**CarDockGarage**

Maak een Nieuw project aan voor deze oefening

In dit project maak je voor nu 4 Packages aan:

1) Garage Package: waar je alle Garage en Auto klasses in aanmaakt

2) Interface packes waar je alle interfaces gaat in onder bergen

3) enum Packages

4) app Packages waar je alle verschilende mains zal in onder bergen

**Opgave1**

Begin met het uit werken van de uml en voorzie alle klasses van de nodige constructors en methodes

Maak Hierna een garage object aan met een maximum capaciteit van 80 auto's.

Opdracht1:

maak zelf 10 Autos aan en plaats ze in de garage. Dit mag je doen met gelijk welke Methode

opdracht2: maak 40 Autos aan met behulp van een random genator. Zowel kleur als random acceleratie snelheid. (ze komen van de boot gerold)

tips:modulo, switch of if else

opdracht3: maak 100 auto's aan en probeer deze aan de garage toe te voegen

opdracht4: vang het probleem uit opdracht 3 op met exceptions en een vriendelijke boodschap terug naar de user

maak een 5de packages aan met Alle eigen Exceptions

<https://www.geeksforgeeks.org/exceptions-in-java/>  
 <https://www.geeksforgeeks.org/types-of-exception-in-java-with-examples/>  
 HeatException Example op github

**Opgave2**

JAVADOCS

Becomentarieer je eigen project en maak met behulp van javadocs een website voor aan.

Voeg aan de main een auteur, versie en beschrijving van je programma toe.

Doe dit ook voor de klassen en voorzie ook de constructors en de methoden van de nodige Comentaar en annotaties.

Genereer de javadoc aan de hand van je IDE.

Vind deze bestanden en ga eens zien naar de structuur van gegenereerde bestanden. Bekijk deze en navigeer door je documentatie en zie waar er nog eventuele fouten zitten.

**Opgave 3 finalize**

Bekijk de cabriolet oefening.

1) Implementeer een count-- in de finalize methodes en bekijk wat deze doet met de counters

2) Implementeer een betrouwbare manier om het aantal autos te gaan bijouden die op de dokken staan.

tip: loops en !null

<https://www.tutorialspoint.com/java/lang/object_finalize.htm>

<https://www.geeksforgeeks.org/g-fact-24-finalfinally-and-finalize-in-java/>

**Opgave 4**

Equals en HashCode Implementeren

We gaan een equals methode implementeren op basis van het kleur en de naam van het autoObject. Bij het implementeren van de equasl methode is het ook altijd aangeraden de hashCode methode te gaan overschrijven. Hier zullen we dat doen aan de hand van een berekening van de snelheid maal optelling van de RGB waarden en

zie:

[**https://www.geeksforgeeks.org/equals-hashcode-methods-java/**](https://www.geeksforgeeks.org/equals-hashcode-methods-java/)

**Heeeeel duidelijk filmpje:**

[**https://www.youtube.com/watch?v=7V3589CReug&feature=youtu.be&fbclid=IwAR04MvyDyFR\_7nzHwdptGybVT1CfSnHDDXV8-rmgjrc2YNcasqmVS6Fy4M8**](https://www.youtube.com/watch?v=7V3589CReug&feature=youtu.be&fbclid=IwAR04MvyDyFR_7nzHwdptGybVT1CfSnHDDXV8-rmgjrc2YNcasqmVS6Fy4M8)

**Opgave 5**

Maak een nieuwe mainclasse aan in je app package: LokalClassCarsApp

We krijgen in de garage een lading van prototype wagens binnen. Maak een loop zodat er aan de hand van een lokale klasse in de methode 10 prototype autos worden aangemaakt.

Deze auto's zijn allemaal standaard wit en hebben een extra boolean wrapped.