# Inleiding

In dit document vind je één of meerdere challenges die passen bij het onderwerp **Inheritance**. Je kunt de challenges gebruiken zoals gegeven, of gebruiken als voorbeeld en inspiratie voor challenges die je zelf opstelt.

## Challenge 1, deel 1 van 2

In een dierenasiel worden huisdieren opgevangen. Dit kunnen gevonden huisdieren zijn, of huisdieren waarvan de eigenaar afstand wil doen.

Voor een asiel moet een applicatie gemaakt worden, waarin gegevens over de huisdieren worden bijgehouden. Zo wil men snel een overzicht kunnen krijgen van alle dieren in het asiel en wil men per dier gegevens op kunnen vragen. Indien een nieuw huisdier bij het dierenasiel binnenkomt, moet deze natuurlijk in het systeem kunnen worden ingevoerd. Een huisdier moet kunnen worden verkocht aan een nieuwe eigenaar, etc..

In het asiel worden honden en katten opgevangen. De medewerkers van het dierenasiel proberen iedere hond één keer per dag uit te laten. Het uitlaten wordt bij de honden dan ook geregistreerd door voor iedere hond de datum bij te houden wanneer deze hond voor het laatst is uitgelaten. Katten worden niet uitgelaten, maar van katten wil men extra informatie over hun gedrag bij kunnen houden. Men kan dan denken aan informatie, zoals "Is erg bazig", "Valt andere katten aan", "kan slecht alleen zijn" en "Niet geschikt voor kinderen". Verder kan een toekomstige eigenaar een huisdier reserveren. Zo weet hij zeker dan iemand anders er niet met zijn favoriete dier vandoor gaat.

In de toekomst hoopt men ook andere dieren op te kunnen vangen, maar voorlopig alleen honden en katten.

In deze opgave gaan we een eerste versie voor het administratie systeem maken. Dit systeem kan maar één hond of één kat administreren.

Om het systeem te implementeren gebruiken we:

* de klasse Animal (is compleet)
* de klasse Dog en Cat (zijn niet af)
* de GUI klasse AdministrationForm (is niet af).

Het klassediagram van de code:



**Opgaven**

1. Zie directory ”AnimalShelter” en lees de code en het commentaar door.
2. Maak de klassen Dog en Cat af. Zie TODO’s in het commentaar van de code.
3. Kun je in het bovenstaande klassediagram het symbool voor inheritance vinden? Kun je deze ook terug vinden in de code? Waar?
4. Gebruik de AdministrationForm klasse voor het maken van een user interface van waaruit de gebruiker een hond of een kat aan de administratie toe kan voegen.  
   De administratie kan (en moet) op het moment maar één hond of één kat bevatten, zie het dataveld animal.  
   Er staan reeds een aantal knoppen en een combobox op het form. Zie de TODO’s in de code om te zien wat je hier mee moet doen.  
   Plaats zelf nog textboxes en labels voor het invoeren en tonen van informatie.
5. Ga na of je de properties van de Animal klasse kunt benaderen van een kat of een hond die je aan het animal dataveld hebt toegewezen.   
   Kun je ook de specifieke properties van de hond (LastWalkDate) en de kat (BadHabits)benaderen vanuit het animal dataveld? Waarom wel/niet?
6. Test je code.

## Challenge 1, deel 2 van 2

In deel 1 van challenge 1, heb je de basis gelegd voor een administratie van een dierenasiel. De administratie had de beperking dat er maar één dier in opgeslagen kon worden. Niet erg praktisch. De klant heeft het proof of concept gezien en wil verder met de volgende stap. Hij wil meerdere dieren in de administratie op kunnen slaan. Ook wil hij dieren in de administratie terug kunnen vinden met het registratienummer van de chip, dieren kunnen verwijderen en dieren kunnen toevoegen. Verder moet men de reservering van een dier aan kunnen passen.

Het klassediagram wordt nu:



**Opgaven**

1. Kopieer je solution van opgave 2.
2. Maak de klasse Administration zoals in het klassediagram.

Hierbij een beschrijving van de methoden:

**Add(animal: Animal): bool**

Deze methode voegt een dier toe aan de administratie. Een dier mag alleen toegevoegd worden als er nog geen ander dier is met hetzelfde chipnummer.

De returnwaarde geeft aan of het toevoegen gelukt is.

**RemoveAnimal(chipRegistrationNumber: int): bool**

Deze methode verwijdert het dier met het opgegeven chipnummer. Indien er geen dier is met het opgegeven chipnummer dan wordt er niets gedaan.

De returnwaarde geeft aan of het verwijderen gelukt is.

**FindAnimal(chipRegistrationNumber: int): Animal**

Deze methode zoekt het dier met het opgegeven chipnummer.

De returnwaarde bevat het dier indien gevonden of null indien niet gevonden.  
  
Let op: Het is **NIET** toegestaan List<T>.Find(…) te gebruiken of andere Collection methoden met labda expressies, delegates of anonieme methodes als parameter. Ook het gebruik van linq is niet toegestaan. Het doel is dat je laat zien dat je zelf een eenvoudig zoek algoritme kan maken met een loop.

1. Gebruik de Administration klasse in de AdministrationForm klasse zodat je vanuit de form de administratie met dieren kunt gebruiken.  
    Je eerste stap moet hierbij zijn het verwijderen van het dataveld animal uit de AdministrationForm klasse. Gebruik de Administration in plaats daarvan, zoals aangegeven in het klassediagram.   
   Pas vervolgens de user interface en de code van de AdministrationForm klasse aan, zodat de gebruiker de volgende taken kan uitvoeren met de form.
   * Een hond of een kat toevoegen aan de administratie.
   * Een hond of een kat verwijderen uit de administratie.
   * Een hond of kat reserveren of vrijgeven voor reserveren.

De user interface moet te allen tijde de dieren in de administratie op het scherm tonen.

Dit moet gebeuren in 2 lijsten: één met gereserveerde dieren en één met niet gereserveerde dieren.

1. Zorg ervoor dat er bij het opstarten van de applicatie al huisdieren in de administratie zitten.
2. Natuurlijk maak je unit tests voor alle gemaakte klassen. Zo ook deze keer.
3. Test je code.

## Challenge 2

Denk terug aan de case ‘A car rental went bad’, die tijdens de workshop behandeld is.

Tijdens het uitwerken van deze case heb je wijzigingen aangebracht op de code van het ‘CarRentalWentBad’ project.

In de code van het ‘CarRentalWentBad’ project is geen inheritance gebruikt, waardoor er veel code gedupliceerd moet worden. Tijdens het uitwerken van de case heb je ondervonden dat dit de onderhoudbaarheid niet ten goede komt.

Opdracht:

* Voeg aan de code inheritance toe voor de auto’s die in het project aanwezig zijn. Zorg ervoor dat de RentalAdministration klasse en de GUI met deze nieuwe code kunnen werken. Zie onderstaand klassediagram voor de specificatie van de klassen.



* Codeer nu opnieuw de 3 klantwensen uit de case ‘A car rental went bad’:
  + De nummerborden van de auto’s moeten alle karakters kunnen bevatten. Niet alleen cijfers zoals in de huidige situatie.
  + Van auto’s die aan de administratie toegevoegd worden, moet het bouwjaar vastgelegd worden. In de lijsten met auto’s moet deze ook zichtbaar zijn.
  + Ook auto’s van het type ‘Offroad’ moeten in de verhuur. Van een offroad auto moet vastgelegd worden: de fabrikant, het model, het bouwjaar, de nummerplaat, het aantal gereden kilometers, heeft hij 4 wiel aandrijving ja/nee, de verhuurdatum en is hij beschikbaar voor verhuur. De verhuurkosten zijn 170 euro per dag vanwege het verhoogd risico van off-road rijden. Per gereden kilometer moet 35 cent berekend worden.