

```

1 package be.vdab;
2
3 import be.vdab.schoolgerief.Boekentas;
4 import be.vdab.util.*;
5 import be.vdab.util.mens.Mens;
6 import be.vdab.voertuigen.Personenwagen;
7 import be.vdab.voertuigen.Pickup;
8 import be.vdab.voertuigen.Voertuig;
9 import be.vdab.voertuigen.Vrachtwagen;
10
11 import java.awt.*;
12 import java.io.*;
13 import java.nio.file.Path;
14 import java.nio.file.Paths;
15 import java.util.SortedSet;
16 import java.util.TreeSet;
17
18 import static be.vdab.util.mens.Rijbewijs.*;
19
20 public class Main {
21
22     public static void main(String[] args) throws VolumeException,
23         DatumException {
24         SortedSet<Voertuig> voertuigen = new TreeSet<>();
25         final Volume VOLUME10;
26         final Volume VOLUME12;
27         final Datum datum;
28         final Datum datum2;
29         datum = new Datum(1, 2, 3456);
30         datum2 = new Datum(1, 2, 2134);
31         final int GETAL4=4;
32         final int ZITPLAATSEN_4=4;
33         final String datumString;
34         final String GETAL4_STRING;
35         final int ZITPLAATSEN5=5;
36         final Color KLEUR=Color.PINK;
37
38         // bestuurders
39         Mens BESTUURDER_BBECCE = new Mens("Ammelie", B,BE,C,CE);
40         Mens BESTUURDER_BBE = new Mens("Babette-Emanuella", B,BE);
41
42         /*
43             Creeer een sortedset van voertuigen en voorzie hierin
44             minstens een zestal voertuigen (2
45             personenwagens, 2 pickups en 2 vrachtwagens). Geef ze weer
46             op het scherm.
47             */
48         try {
49             VOLUME10 = new Volume(10, 10, 10, Maat.decimeter);
50             VOLUME12 = new Volume(12, 12, 12, Maat.decimeter);

```

```
48             Personenwagen pw1 = new Personenwagen("opel", datum,
100,ZITPLAATSEN_4, KLEUR, BESTUURDER_BBECCE);
49             Personenwagen pw2 = new Personenwagen("volkswagen",
datum2, 100,ZITPLAATSEN_4, KLEUR, BESTUURDER_BBE);
50             Vrachtwagen vw1 = new Vrachtwagen("Scania", datum2,
46000, 3, VOLUME12, 13500, 3, BESTUURDER_BBECCE);
51             Vrachtwagen vw2 = new Vrachtwagen("DAF", datum, 36000,
3, VOLUME10, 7500, 2, BESTUURDER_BBECCE);
52             Pickup pickup1 = new Pickup("Cadillac", datum, 100,
GETAL4, KLEUR, VOLUME10,BESTUURDER_BBECCE);
53             Pickup pickup2 = new Pickup("Dodge", datum2, 100,
GETAL4, KLEUR, VOLUME12,BESTUURDER_BBECCE);
54             //voertuigen.add(new Vrachtwagen("DAF", datum, 36000,
3, VOLUME10, 7500, 2, BESTUURDER_BBECCE));
55             //voertuigen.add(new Vrachtwagen("Scania", datum2,
46000, 3, VOLUME12, 13500, 3, BESTUURDER_BBECCE));
56             voertuigen.add(vw1);
57             voertuigen.add(vw2);
58
59             voertuigen.add(pw1);
60             voertuigen.add(pw2);
61             voertuigen.add(pickup1);
62             voertuigen.add(pickup2);
63         } catch (Exception e) {
64             e.printStackTrace();
65         }
66
67         System.out.println("Voertuigen: ");
68         for (Voertuig v : voertuigen) {
69             System.out.println(v.toString());
70         }
71
72         /*
73             Bewaar de voertuigen in een bestand wagenpark.dat.
74             */
75         // Opslaan naar bestand
76         File file = new File("/data/wagenpark.dat");
77         try {
78             FileOutputStream fos = new FileOutputStream(file);
79             ObjectOutputStream oos = new ObjectOutputStream(fos);
80
81             oos.writeObject(voertuigen);
82             oos.close();
83         } catch (IOException e) {
84             e.printStackTrace();
85         }
86
87         /*
88             Lees het bestand wagenpark.dat terug in in een sortedset
en geef ze weer op het scherm. Alle
```

```

89             voertuigen zouden terug ingelezen en weergegeven moeten
90             zijn.
91             */
92             SortedSet<Voertuig> voertuigen2 = new TreeSet<>();
93             final Path PATH = Paths.get("/data/wagenpark.dat");
94
95             // lees bestand
96             try {
97                 FileInputStream fis = new FileInputStream(file);
98                 ObjectInputStream ois = new ObjectInputStream(fis);
99
100                voertuigen2 = (SortedSet<Voertuig>) ois.readObject();
101            } catch (IOException e) {
102                e.printStackTrace();
103            } catch (ClassNotFoundException e) {
104                e.printStackTrace();
105            }
106
107            // print geleesde voertuigen uit
108            System.out.println();
109            System.out.println("Uitgeleesde voertuigen: ");
110            //System.out.println(voertuigen2.size());
111            for (Voertuig v : voertuigen2) {
112                System.out.println(v.toString());
113            }
114
115            /*
116             Maak vervolgens enkele boekentas-objecten aan en geef ze
117             weer op het scherm.
118             */
119             Boekentas boekentassen[] = new Boekentas[3];
120             try {
121                 boekentassen[0] = new Boekentas("groen", new Volume(
122                     30, 40, 20, Maat.centimeter));
123                 boekentassen[1] = new Boekentas("rood", new Volume(40
124                     , 50, 30, Maat.centimeter));
125                 boekentassen[2] = new Boekentas("zwart", new Volume(
126                     50, 60, 40, Maat.centimeter));
127             }
128             catch (VolumeException e) {
129                 e.printStackTrace();
130             }
131
132             System.out.println();
133             System.out.println("Boekentassen: ");
134             for (Boekentas b : boekentassen) {
135                 System.out.println(b.toString());
136             }
137
138             /*

```

```
134          Maak een array van het interface type Laadbaar. Vul deze  
135          met enkele voertuig objecten en  
136          boekentas objecten en geef de inhoud van de array weer op  
137          het scherm. Toon tenslotte het totale  
138          laadvolume van deze laadbaar objecten.  
139          */  
140          final Volume VOLUME20 = new Volume(20, 20, 20, Maat.  
141          decimeter);  
142          Pickup pickup1 = new Pickup("Cadillac", datum, 100, GETAL4  
143          , KLEUR, VOLUME20,BESTUURDER_BBECCE);  
144          Vrachtwagen vw1 = new Vrachtwagen("Scania", datum2, 46000  
145          , 3, VOLUME20, 13500, 3, BESTUURDER_BBECCE);  
146  
147          Laadbaar[] laadbareDingen = new Laadbaar[3];  
148          laadbareDingen[0] = boekentassen[0];  
149          laadbareDingen[1] = pickup1;  
150          laadbareDingen[2] = vw1;  
151  
152      }  
153  }
```

```
1 package be.vdab.util;
2
3 public enum Maat {
4     centimeter(1), decimeter(1000), meter(1_000_000);
5     private final int waarde;
6
7     Maat(int waarde) {
8         this.waarde = waarde;
9     }
10
11    public int getWaarde() {
12        return waarde;
13    }
14
15    /*CENTIMETER("centimeter"), DECIMETER("decimeter"), METER(
16     "meter");
17    private final String waarde;
18
19    Maat(String waarde) {
20        this.waarde = waarde;
21    }
22
23    public String getWaarde() {
24        return waarde;
25    } */
26 }
```

```
1 package be.vdab.util;
2
3 import java.io.Serializable;
4 import java.util.Objects;
5
6 public final class Datum implements Serializable, Comparable<Datum> {
7     private static final long serialVersionUID = 1L;
8     private final int dag;
9     private final int maand;
10    private final int jaar;
11
12    public Datum(int dag, int maand, int jaar) throws
13        DatumException {
14        //if (isValidDatum(dag, maand, jaar)) {
15        if (isValidDatum(dag, maand, jaar)) {
16            this.dag = dag;
17            this.maand = maand;
18            this.jaar = jaar;
19        } else throw new DatumException("Datum is niet correct");
20    }
21
22    public int getDag() {
23        return dag;
24    }
25
26    public int getMaand() {
27        return maand;
28    }
29
30    public int getJaar() {
31        return jaar;
32    }
33
34    /*private boolean isValidDatum(int dag, int maand, int jaar) {
35        return (dag >= 01 && maand >= 01 && jaar >= 1583 && dag
36        <= 31 && maand <= 12 && jaar <= 4099);
37    }*/
38
39    private boolean isValidDatum (int dag, int maand, int jaar){
40        if (jaar < 1583 || jaar > 4099 || maand < 1 || maand > 12
41        || dag < 1 || dag > 31) return false;
42        if (maand == 4 || maand == 6 || maand == 9 || maand == 11)
43        return (dag <= 30);
44        else if (maand == 2) return ((dag <= 29 && isSchrikkeljaar
(jaar)) || (dag <= 28 && !isSchrikkeljaar (jaar)));
45        return true;
46    }
47
48}
```

```
45     private boolean isSchrikkeljaar (int jaar) {
46         return (jaar % 4 == 0 && !(jaar % 100 == 0) || jaar % 400
47             == 0));
48     }
49
50     @Override
51     public String toString() {
52         //return dag + "/" + maand + "/" + jaar;
53
54         // formaat : dd/mm/jjjj
55         return String.format("%02d/%02d/%4d", dag, maand, jaar);
56     }
57
58     @Override
59     public int hashCode() {
60         return Objects.hash(dag, maand, jaar);
61     }
62
63     @Override
64     public boolean equals(Object o) {
65         if (this == o) return true;
66         if (o == null || getClass() != o.getClass()) return false;
67         Datum datum = (Datum) o;
68         return dag == datum.dag &&
69                 maand == datum.maand &&
70                 jaar == datum.jaar;
71     }
72
73     @Override
74     public int compareTo(Datum o) {
75         // eerst vergelijken op jaar, dan op maand, dan op dag
76         int vgl = jaar - o.jaar;
77         if (vgl == 0) {
78             vgl = maand - o.maand;
79             if (vgl == 0) vgl = dag - o.dag;
80         }
81         return vgl;
82     }
83 }
```

```

1 package be.vdab.util;
2
3 import java.io.Serializable;
4 import java.util.Objects;
5
6 public final class Volume implements Serializable, Comparable<
7     Volume> {
8     private static final long serialVersionUID = 1L;
9     private final int breedte;
10    private final int hoogte;
11    private final int diepte;
12    private final Maat maat;
13
14    /**
15     * Constructor Volume
16     * @param hoogte
17     * @param breedte
18     * @param diepte
19     * @param maat
20     */
21    public Volume(int hoogte, int breedte, int diepte, Maat maat)
22        throws VolumeException {
23        if (breedte >= 0 && hoogte >= 0 && diepte >= 0) {
24            this.hoogte = hoogte;
25            this.breedte = breedte;
26            this.diepte = diepte;
27            this.maat = maat;
28        }
29        else throw new VolumeException("Volume mag niet negatief
30        zijn");
31    }
32
33
34    public int getBreedte() {
35        return breedte;
36    }
37
38    public int getHoogte() {
39        return hoogte;
40    }
41
42    public Maat getMaat() {
43        return maat;
44    }
45
46    public long getVolume() {
47        return breedte * hoogte * diepte;

```

```
48     }
49
50     /*@Override
51     public boolean equals(Object o) {
52         if (this == o) return true;
53         if (o == null || getClass() != o.getClass()) return false;
54         Volume volume = (Volume) o;
55         return this.getVolume() == volume.getVolume() && maat ==
56         volume.maat;
56     }*/
57
58     @Override
59     public boolean equals(Object o) {
60         if (this == o) return true;
61         if (o == null || getClass() != o.getClass()) return false;
62         Volume volume = (Volume) o;
63         return breedte == volume.breedte &&
64             hoogte == volume.hoogte &&
65             diepte == volume.diepte &&
66             maat == volume.maat;
67     }
68
69     @Override
70     public int hashCode() {
71         return Objects.hash(breedte, hoogte, diepte, maat);
72     }
73
74     /*@Override
75     public int compareTo(Volume o) {
76         return this.maat.compareTo(o.maat);
77     }*/
78
79     @Override
80     public int compareTo(Volume o) {
81         long thisVolume = this.getVolume();
82         long otherVolume = o.getVolume();
83
84         /*switch (this.maat) {
85             case decimeter :
86                 thisVolume *= 1000L;
87                 break;
88             case meter :
89                 thisVolume *= 1000_000L;
90                 break;
91         }*/
92         thisVolume *= this.getMaat().getWaarde();
93
94         /*switch (o.getMaat()) {
95             case decimeter :
96                 otherVolume *= 1000L;
```

```
97                     break;
98         case meter :
99             otherVolume *= 1000_000L;
100            break;
101        }*/
102        otherVolume *= o.getMaat().getWaarde();
103
104        if (thisVolume == otherVolume) {
105            return 0;
106        }
107        return (thisVolume > otherVolume ? 1 : -1);
108    }
109
110    @Override
111    public String toString() {
112        return hoogte + " (h) x" + breedte + " (b) x" + diepte + " (d"
113        ) " + getMaat();
114    }
115
```

```
1 package be.vdab.util;  
2  
3 public interface Laadbaar {  
4     Volume getLaadvolume();  
5     void setLaadvolume(Volume volume);  
6 }  
7
```

```
1 package be.vdab.util;
2
3 public class DatumException extends Exception {
4
5     /**
6      * Default Constructor DatumException
7      */
8     public DatumException() {
9         super();
10    }
11
12    /**
13     * Constructor DatumException
14     * @param message
15     */
16    public DatumException(String message) {
17        super(message);
18    }
19
20    /**
21     * Constructor DatumException
22     * @param message
23     * @param cause
24     */
25    public DatumException(String message, Throwable cause) {
26        super(message, cause);
27    }
28
29    /**
30     * Constructor DatumException
31     * @param cause
32     */
33    public DatumException(Throwable cause) {
34        super(cause);
35    }
36}
37
```

```
1 package be.vdab.util;
2
3 public class VolumeException extends Exception {
4     /**
5      * Default Constructor VolumeException
6      */
7     public VolumeException() { }
8
9     /**
10      * Constructor VolumeException
11      * @param message
12      */
13    public VolumeException(String message) {
14        super(message);
15    }
16
17    /**
18     * Constructor VolumeException
19     * @param message
20     * @param cause
21     */
22    public VolumeException(String message, Throwable cause) {
23        super(message, cause);
24    }
25
26    /**
27     * Constructor VolumeException
28     * @param cause
29     */
30    public VolumeException(Throwable cause) {
31        super(cause);
32    }
33 }
34
```

```
1 package be.vdab.util.mens;
2
3 import java.io.Serializable;
4 import java.util.Iterator;
5 import java.util.Objects;
6 import java.util.Set;
7 import java.util.TreeSet;
8
9 public class Mens implements Serializable, Comparable<Mens> {
10     private String naam;
11     private Set<Rijbewijs> rijbewijzen = new TreeSet<>();
12
13     /**
14      *
15      * param naam
16      */
17     public Mens(String naam) {
18         this.naam = naam;
19     }
20
21     /**
22      *
23      * param naam
24      */
25     public Mens(String naam, Rijbewijs... rijbewijzen) {
26         this.naam = naam;
27
28         for (Rijbewijs rb : rijbewijzen) {
29             this.rijbewijzen.add(rb);
30         }
31     }
32
33     public String getNaam() {
34         return naam;
35     }
36
37     public Set<Rijbewijs> getRijbewijzen() {
38         return rijbewijzen;
39     }
40
41     public Rijbewijs[] getRijbewijs() {
42         Rijbewijs[] rijbewijsArray = new Rijbewijs[rijbewijzen.
size()];
43         return rijbewijsArray = rijbewijzen.toArray(rijbewijsArray
);
44     }
45
46     @Override
47     public String toString() {
48         //return naam + ", " + rijbewijzen;
```

```

49
50         String str = naam;
51
52         if (rijbewijzen.size() > 0) {
53             str += "(";
54             /*for (Rijbewijs rb : rijbewijzen) {
55                 str += rb.toString() + ", ";
56             }*/
57             for (Iterator<Rijbewijs> it = rijbewijzen.iterator();
58                 it.hasNext(); ) {
59                 str += it.next().toString();
60                 if (it.hasNext()) str += ", ";
61                 else str += ")";
62             }
63         }
64         return str;
65     }
66
67     @Override
68     public boolean equals(Object o) {
69         if (this == o) return true;
70         if (o == null || getClass() != o.getClass()) return false;
71         Mens mens = (Mens) o;
72         return Objects.equals(naam, mens.naam) &&
73                Objects.equals(rijbewijzen, mens.rijbewijzen);
74     }
75
76     @Override
77     public int hashCode() {
78         return Objects.hash(naam, rijbewijzen);
79     }
80
81     @Override
82     public int compareTo(Mens o) {
83         int vgl = this.naam.compareTo(o.naam);
84         /*if (vgl == 0) {
85             for (Iterator<Rijbewijs> rb = rijbewijzen.iterator();
86                 vgl == 0 && rb.hasNext() ; ) {
87                 vgl = rb.next().compareTo(o.rijbewijzen.iterator
88                     .next());
89             }
90         }*/
91     }
92 }
```

```
1 package be.vdab.util.mens;  
2  
3 public enum Rijbewijs {  
4     A, B, BE, C, CE, D, DE;  
5  
6     @Override  
7     public String toString() {  
8         String str = super.toString();  
9         if (str.contains("E")) str = str.substring(0,1) + "+" +  
10            str.substring(1);  
11        return str;  
12    }  
13}
```

```
1 package be.vdab.util.mens;
2
3 public class MensException extends RuntimeException {
4     /**
5      * Default Constructor
6      */
7     public MensException() {
8
9
10    /**
11     * Constructor MensException
12     * @param message
13     */
14    public MensException(String message) {
15        super(message);
16    }
17
18    /**
19     * Constructor MensException
20     * @param message
21     * @param cause
22     */
23    public MensException(String message, Throwable cause) {
24        super(message, cause);
25    }
26
27    /**
28     * Constructor MensException
29     * @param cause
30     */
31    public MensException(Throwable cause) {
32        super(cause);
33    }
34}
```

```
1 package be.vdab.voertuigen;
2
3 import be.vdab.util.Datum;
4 import be.vdab.util.Laadbaar;
5 import be.vdab.util.Volume;
6 import be.vdab.util.mens.Mens;
7
8 import java.awt.*;
9
10 /*
11 Pickup
12
13 Deze class is afgeleid van Personenwagen en implementeert Laadbaar
14
15 Voorzie de nodige getters en setter, override de nodige methods.
16 De class heeft een constructor om alle fields te initialiseren.
17 */
18 public class Pickup extends Personenwagen implements Laadbaar {
19     private Volume laadbaarVolume;
20
21     public Pickup(String merk, Datum datumEersteIngebruikName, int
22                   aankoopprijs, int zitplaatsen, Color kleur, Volume laadbaarVolume
23                   , Mens bestuurder, Mens... passagiers) {
24         super(merk, datumEersteIngebruikName, aankoopprijs,
25               zitplaatsen, kleur, bestuurder, passagiers);
26         this.laadbaarVolume = laadbaarVolume;
27     }
28
29
30     @Override
31     public Volume getLaadvolume() {
32         return laadbaarVolume;
33     }
34
35
36     @Override
37     public String toString() {
38         return super.toString() + " " + laadbaarVolume.toString();
39     }
40 }
41
```

```
1 package be.vdab.voertuigen;
2
3 //import org.apache.commons.collections4.IterableUtils;
4
5 import be.vdab.util.Datum;
6 import be.vdab.util.mens.Mens;
7 import be.vdab.util.mens.MensException;
8 import be.vdab.util.mens.Rijbewijs;
9 import be.vdab.voertuigen.div.DIV;
10 import be.vdab.voertuigen.div.Nummerplaat;
11 import org.apache.commons.lang3.StringUtils;
12
13 import java.io.Serializable;
14 import java.util.*;
15
16 public abstract class Voertuig implements Serializable, Comparable<Voertuig> {
17     private static final long serialVersionUID = 1L;
18     private final Nummerplaat nummerplaat = DIV.INSTANCE.getNummerplaat();
19     private String merk;
20     private Datum datumEersteIngebruikName;
21     private int aankoopprijs;
22     private final int zitplaatsen;
23     private Mens bestuurder;
24     private Set<Mens> inzittenden = new TreeSet<>();
25
26     public Voertuig(String merk, Datum datumEersteIngebruikName,
27                      int aankoopprijs, int zitplaatsen, Mens bestuurder, Mens...
28                      passagiers) {
29         setMerk(merk);
30         setDatumEersteIngebruikname(datumEersteIngebruikName);
31         setAankoopprijs(aankoopprijs);
32         if (isValidAantalZitplaatsen(zitplaatsen))
33             this.zitplaatsen = zitplaatsen;
34         else throw new IllegalArgumentException("Aantal
35             zitplaatsen mag niet negatief zijn");
36
37         if (isValidRijbewijs(bestuurder)) {
38             this.bestuurder = bestuurder;
39         }
40
41         for (Mens p : passagiers) {
42             if (!this.bestuurder.equals(p)) {
43                 if (isValidAantalInzittenden(p)) {
44                     inzittenden.add(p);
45                 } else throw new MensException("Is geen valide
46                     passagier, of zitplaatsen zijn al vol");
47             }
48         }
49     }
50 }
```

```
45     }
46
47     abstract Rijbewijs[] getToegestaneRijbewijzen();
48     abstract int getMAX_ZITPLAATSEN();
49
50     public Nummerplaat getNummerplaat() {
51         return nummerplaat;
52     }
53
54     // Merk
55     public String getMerk() {
56         return merk;
57     }
58
59     public void setMerk(String merk) throws
60         IllegalArgumentException {
61         if (isValidMerk(merk)) this.merk = merk;
62         else throw new IllegalArgumentException("Merk mag niet
63         null zijn!");
64     }
65
66     private static boolean isValidMerk(String merk) {
67         return !merk.trim().isEmpty() && !StringUtils.isBlank(merk
68     );
69
70     }
71
72     // Datum eerste ingebruikname
73     public Datum getDatumEersteIngebruikname() {
74         return datumEersteIngebruikName;
75     }
76
77     public void setDatumEersteIngebruikname(Datum
78         datumEersteIngebruikName) throws IllegalArgumentException {
79         if (isValidDatum(datumEersteIngebruikName)) this.
80         datumEersteIngebruikName = datumEersteIngebruikName;
81         else throw new IllegalArgumentException("Datum mag niet
82         null zijn!");
83     }
84
85     private static boolean isValidDatum(Datum datum) {
86         return datum != null;
87     }
88
89     // Aankoopprijs
90     public int getAankoopprijs() {
91         return aankoopprijs;
92     }
93
94     public void setAankoopprijs(int aankoopprijs) throws
95         IllegalArgumentException {
```

```

88         if (isValidAankoopprijs(aankoopprijs)) this.aankoopprijs
89             = aankoopprijs;
90         else throw new IllegalArgumentException("Aankoop prijs
91             mag niet negatief zijn");
92     }
93     private static boolean isValidAankoopprijs(int aankoopprijs)
94     {
95         return aankoopprijs > 0;
96     }
97     // Zitplaatsen (final)
98     public int getZitplaatsen() {
99         return zitplaatsen;
100    }
101    private boolean isValidAantalZitplaatsen(int zitplaatsen) {
102        return zitplaatsen > 0 && zitplaatsen <=
103            getMAX_ZITPLAATSEN();
104    }
105    // Bestuurder
106    public Mens getBestuurder() {
107        return bestuurder;
108    }
109
110    public void setBestuurder(Mens bestuurder) {
111        if /*!this.bestuurder.equals(bestuurder) &&*/(
112            isValidRijbewijs(bestuurder) && isValidAantalInzittenden(
113                bestuurder)) {
114            inzittenden.add(this.bestuurder);
115            this.bestuurder = bestuurder;
116            if (inzittenden.contains((bestuurder))) inzittenden.
117                remove(bestuurder);
118        }
119        else throw new MensException("Teveel man!");
120    }
121
122    //
123    public void addIngezetene(Mens passagier) throws
124        MensException {
125        if (isValidAantalInzittenden(passagier)) {
126            inzittenden.add(passagier);
127        }
128        else throw new MensException("zitplaatsen zijn al op!");
129    }

```

```
130
131     public boolean isIngezetene(Mens passagier) {
132         return inzittenden.contains(passagier) || passagier.
133             equals(bestuurder);
134             //return getIngezetenen().contains(passagier);
135             //return (bestuurder.equals(passagier) || inzittenden.
136             contains(passagier));
137     }
138
139     public Set<Mens> getIngezetenen() {
140         Set<Mens> PassagiersPlusBestuurder = new TreeSet<>(inzittenden);
141         PassagiersPlusBestuurder.add(bestuurder);
142         return PassagiersPlusBestuurder;
143     }
144
145     public Set<Mens> getIngezeteneExclusiefBestuurder() {
146         return inzittenden;
147     }
148
149     private boolean isValidAantalInzittenden(Mens passagier) {
150         if (bestuurder.equals(passagier) || inzittenden.size() +
151             1 < zitplaatsen)
152             return true;
153
154             // Werkt niet?
155             //if (getIngezetenen().contains(passagier)) {
156             if (inzittenden.contains(passagier)) {
157                 return true;
158             }
159
160             /*for (Mens inz : inzittenden) {
161                 if (inz.equals(passagier)) return true;
162             }*/
163
164             return false;
165     }
166
167     private boolean isValidRijbewijs(Mens bestuurder) {
168         if (bestuurder.getRijbewijs().length == 0) {
169             throw new MensException(bestuurder.toString() + " :
170             heeft geen enkel rijbewijs!");
171         }
172         else {
173             for (Rijbewijs geldig : getToegestaneRijbewijzen()) {
174                 for (Rijbewijs chauffeurRB : bestuurder.
175                     getRijbewijs()) {
176                     if(geldig.equals(chauffeurRB)) {
177                         return true;
178                     }
179             }
180         }
181     }
182 }
```

```
174             }
175         }
176     }
177     throw new MensException (bestuurder.toString () + " :
178     ongeldig rijbewijs");
179
180     @Override
181     public boolean equals(Object o) {
182         if (this == o) return true;
183         if (o == null || getClass() != o.getClass()) return false
184     ;
185         Voertuig voertuig = (Voertuig) o;
186         return aankoopprijs == voertuig.aankoopprijs &&
187             zitplaatsen == voertuig.zitplaatsen &&
188             Objects.equals(nummerplaat, voertuig.nummerplaat)
189             &&
190             Objects.equals(merk, voertuig.merk) &&
191             Objects.equals(datumEersteIngebruikName, voertuig
192             .datumEersteIngebruikName) &&
193             Objects.equals(bestuurder, voertuig.bestuurder)
194             &&
195             Objects.equals(inzittenden, voertuig.inzittenden)
196     ;
197     }
198
199     @Override
200     public int hashCode() {
201         return Objects.hash(nummerplaat, merk,
202             datumEersteIngebruikName, aankoopprijs, zitplaatsen, bestuurder,
203             inzittenden);
204     }
205
206     @Override
207     public int compareTo(Voertuig o) {
208         return this.nummerplaat.compareTo(o.nummerplaat);
209     }
210
211     }
212
213     public static Comparator<Voertuig> getMerkComparator() {
214         return new Comparator<Voertuig>() {
215             @Override
216             public int compare(Voertuig o1, Voertuig o2) {
217                 return o1.merk.compareTo(o2.merk);
218             }
219         };
220     }
221
222     public static Comparator<Voertuig> getAankoopprijsComparator(
223     ) {
224         return new Comparator<Voertuig>() {
```

```
215             @Override
216             public int compare(Voertuig o1, Voertuig o2) {
217                 return o1.aankoopprijs - o2.aankoopprijs;
218             }
219         };
220     }
221
222     @Override
223     public String toString() {
224         String str = nummerplaat + " " + merk + " " +
225             datumEersteIngebruikName + " " + aankoopprijs + " ";
226         String strBestuurder = bestuurder.getRijbewijzen().
227             toString().replace("[", "(").replace("]", ")");
228         str += bestuurder.getNaam() + strBestuurder;
229         if (inzittenden.size() > 0) str += " [";
230         for (Iterator<Mens> it = inzittenden.iterator(); it.
231             hasNext();) {
232             str += it.next().getNaam();
233             if (it.hasNext()) str += ", ";
234             else str += "]";
235         }
236     }
```

```

1 package be.vdab.voertuigen;
2
3 import be.vdab.util.Datum;
4 import be.vdab.util.Laadbaar;
5 import be.vdab.util.Volume;
6 import be.vdab.util.mens.Mens;
7 import be.vdab.util.mens.Rijbewijs;
8
9 /*
10 Vrachtwagen
11
12 Deze class is afgeleid van voertuig en implementeert Laadbaar.
13 De class heeft 3 fields laadvolume, de int maximaalToegelatenMassa
   en de int aantalAssen.
14 Voorzie de nodige getters en setter, override de nodige methods.
15 Een vrachtwagen heeft maximum 3 zitplaatsen, anders ontstaat er
   een IllegalArgumentException.
16 */
17 public class Vrachtwagen extends Voertuig implements Laadbaar {
18     private Volume laadvolume;
19     private int maximaalToegelatenMassa;
20     private int aantalAssen;
21
22     public Vrachtwagen(String merk, Datum datumEersteIngebruikName
23 , int aankoopprijs, int zitplaatsen, Volume laadvolume, int
24 maximaalToegelatenMassa,
25             int aantalAssen, Mens bestuurder, Mens...
26             passagiers) {
27         super(merk, datumEersteIngebruikName, aankoopprijs,
28         zitplaatsen, bestuurder, passagiers);
29         setLaadvolume(laadvolume);
30         setMaximaalToegelatenMassa(maximaalToegelatenMassa);
31         setAantalAssen(aantalAssen);
32     }
33
34     @Override
35     Rijbewijs[] getToegestaneRijbewijzen() {
36         //return new Rijbewijs[]{Rijbewijs.D, Rijbewijs.DE};
37         return new Rijbewijs[]{Rijbewijs.B, Rijbewijs.BE};
38     }
39
40     @Override
41     int getMAX_ZITPLAATSEN() {
42         return 3;
43     }
44
45     // Laadbaar
46     @Override
47     public Volume getLaadvolume() {
48         return laadvolume;

```

```
45      }
46
47      @Override
48      public void setLaadvolume(Volume laadvolume) {
49          if (laadvolume == null) throw new IllegalArgumentException(
50              "Volume mag niet null zijn");
51          this.laadvolume = laadvolume;
52      }
53
54      // maximaal toegelaten massa
55      public int getMaximaalToegelatenMassa() {
56          return maximaalToegelatenMassa;
57      }
58
59      public void setMaximaalToegelatenMassa(int
60          maximaalToegelatenMassa) {
61          if (isValidMaximaalToegelatenMassa(maximaalToegelatenMassa))
62              this.maximaalToegelatenMassa = maximaalToegelatenMassa;
63          else throw new IllegalArgumentException("maximaal
64              toegelaten massa mag niet negatief zijn!");
65      }
66
67      // aantal assen
68      public int getAantalAssen() {
69          return aantalAssen;
70      }
71
72      public void setAantalAssen(int aantalAssen) {
73          if (isValidAantalAssen(aantalAssen)) this.aantalAssen =
74              aantalAssen;
75          else throw new IllegalArgumentException("Aantal assen mag
76              niet minder dan twee zijn!");
77      }
78
79      private static boolean isValidAantalAssen(int aantalAssen) {
80          return aantalAssen >= 2;
81      }
82
83      @Override
84      public String toString() {
85          return super.toString() + " assen:" + aantalAssen + ",  

86              maximaal toegelaten massa:" + maximaalToegelatenMassa + ",  

87              laadvolume:" + laadvolume.toString();
88      }
89  }
```

```
1 package be.vdab.voertuigen;
2
3 import be.vdab.util.Datum;
4 import be.vdab.util.mens.Mens;
5 import be.vdab.util.mens.Rijbewijs;
6
7 import java.awt.*;
8
9 public class Personenwagen extends Voertuig {
10     private static final long serialVersionUID = 1L;
11     private static final int MAX_ZITPLAATSEN = 8;
12     private Color kleur;
13
14     public Personenwagen(String merk, Datum
15         datumEersteIngebruikName, int aankoopprijs, int zitplaatsen, Color
16         kleur, Mens bestuurder, Mens... passagiers) {
17         super(merk, datumEersteIngebruikName, aankoopprijs,
18             zitplaatsen, bestuurder, passagiers);
19         setKleur(kleur);
20     }
21
22     @Override
23     protected Rijbewijs[] getToegestaneRijbewijzen() {
24         return new Rijbewijs[]{Rijbewijs.B, Rijbewijs.BE};
25     }
26
27     @Override
28     protected int getMAX_ZITPLAATSEN() {
29         return MAX_ZITPLAATSEN;
30     }
31
32     public Color getKleur() {
33         return kleur;
34     }
35
36     public void setKleur(Color kleur) throws
37         IllegalArgumentException {
38         if (isValidKleur(kleur)) this.kleur = kleur;
39         else throw new IllegalArgumentException("Kleur mag niet
40             null zijn");
41     }
42
43     @Override
44     public String toString() {
45         return super.toString() + " " + getZitplaatsen();
46     }
47 }
```

46 }

47

```
1 package be.vdab.voertuigen.div;  
2  
3 /*  
4 Maak een enum DIV.  
5 Plaats de enum in een package be.vdab.voertuigen.div.  
6 Deze enum is een singleton, dit wil zeggen dat er slechts één  
instance is: INSTANCE;  
7 Maak een method getNummerplaat, die nummerplaat objecten terug  
geeft.  
8 Om de complexiteit rond de nummerplaat te beperken spreken we af  
dat:  
9 - een nummerplaat start met AAA gevuld door 3 cijfers. Je start  
met 001.  
10 TIP: Lees eens de API documentatie van de format method van  
String.  
11 - telkens een nieuwe nummerplaat gevraagd wordt, verhoogd de  
nummer.  
12 - eenmaal aan 999 gekomen mag terug verder gegaan worden met 001.  
13 Zorg ervoor dat DIV nummerplaatobjecten kan maken, maar andere  
classes (buiten het package  
14 be.vdab.voertuigen.div) niet.  
15 */  
16  
17 public enum DIV {  
18     INSTANCE;  
19     private int nummer = 1;  
20  
21     public Nummerplaat getNummerplaat() {  
22         String plaat = String.format("1-AAA-%03d", nummer++);  
23         //if (nummer >= 999) nummer = 1;  
24         if (nummer > 999) nummer = 1;  
25         return new Nummerplaat(plaat);  
26     }  
27 }
```

```

1 package be.vdab.voertuigen.div;
2
3 import org.apache.commons.lang3.StringUtils;
4
5 import java.io.Serializable;
6 import java.nio.file.Path;
7 import java.nio.file.Paths;
8 import java.util.Objects;
9
10 /*
11 Maak een immutable class nummerplaat (immutable → final fields).
12 Plaats de class in een package be.vdab.voertuigen.div
13 Er is één constructor en die aanvaardt een String plaat en heeft
   default visibility. De String bevat minstens één leesbaar teken
   zoniet wordt een IllegalArgumentException geworpen.
14 Voorzie een getPlaat().
15 Voorzie een toString, een equals en een hashCode.
16 Zorg ervoor dat nummerplaten in een OutputStream kunnen bewaard
   worden.
17 Implementeer de interface Comparable.
18 */
19
20 public class Nummerplaat implements Comparable<Nummerplaat>,
21   Serializable {
22     private static final long serialVersionUID = 1L;
23     private final static Path PATH = Paths.get("/data/wagenpark.
24       ser");
25     private final String plaat;
26
27     Nummerplaat(String plaat) throws IllegalArgumentException {
28       if (isValidPlaat(plaat)) this.plaat = plaat;
29       else throw new IllegalArgumentException("plaat mag niet
30         leeg zijn");
31     }
32
33     public String getPlaat() {
34       return plaat;
35     }
36
37     private static boolean isValidPlaat(String plaat) {
38       return !plaat.trim().isEmpty() && !StringUtils.isBlank(
39         plaat);
40     }
41
42     @Override
43     public boolean equals(Object o) {
44       if (this == o) return true;
45       if (o == null || getClass() != o.getClass()) return false;
46       Nummerplaat that = (Nummerplaat) o;
47       return plaat.equals(that.plaat);

```

```
44      }
45
46      @Override
47      public int hashCode() {
48          return Objects.hash(plaat);
49      }
50
51      @Override
52      public String toString() {
53          return plaat;
54      }
55
56      @Override
57      public int compareTo(Nummerplaat o) {
58          return this.plaat.compareTo(o.plaat);
59      }
60  }
61
62
```

```

1 package be.vdab.schoolgerief;
2
3 import be.vdab.util.Laadbaar;
4 import be.vdab.util.Volume;
5 import org.apache.commons.lang3.StringUtils;
6
7 import java.io.Serializable;
8 import java.util.Objects;
9
10 /*
11 Boekentas
12
13 De class Boekentas zit in het package be.vdab.schoolgerief en
14 implementeert Laadbaar.
15 De class heeft twee fields, laadvolume en kleur (een String).
16 Voorzien een constructor met parameters om de fields te
17 initialiseren.
18 Voorzie de nodige getters en setters en override de nodige methods
19 .
20 Zorg ervoor dat Boekentassen in een OutputStream kunnen bewaard
21 worden.
22 Voorzie een toString, equals en HashCode.
23 De equals maak je op basis van de laadvolume en kleur.
24 Laadvolume en kleur moeten ingevuld worden, zoniet wordt een
25 IllegalArgumentException gethrowd.
26 */
27
28 public class Boekentas implements Laadbaar, Serializable {
29     //private final static Path PATH = Paths.get("/data/wagenpark.
30     dat");
31     private static final long serialVersionUID = 1L;
32     private String kleur;
33     private Volume laadVolume;
34
35     public Boekentas(String kleur, Volume laadVolume) throws
36     IllegalArgumentException {
37         setKleur(kleur);
38         if (laadVolume != null) setLaadvolume(laadVolume);
39         else throw new IllegalArgumentException("Laadvolume mag
40         niet leeg zijn");
41     }
42
43     // laadvolume
44     @Override
45     public Volume getLaadvolume() {
46         return laadVolume;
47     }
48
49     @Override
50     public void setLaadvolume(Volume laadVolume) {
51         if (laadVolume == null) throw new IllegalArgumentException

```

```
42 ("Volume mag niet null zijn");
43         this.laadVolume = laadVolume;
44     }
45
46     // kleur
47     public String getKleur() {
48         return kleur;
49     }
50
51     public void setKleur(String kleur) {
52         //if (kleur != null && !kleur.trim().isEmpty() && !
53         //StringUtils.isBlank(kleur)) this.kleur = kleur;
54         if (isValidKleur(kleur)) this.kleur = kleur;
55         else throw new IllegalArgumentException("Kleur mag niet
leeg zijn");
56     }
57
58     private static boolean isValidKleur(String kleur) {
59         return (kleur != null && !kleur.trim().isEmpty() && !
60         StringUtils.isBlank(kleur));
61     }
62
63     @Override
64     public String toString() {
65         return "boekentas " + kleur + " " + laadVolume.toString();
66     }
67
68     @Override
69     public boolean equals(Object o) {
70         if (this == o) return true;
71         if (o == null || getClass() != o.getClass()) return false;
72         Boekentas boekentas = (Boekentas) o;
73         return kleur.equals(boekentas.kleur) &&
74                 laadVolume.equals(boekentas.laadVolume);
75     }
76
77     @Override
78     public int hashCode() {
79         return Objects.hash(kleur, laadVolume);
80     }
81 }
```