**Træningsopgave (SWC – jan2021)**   
  
Dette opgavesæt er ment som et træningssæt til prøven I januar. Derfor er der fokus på Consol Apps, Klasser, Collections (List/Dictionary) Løkker (specielt foreach) Exceptions og betingede sætninger (conditional statements - if/else konstruktioner).  
  
Sættet er ment som træning dvs til at få ”speedet” din kode hastighed op, derfor er første punkt:   
  
**Check din Resharper!**  
  
  
**Class Car**

1. Åben Visual Studio 2019 og lav et nyt projekt at typen Console App (.NET Core) kald det *CarConsoleApp*
2. Lav en klasse Car med følgende properties:
   1. *String RegNo*
   2. *String Model*
   3. *Int Hk*

Klassen skal desuden have 2 konstruktører en default (uden parametre) og en der kan tage parametre til at initialisere alle properties i klassen. Der skal også være en ToString metode.

**Hint:**

* **PROP ->** (til property)
* **CTOR ->** (til default constructor)
* **CTORP ->** (til constructor med properties som parametre)
* **Alt ins - Formating Members**

1. Slet klassen og gentag til du kan lave klassen på **max 4 minutter !!!**
2. I Main opret mindst 3 objekter og udskriv objekterne vha ToString()  
   Eksperimenter med følgende:  
   1. Opret objektreferencer der peger på objekterne og brug referencen til at udskrive
   2. Opret objekter ”on the fly” uden referencer og udskriv
   3. Opret objekter med default constructor og tildel værdier til properties efterfølgende
   4. Prøv at udskrive både med og uden kald explicit kald af ToString()

**Hint**   
Brug minimum ½ time til at eksperimentere/lege med sætninger ala:

* Car car1 = new Car(“XX12345”, “BMW”, 150);
* Car car2 = new Car(){RegNo=”XX12345”, Model=”BMW”, Hk=150};
* Car car3 = new Car(); car3.RegNo=”XX12345”; car3.Model=”BMW”; car3.Hk=150;
* Console.WriteLine(car1.ToString());
* Console.WriteLine(car1);
* Console.WriteLine(new Car(“XX12345”, “BMW”, 150));

**Class CarCatalog**

Lave en class **CarCatalog** :

1. Klassen skal indeholde en collection af **Car** objekter (*List<Car>* eller *Dictionary<string, Car>*)
2. Konstruktøren til **CarCatalog** skal initialisere din collection (bemærk ordvalget)
3. **CarCatalog** skal have en metode *AddCar(string regNo, string model, int hk)* der kan oprette at Car objekt og tilføje det til collection
4. **CarCatalog** skal også have en metode *AddCar(Car car)* der kan tage et car-objekt og tilføje til collection
5. Metoden *AddCar(..)* er overdefineret (overloadet) hvad betyder det?? Og hvordan virker det??  
   Find ud af svaret og skriv det ned i dine noter!
6. **CarCatalog** skal have en metode *PrintCars()* der kan udskrive alle car-objekterne i cataloget.
7. **Main** – test/afprøv klassen ved at oprette et objekt af **CarCatalog** og tilføj min 5 car-objekter. Afprøv begge *AddCar()* metoder og afprøv *PrintCars()*

Udvidelse af **CarCatalog:**

1. **CarCatalog** skal have en metode *GetCar(string regNo).*

Metoden skal returnere car-objektet med det givne regNo og null, hvis der ikke findes en bil med dette registreringsnummer (Hint: brug en foreach-løkke til at gennemløbe din collection).

**Hint**

* List<Car> carList;  
  CarCatalog(){ carList = new List<Car>();}
* Dictionary<string, Car> carDictionary;  
  CarCatalog(){ carDictionary = new Dictionary<string, Car>();}

1. **Main** – test/afprøv *GetCar()*
2. Prøv at udskifte din foreach-løkke med en almindelig for-løkke samt en while-løkke  
   Hvad er forskellen? Fordele/ulemper? - Forklar og skriv ned i dine noter!!
3. **CarCatalog** skal have en metode *DeleteCar(string regNo)* der kan slette en car fra cataloget (collection) metoden skal returnere det slettede car-objekt eller null, hvis det ikke findes.  
     
   Hvad er et break og kan det benyttes her?
4. **Main** – test/afprøv *DeleteCar()*

**Svær:**

1. **CarCatalog** skal have en metode *FindAllModels(string model)* metoden skal returnere en liste med alle Car af den givne model.   
   (Overvej om der skal returneres en tom liste eller null, hvis der ikke er nogle biler af den givne model i collection)
2. **Main** – test/afprøv *FindAllModels(string model)*
3. **CarCatalog** skal have en metode *FindMaxHk()*, metoden skal returnere det Car-objekt der har højeste Hk-værdi   
   (Hvad hvis der er flere der har samme værdi? Skal man så returnere den første, den sidste eller dem alle dvs som liste? – hvad betyder dit valg for returtypen for metoden?)
4. **Main** – test/afprøv *FindMaxHk()*

**Exceptions**

Du skal nu ”refaktorerer” dvs ændre din kode så der kastes Exceptions, hvis der prøves på at oprette Car-objekter med ”ulovlige” værdier.  
  
Det kan gøres på mange måder, her vælger vi at ”lege” med Exceptions ved at lade metoden *AddCar()* kaste en *Exception*, hvis *RegNo* ikke er en streng med længden 7!

1. Refaktorer (opdater) metoden *AddCar()* så den kaster en ny *ArgumentException*, hvis *RegNo* ikke er 7 karakter lang. Der skal være en passende ”fejlmeddelelse”.  
   (**Hint** *RegNo* er en String og String har property *Length* der angiver antal karaktere i strengen)
2. **Main** – test/afprøv AddCar() – vis at der kastes en ArgumentException ved ulovligt RegNo  
   Benyt Try/Catch.
3. Refaktorer AddCar() metoden så der kastes ArgumentExceptions ved andre ”ulovlige” værdier  
   fx Hk < 0 eller Hk > 2000 – find selv på noget og afprøv!!

**Arv**Du skal nu udvide dit program med nedarvning. Du skal lave en klasse **StationCar** der arver fra klassen **Car**. StationCar skal have en property *Volume* til at registrere størrelsen på bagagerummet.  
  
**Hint**   
Når der arvers angives det med ”:” (class StationCar : Car)  
Når base-class delen (dvs properties i class Car) skal initialiseres, gøres det ved at kalde *base(…)* (kald af base-class’ens konstruktør)

*public StationCar(string regNo, string model, int hk, int volume) : base(regNo, model, hk)*

*{*

*Volume = volume;*

*}*

Afprøv i Main og find selv på mere at ”lege” med!  
  
  
**Next**  
Når du er færdig laver du det samme med *Planter* i morgen og med *Cykler* i overmorgen og med *Computer* i overovermorgen og med …. I overoverover….. ….