Autoworld

Nummerplaat

Maak een class **Nummerplaat** met een membervariabele plaat van het type String. Deze membervariabele is niet wijzigbaar.
Plaats de class in een package *be.vdab.voertuigen.div*
De constructor aanvaardt een String *plaat*.
Voorzie een *getPlaat()*.
Voorzie een *toString()*, een *equals()* en een *hashCode()*.
Zorg ervoor dat nummerplaten in een OutputStream kunnen bewaard worden.
Implementeer de interface Comparable.

DIV

Maak een class **Div**.
Plaats de class in een package *be.vdab.voertuigen.div*.

Deze class is een Singleton en wordt meegeleverd. Deze moet je niet zelf maken.
(Exrta uitleg hierover en de bijgeleverde source-code vindt je terug op de laatste pagina).

Voertuig

Plaats de class **Voertuig** in *be.vdab.voertuigen*.
De class heeft de volgende fields:

- **nummerplaat** (type Nummerplaat): het field nummerplaat wordt bij declaratie onmiddellijk een waarde gegeven en kan later niet meer gewijzigd worden (systeem van Nederland, een nummerplaat wordt toegekend aan een voertuig en niet aan de eigenaar).
- **merk** (type String): kan niet null zijn.
- **aankoopprijs** (type int): kan niet negatief zijn.

Een voertuig heeft een constructor die 2 parameters aanvaardt: een merk en een aankoopprijs.
Voorzie de nodige setters en getters.
Voorzie een *toString()*, *equals()* en *hashCode()*.
De equals() maak je op basis van de nummerplaat.
Implementeer de interface Comparable op basis van nummerplaat.
Zorg ervoor dat voertuigen in een OutputStream kunnen bewaard worden.

Personenwagen

Leid de class **Personenwagen** af van Voertuig.
Plaats de class in *be.vdab.voertuigen*.
De class heeft één field:

- **zitplaatsen** (type int): kan niet 0 of negatief zijn, default-waarde is 1.

Een personenwagen heeft een constructor met parameters om de fields te initializeren.
Voorzie de nodige getters en setters.
Override de nodige methods.

Maat

Creëer een enum Maat met waardes: centimeter, decimeter en meter.
Gebruik hiervoor een enum met een instance variabele en een method.

Volume

Plaats de class **Volume** in *be.vdab.util*.

Een volume heeft 4 fields:

- **breedte** (type int)
- **hoogte**, (type int)
- **diepte** (type int)
- **maat** (type Maat)

De class Volume is immutable, de 4 fields kunnen slechts éénmaal een waarde krijgen (onmiddellijk bij de declaratie of in de constructor). Negatieve volumes kunnen niet en leiden tot een VolumeException.

Een volume heeft een constructor met 4 parameters.
Voorzie een method *getVolume()* die het volume berekent, het resultaat is een long.
Schrijf ook een *equals()* method en bepaal zelf wanneer twee volume-objecten aan elkaar gelijk zijn.
Implementeer de interface Comparable op basis van volume.
Zorg ervoor dat volumes in een OutputStream kunnen bewaard worden.

VolumeException

Plaats de class **VolumeException** in *be.vdab.util*.
De class is afgeleid van Exception. Voorzie naast de default constructor ook een constructor waarbij je een message kan meegeven.
Verder zijn er geen methods nodig.

Laadbaar

Deze **interface Laadbaar** zit in de package *be.vdab.util*.
De interface definieert een getter en een setter voor laadvolume (= een variabele van type Volume).

Pickup

De class **Pickup** is afgeleid van Personenwagen en implementeert Laadbaar. Plaats ze in *be.vdab.voertuigen*.
De class heeft 1 field:

- **laadvolume** (type Volume).

Voorzie de nodige getters en setters, override de nodige methods.
De class heeft een constructor om alle fields te initialiseren.

Vrachtwagen

De class **Vrachtwagen** is afgeleid van Voertuig en implementeert Laadbaar. Plaats ze in *be.vdab.voertuigen*.
De class heeft 3 fields:

- **laadvolume** (type Volume),
- **maximaalToegelatenMassa** (type int) en
- **aantalAssen** (type int).

Waardes hiervoor kunnen niet 0 of negatief zijn.
Voorzie de nodige getters en setters, override de nodige methods.

Boekentas

De class **Boekentas** zit in de package *be.vdab.schoolgerief* en implementeert Laadbaar.
De class heeft twee fields:

- **kleur** (type String)
- **laadvolume** (type Volume).

Voorzie een constructor met parameters om de fields te initialiseren.
Voorzie de nodige getters en setters en override de nodige methods.
Zorg ervoor dat Boekentassen in een OutputStream kunnen bewaard worden.
Voorzie een *toString()*, *equals()* en *hashCode()*.
De *equals()* maak je op basis van laadvolume en kleur.
Laadvolume en kleur moeten ingevuld worden, zoniet wordt een IllegalArgumentException gethrowd.

Main programma

Creëer een sortedset van voertuigen en voorzie hierin minstens een zestal voertuigen (2 personenwagens, 2 pickups en 2 vrachtwagens). Geef ze weer op het scherm.

Bewaar de voertuigen in een bestand *wagenpark.dat*.

Lees het bestand *wagenpark.dat* terug in in een sortedset en geef ze weer op het scherm. Alle voertuigen zouden terug ingelezen en weergegeven moeten zijn.

Maak vervolgens enkele boekentas-objecten aan en geef ze weer op het scherm.

Maak een array van het interfacetype Laadbaar. Vul deze met enkele voertuig objecten en boekentas objecten en geef de inhoud van de array weer op het scherm. Toon tenslotte het totale laadvolume van deze laadbaar objecten.

Class Div:

Van deze class maken we een Singleton. Dit wil zeggen dat er slechts één instantie kan worden aangemaakt van deze class. Er zijn verschillende manieren om met Singletons te werken. De eenvoudigste manier is het aanmaken van een ENUM met slechts één instantie. De gebruiker wordt dan verplicht om telkens te werken met deze ene instantie. Via deze instantie kunnen dan alle publieke methodes en eigenschappen aangeroepen worden.

Methode *getNummerplaat()*:
Om de complexiteit rond de nummerplaat te beperken gelden volgende regels:

- een nummerplaat start met 1-AAA- gevolgd door 3 cijfers. Je start met 001.
- telkens een nieuwe nummerplaat gevraagd wordt, wordt het nummer verhoogd.
- éénmaal aan 999 gekomen, mag terug verder gegaan worden met 001.

Code:

```
package be.vdab.voertuigen.div;

public enum Div {
    INSTANCE;  //stelt het enige object voor
    private int nummer=1;

    public Nummerplaat getNummerplaat(){
        String plaat = String.format("1-AAA-%03d",nummer++);
        if (nummer>=999){
            nummer=1;
        }
        return new Nummerplaat(plaat);
    }
}

// String.format("1-AAA-%03d",nummer++):
// %03d% wordt ingevuld met de waarde van de variabele nummer
// (deze wordt steeds met 1 verhoogd door de ++)
// 3 betekent dat het getal uit 3 cijfers bestaat (van 001 tot 999)
// 0 betekent dat voorloopnullen toegevoegd worden waardoor je 1-AAA-001
// bekomt en niet 1-AAA-1
```

Met de volgende regel code kan je in de betreffende class dan een nummerplaat genereren:

```
private final Nummerplaat nummerplaat = Div.INSTANCE.getNummerplaat();
```