).average().getAsDouble();
    }

    // VRAAG4
    public List<LocalDate> geefUniekeEinddatums() {
```

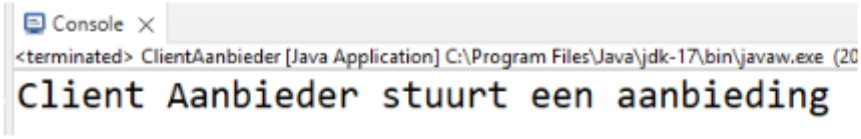
```
        return reductiebonLijst.stream().
                map(Reductiebon::getEinddatum).
                distinct().sorted().
                collect(Collectors.toList());
    }

}
```

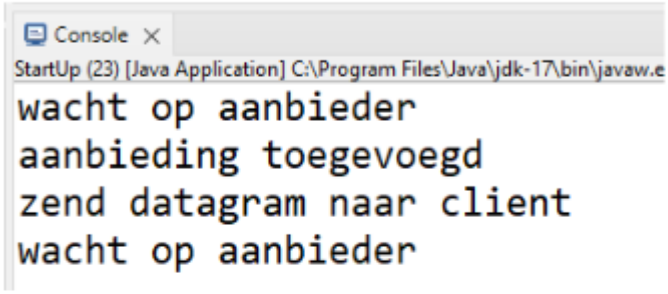
Garage

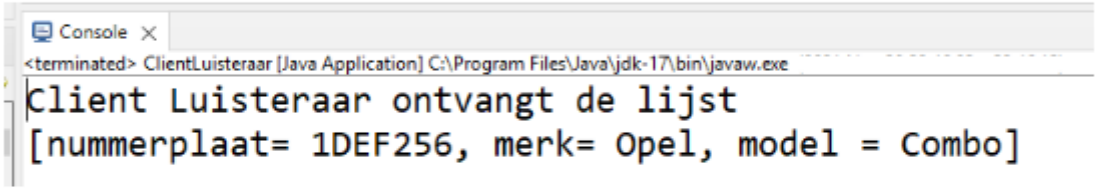
Netwerken

OEFENING GARAGE MET NETWERKSERVICE ASDI_Java_Garage_Start

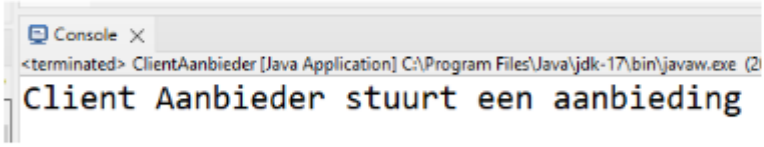
Start **Koopjesservice** op.
Start **ClientLuisteraar** op.
Start **ClientAanbieder** op. Met nummerplaat **1DEF256**.

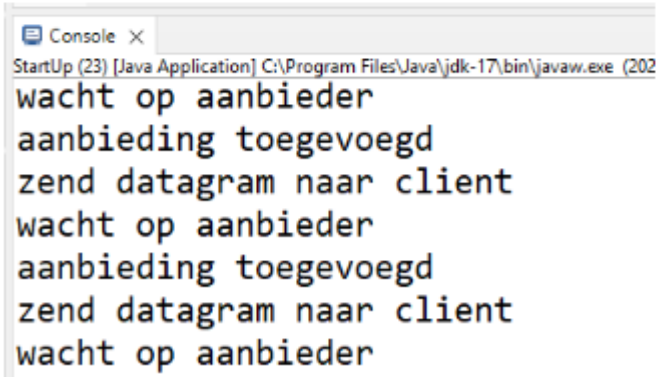


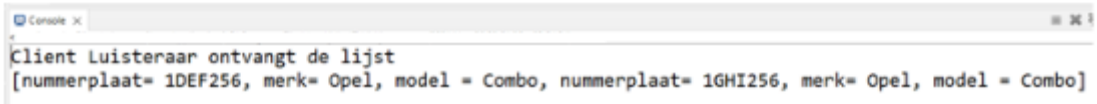




Start opnieuw een **Luisteraar** client op.
Start opnieuw een **Aanbieder** client op. Met nummerplaat **1GHJ256**







Voor de **ClientLuisteraar** en **ClientAanbieder** dien je nog de code aan te vullen voor de netwerk commmunciatie. Beide clients maken slechts één keer verbinding met de server. Zie TODO in beide klassen in het project.

De KoopjesService is al javaThread reeds voorzien. **Je dient de run methode nog aan te vullen voor onderstaande netwerkservice.**

Clients kunnen aanbiedingen via TCP naar de GarageServer doorsturen. De client zal bij connectie met de KoopjesService één auto kunnen doorgeven (een tekstregel met de drie kenmerken Nummerplaat Merk Model). Daarna wordt de verbinding met deze client gesloten.

ClientLuisteraars (dat kan een willekeurig aantal zijn, elk op een andere host uiteraard) kunnen zich bij de GarageService registreren. Dit werken we niet uit en houden voorlopig een lijst van luisteraars bij in de GarageService (voor de oefening is deze lijst dan al opgevuld met localhost als luisteraar).

In de uitvoer console schermen zie je de werking.

Wanneer de Server is opgestart en er één aanbieding gebeurt met nummerplaat **1DEF256**. Vervolgens herstarten we de luisteraar en doen nog eens een aanbieding met nummerplaat **1GHI256**.

De **KoopjesService** wacht op poort **23400** voortdurend op **ClientAanbieders** die telkens één auto doorgeven. De overdracht gebeurt door de data als tekst door te sturen. De nieuwe autoObjecten voor verkoop worden bijgehouden in een lijst bij de **KoopjesService**. Telkens als er een nieuwe aanbieding bijkomt zal de **Koopjesservice** alle geregistreerde **ClientLuisteraars** de volledige lijst van autos doorsturen via een UDP service. Deze lijst zal in onze toepassing enkel de client op localhost bevatten. Het registreren werken we nu niet verder uit. De lijst van autoObjecten wordt als één tekst in een datagram meegeven (toString van de List). **ClientLuisteraars** dienen op poort 9999 aangesproken te worden voor deze service.

Het startproject voor deze oefening is ASDI_Java_Garage_start.

Opgave **oefening Garage** geeft verder aan hoe de werking van de garage zelf dient uitgewerkt te worden. Dat staat verder los van de KoopjesService die de garage voorziet.

```
public class ClientAanbieder {

    public static void main(String[] args) {
        new  ClientAanbieder().run();
    }


    private void run() {
        System.out.println("Client Aanbieder stuurt een aanbieding");
        //TODO Maak een TCP verbinding met localhost op poort 23400
        //     Stuur een regel (= auto in aanbieding) "1DEF256", "Opel", "Combo"
        //     Beeindig de client
        try(Socket s = new Socket(InetAddress.getLocalHost(), 23400)){
            Formatter output = new Formatter(s.getOutputStream());
            output.format("%s %s %s%n", "1DEF256", "Opel", "Combo");
            output.flush();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

```
public class ClientLuisteraar {
    public static void main(String[] args) {
```

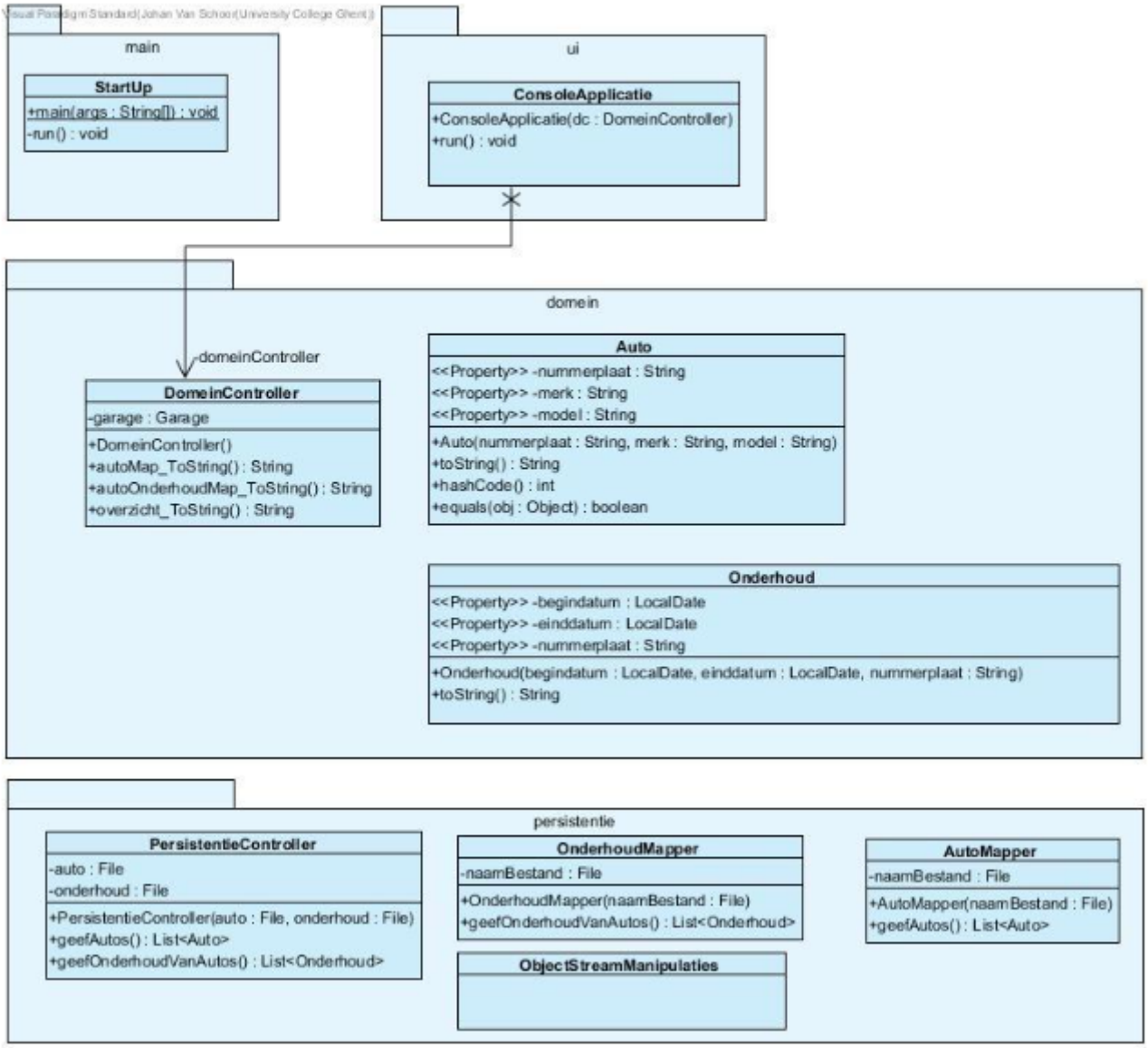
```
        new ClientLuisteraar().run();
    }
    private void run() {
        System.out.println("Client Luisteraar ontvangt de lijst");
        //Gegeven:
        //      Maak een UDP toegang om een pakket te ontvangen op poort 9999
        //      Wacht op het pakket en toon de ontvangen lijst van auto's af op
het scherm
        //      Beeindig de client
        try (DatagramSocket dgSocket = new DatagramSocket(9999)){
            byte[] buffer = new byte[100];
            DatagramPacket dgPacket = new DatagramPacket(buffer, buffer.length);
            dgSocket.receive(dgPacket);
            System.out.println(new String(dgPacket.getData(),
dgPacket.getOffset(), dgPacket.getLength()));
        } catch (IOException ex) {
            ex.printStackTrace();
        }
    }
}
```

```
public class KoopjesService extends Thread {
    private List<Auto> koopjes = new ArrayList<>();
    private List<InetAddress> clients = new ArrayList<>();

    public KoopjesService() { // voorlopig één klant op localhost (dit werken we
niet verder uit, klanten
                              // kunnen zich hier registreren)
        try {
            clients.add(InetAddress.getLocalHost());
        } catch (UnknownHostException e) {
            e.printStackTrace();
        }
    }

    @Override
    public void run() {
        // Gegeven: toegang tot UDP (uitvoer lijst van auto's)
        try (DatagramSocket dgSocket = new DatagramSocket()) {
            // TODO voorzie toegang tot TCP (invoer auto's)
            try (ServerSocket ss = new ServerSocket(23400, 5)) {

                // start lus om telkens de info van één aanbieder te ontvangen
                while (true) {
                    System.out.println("wacht op aanbieder");
                    Socket s = ss.accept();
                    // indien verbinding lees de info
                    Scanner input = new Scanner(s.getInputStream());
                    koopjes.add(new Auto(input.next(), input.next(),
input.next()));
                    System.out.println("aanbieding toegevoegd");
```

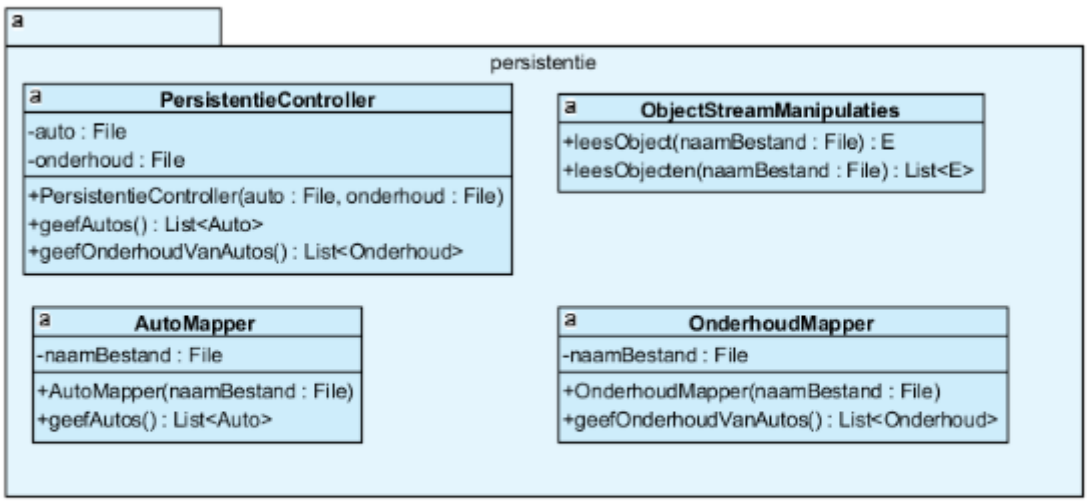
```
                    // beëindig de TCP verbinding
                    s.close();


                // Gegeven: maak pakketje klaar met de lijst van auto's als één
string en
                // verstuur naar alle
                // alle geregistreerde ClientLuisteraars
                DatagramPacket dgPakket = new
DatagramPacket(koopjes.toString().getBytes(),
                        koopjes.toString().getBytes().length);
                clients.forEach(clientAddress -> {
                    System.out.println("zend datagram naar client");
                    try {
                        dgPakket.setPort(9999);
                        dgPakket.setAddress(clientAddress);
                        dgSocket.send(dgPakket);
                    } catch (IOException e) {
                        e.printStackTrace();
                    }
                });
                }
            } catch (Exception e) {
                e.printStackTrace();
            }
            // einde lus
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

Collections - generics - streams

Gegeven: ASDI_Java_Garage_start project (zie chamilo) + twee geserialiseerde bestanden autoObjecten.dat en onderhoudObjecten.dat.



Gevraagd:
Vul de **persistentielaag** aan.

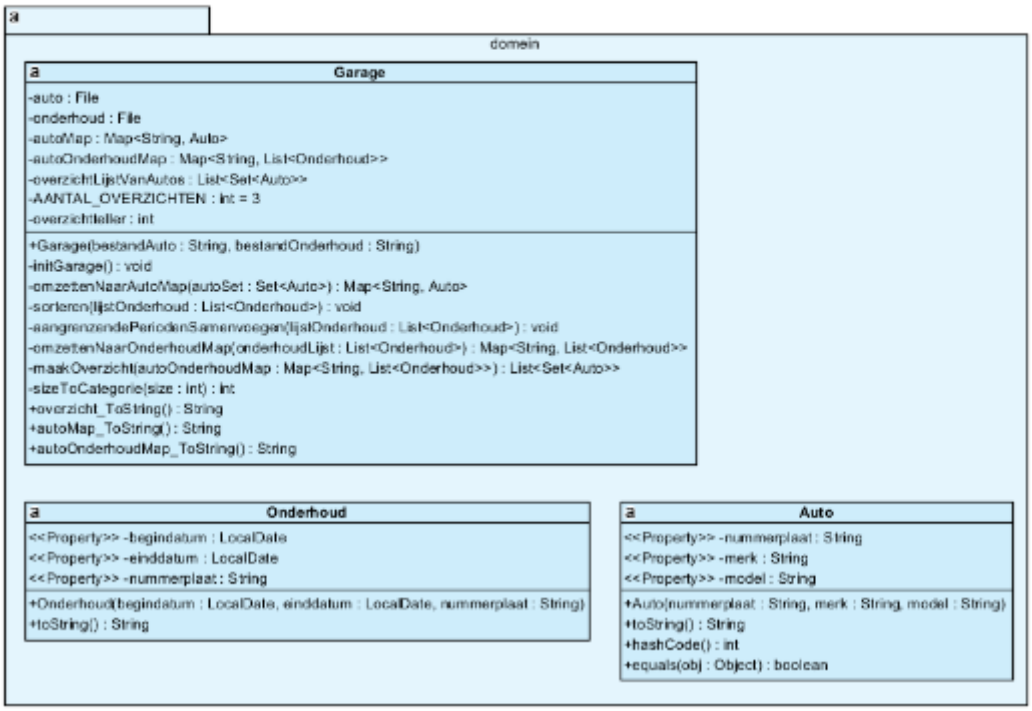


Het bestand "autoObjecten.dat" bevat één object, nl. een List<Auto>.

Het bestand "onderhoudObjecten.dat" bevat meerdere objecten. Onderhoud-objecten werden één per één weggeschreven.

Maak de klasse ObjectStreamManipulaties.
Pas de persistentieController aan die de bestandsnamen krijgt en doorgeeft aan de mapper.
Pas de twee mapper-klassen aan.

Voer in de **domeinlaag** de verschillende stappen uit.



Klasse Garage (domein).

```
public Garage(String autoBes, String onderHoudBes) { …; run(); }

private void run()
{
```

1) Een List<Auto> aan de persistentie vragen en bewaren in een Set<Auto>
   `Set<Auto> autoSet = …`

   Bv. autoSet bestaat uit de volgende auto-objecten
   `{ azerty , BMW, Berline} , {123xyz, Toyota, Yaris}, ...`

2) De "autoSet" omzetten naar een Map. De sleutel is de nummerplaat en de waarde is de overeenkomstige auto. De map wordt als attribuut bewaard.

   Bv. de map wordt

sleutel	**waarde**
azerty	`{ azerty , BMW, Berline}`
123xyz	`{123xyz, Toyota, Yaris}`

3) Een List<Onderhoud> aan de persistentie vragen en bewaren.

   Bv. de lijst bestaat uit de volgende onderhoud-objecten

```
{ 15-12-22, 16-12-22, azerty }
{ 15-12-22, 16-12-22, 123xyz }
{ 20-12-22, 22-12-22, azerty }
{ 10-12-22, 11-12-22, 123xyz }
{ 17-12-22, 19-12-22, 123xyz }
{ 18-12-22, 18-12-22, azerty }
{ 17-12-22, 17-12-22, azerty }
```

4) De lijst van Onderhoud sorteren in stijgende volgorde volgens nummerplaat. Bij gelijke nummerplaten, volgens begindatum (stijgende volgorde).

   Bv.

```
{ 10-12-22, 11-12-22, 123xyz }
{ 15-12-22, 16-12-22, 123xyz }
{ 17-12-22, 19-12-22, 123xyz }
{ 15-12-22, 16-12-22, azerty }
{ 17-12-22, 17-12-22, azerty }
{ 18-12-22, 18-12-22, azerty }
{ 20-12-22, 22-12-22, azerty }
```

5) Per auto de aangrenzende perioden samenvoegen

Bv. De lijst van Onderhoud wordt als volgt gewijzigd:

```
{ 10-12-22, 11-12-22, 123xyz }
{ 15-12-22, 19-12-22, 123xyz }
{ 15-12-22, 18-12-22, azerty }
{ 20-12-22, 22-12-22, azerty }
```

6) De lijst van Onderhoud omzetten naar een Map. De sleutel is de nummerplaat en de waarde is een lijst van Onderhoud (zie vb.). De map wordt als attribuut bewaard.

Bv. de map wordt

sleutel	**waarde (List<Onderhoud>)**
123xyz	{{ 10-12-22, 11-12-22, 123xyz } { 15-12-22, 19-12-22, 123xyz }}
azerty	{{ 15-12-22, 18-12-22, azerty }, { 20-12-22, 22-12-22, azerty }}

7) We vertrekken van de map (6) en we maken een overzicht. We houden drie groepen bij:
- de auto's die één onderhoud hebben
- de auto's die twee of drie onderhouden hebben
- de auto's die meer dan drie onderhouden hebben

We gaan een List "overzichtLijstVanAutos" van Set<Auto> maken.

Bv.
- overzichtLijstVanAutos.get(0) is een lege Set
- overzichtLijstVanAutos.get(1) bestaat uit de Set {123xyz, Toyota, Yaris}, { azerty , BMW, Berline}
- overzichtLijstVanAutos.get(2) is een lege Set

In de klasse Garage voorzie je ook drie toString methoden:

1) public String autoMap_ToString()

De map (2) wordt als één string teruggegeven (zie uitvoer). De nummerplaten zijn in stijgende volgorde.

Uitvoer:

```
autoMap
nummerplaat= 123xyz, merk= Toyota, model = Yaris
nummerplaat= 456abc, merk= Opel, model = Astra
nummerplaat= 567xyz, merk= Renault, model = Fluence
nummerplaat= 789cde, merk= Mercedes, model = C-klasse Berline
nummerplaat= ab12ab, merk= Opel, model = Zafira
nummerplaat= azerty, merk= BMW, model = Berline
nummerplaat= qwerty, merk= Toyota, model = Avensis
nummerplaat= xy12xy, merk= Peugeot, model = 308
```

2) public String autoOnderhoudMap_toString

De map (6) wordt als één string teruggegeven (zie uitvoer). De nummerplaten zijn in stijgende volgorde.

Uitvoer:

```
autoOnderhoudMap
123xyz:
nummerplaat 123xyz van 10-12-2022 t.e.m. 11-12-2022
nummerplaat 123xyz van 15-12-2022 t.e.m. 19-12-2022
456abc:
nummerplaat 456abc van 17-12-2022 t.e.m. 19-12-2022
567xyz:
nummerplaat 567xyz van 15-02-2023 t.e.m. 15-02-2023
789cde:
nummerplaat 789cde van 05-01-2023 t.e.m. 07-01-2023
nummerplaat 789cde van 10-01-2023 t.e.m. 12-01-2023
ab12ab:
nummerplaat ab12ab van 11-01-2023 t.e.m. 11-01-2023
nummerplaat ab12ab van 15-01-2023 t.e.m. 15-01-2023
nummerplaat ab12ab van 17-01-2023 t.e.m. 17-01-2023
nummerplaat ab12ab van 15-05-2023 t.e.m. 15-05-2023
azerty:
nummerplaat azerty van 15-12-2022 t.e.m. 18-12-2022
nummerplaat azerty van 20-12-2022 t.e.m. 22-12-2022
xy12xy:
nummerplaat xy12xy van 08-12-2022 t.e.m. 08-12-2022
nummerplaat xy12xy van 16-12-2022 t.e.m. 20-12-2022
```

3) public String overzicht_ToString()

De lijst (7) wordt als één string teruggegeven (zie uitvoer).

```
overzicht
1
nummerplaat= 567xyz, merk= Renault, model = Fluence
nummerplaat= 456abc, merk= Opel, model = Astra
2
nummerplaat= xy12xy, merk= Peugeot, model = 308
nummerplaat= azerty, merk= BMW, model = Berline
nummerplaat= 123xyz, merk= Toyota, model = Yaris
nummerplaat= 789cde, merk= Mercedes, model = C-klasse Berline
3
nummerplaat= ab12ab, merk= Opel, model = Zafira
```

```java
public class Garage {

    private final File auto;
    private final File onderhoud;
    private Map<String, Auto> autoMap;
    private Map<String, List<Onderhoud>> autoOnderhoudMap;
    private List<Set<Auto>> overzichtLijstVanAutos;

    private final int AANTAL_OVERZICHTEN = 3;
    private int overzichtteller;

    public Garage(String bestandAuto, String bestandOnderhoud) {
        auto = new File(bestandAuto);
        onderhoud = new File(bestandOnderhoud);
        initGarage();
    }
```

```
    private void initGarage() {
        PersistentieController persistentieController = new
PersistentieController(auto, onderhoud);

        // Set<Auto> inlezen - STAP 1
        Set<Auto> autoSet = new HashSet<>(persistentieController.geefAutos());
        System.out.println("STAP 1");

        // Maak map van auto's: volgens nummerplaat - STAP 2
        autoMap = omzettenNaarAutoMap(autoSet);
        System.out.println("STAP 2");
        autoMap.forEach((key, value) -> System.out.printf("%s %s %n", key,
value));

        // Onderhoud inlezen - STAP 3
        List<Onderhoud> onderhoudLijst =
persistentieController.geefOnderhoudVanAutos();
        System.out.println("STAP 3 : " + onderhoudLijst);

        // lijst sorteren - STAP 4
        sorteren(onderhoudLijst);
        System.out.println("STAP 4");
        onderhoudLijst.forEach(System.out::println);

        // lijst samenvoegen - STAP 5
        aangrenzendePeriodenSamenvoegen(onderhoudLijst);
        System.out.println("STAP 5");
        onderhoudLijst.forEach(System.out::println);

        // Maak map van onderhoud: volgens nummerplaat - STAP 6
        autoOnderhoudMap = omzettenNaarOnderhoudMap(onderhoudLijst);
        System.out.println("STAP 6");
        autoOnderhoudMap.forEach((key, value) -> System.out.printf("%s %s%n", key,
value));

        // Maak overzicht: set van auto's - STAP 7
        overzichtLijstVanAutos = maakOverzicht(autoOnderhoudMap);
        System.out.println("STAP 7");
        overzichtLijstVanAutos.forEach(System.out::println);
    }

    // Maak map van auto's: volgens nummerplaat - STAP 2
    private Map<String, Auto> omzettenNaarAutoMap(Set<Auto> autoSet) {
        return autoSet.stream().collect(Collectors.toMap(Auto::getNummerplaat,
Function.identity())); // OF auto -> auto

// ipv

// function.identity
    }

    // lijst sorteren - STAP 4
    private void sorteren(List<Onderhoud> lijstOnderhoud) {
```

```
lijstOnderhoud.sort(Comparator.comparing(Onderhoud::getNummerplaat).thenComparing(
Onderhoud::getBegindatum));
    }

    // lijst samenvoegen - STAP 5
    private void aangrenzendePeriodenSamenvoegen(List<Onderhoud> lijstOnderhoud) {
//java 7
        Iterator<Onderhoud> it = lijstOnderhoud.iterator();
        Onderhoud onderhoud = null;
        Onderhoud onderhoudNext = null;
        while (it.hasNext()) {
            onderhoud = onderhoudNext;
            onderhoudNext = it.next();
            if (onderhoud != null &&
onderhoud.getNummerplaat().equals(onderhoudNext.getNummerplaat())) {
                if
(onderhoud.getEinddatum().plusDays(1).equals(onderhoudNext.getBegindatum())) {//
samenvoegen:
                    onderhoud.setEinddatum(onderhoudNext.getEinddatum());
                    it.remove();
                    onderhoudNext = onderhoud;
                }
            }
        }
    }

    // Maak map van onderhoud: volgens nummerplaat - STAP 6
    private Map<String, List<Onderhoud>> omzettenNaarOnderhoudMap(List<Onderhoud>
onderhoudLijst) {
        return
onderhoudLijst.stream().collect(Collectors.groupingBy(Onderhoud::getNummerplaat));
    }

    // Hulpmethode - nodig voor STAP 7
    private int sizeToCategorie(int size) {
        return switch (size) {
        case 0, 1 -> 0;
        case 2, 3 -> 1;
        default -> 2;
        };
    }

    // Maak overzicht: set van auto's - STAP 7
    private List<Set<Auto>> maakOverzicht(Map<String, List<Onderhoud>>
autoOnderhoudMap) {
        // Hint:
        // van Map<String, List<Onderhoud>>
        // naar Map<Integer, Set<Auto>> (hulpmethode gebruiken)
        // naar List<Set<Auto>>
        return autoOnderhoudMap.entrySet().stream()
                .collect(Collectors.groupingBy(entry ->
sizeToCategorie(entry.getValue().size()), TreeMap::new,
                        Collectors.mapping(entry -> autoMap.get(entry.getKey()),
Collectors.toSet())))
```

```
                .values().stream().collect(Collectors.toList());
    }

//Oefening DomeinController:
    public String autoMap_ToString() {
        String res =
autoMap.values().stream().sorted(Comparator.comparing(Auto::getNummerplaat)).map(A
uto::toString)
                .collect(Collectors.joining("\n"));
        return res;
    }

    public String autoOnderhoudMap_ToString() {
        String res =
autoOnderhoudMap.entrySet().stream().sorted(Map.Entry.comparingByKey())
                .map(e -> String.format("%s: %n%s", e.getKey(),

e.getValue().stream().map(Onderhoud::toString).collect(Collectors.joining("\n"))))
                .collect(Collectors.joining("\n"));
        return res;
    }

    public String overzicht_ToString() {
        overzichtteller = 1;
        String res = overzichtLijstVanAutos.stream()
                .map(setAuto -> String.format("%d%n%s", overzichtteller++,

setAuto.stream().map(Auto::toString).collect(Collectors.joining("\n"))))
                .collect(Collectors.joining("\n"));
        return res;
    }
}
```