Færdighedsprøve - AutoTeknik

D. 22 januar 2024 kl. 9.00 – 13.00

Regler for eksamen…
Der må anvendes alle hjælpemidler til eksamen undtagen **alle** former for kunstig intelligens, herunder Chat GPT og Co-pilot.

Domæne beskrivelse
Autoteknik er en mindre kæde af små selvstændige automobilværksteder, som har brug for et nyt system til at registrere de reparationer, som deres mekanikere laver på værkstederne. Det endelige system skal kunne håndtere oplysninger om køretøjer, bilejere, reparationer og mekanikere.

I systemet skal der oprettes en klasse **Car**, der indeholder oplysninger om det enkelte køretøj. Klassen skal indeholde følgende properties:

Property / Attributter	***Type***	***Beskrivelse***
Id	int	Unikt Id på Car-objekter
RegNo	string	Bilens registreringsnummer eks. "**AX23577**"
Year	int	Det årstal, hvor bilen blev produceret, eks. **2013**
Make	string	Bilens mærke, eks. **Tesla**
Model	string	Bilens model. Tesla har en model der hedder "**Model Y**"
Mileage	int	Kilometertal. Antal km bilen har kørt, eks. **220000**

Bilerne skal registreres i et **CarRegister,** hvor der ligeledes registreres navnet på ejeren af værkstedet og værkstedets cvr nummer. Klassen skal indeholde en **liste** over automobiler af typen **List<Car>**.

Property / Attributter	***Type***	***Beskrivelse***
Name	string	Værkstedets ejer, "**John Sharp**"
Cvr	int	Unikt virksomhedsnummer, eks. **51761471**

Udviklingen af systemet sker igennem 4 iterationer. For hver iteration skal der tegnes nogle UML diagrammer indenfor analyse og design og en mindre del at systemet skal implementeres i C#. Inden for hver iteration kan opgaverne laves i en "valgfri rækkefølge".

Tegningen af diagrammerne kan laves på papir og der skal så sættes et billede ind i det dokument, der afleveres. Det er også tilladt at tegne vha. Visio eller lignende system.

Iteration 1
I denne iteration skal der modelleres, designes og implementeres en mindre del af systemet, som omhandler oprettelse af klasserne **Car** og **CarRegister**.

A. Tegn en Domænemodel omhandlende iteration 1.

B. Tegn et Design-klassediagram.

C. Skriv en User story med tilhørende acceptance kriterier der omhandler denne del af systemet. Der kan være mange US til denne del af systemet (fx oprettelse af nyt "værksted" eller ny "bil"), men du skal kun skrive en.

D. Opret et nyt .NET Core console Application projekt og kald det **Autoteknik**

E. Implementer klassen **Car**

    a. Klassen skal indeholde relevante instans felter, properties og konstruktør(er) samt en ToString() metode.

    b. Test klassen ved at oprette nogle instanser af den i Program.cs og skriv dem ud til konsollen med Console.WriteLine

F. Implementer klassen **CarRegister**
    a. Klassen skal indeholde relevante instans felter, properties og konstruktør(er) samt en ToString() metoden (incl alle listens Car-objekter)

    b. Implementer en metode **AddCar**, der kan tilføje et **Car** objekt til en liste af **Car** objekter (overvej parametre og return type).

    c. Implementer en metode **DeleteCar**, der kan fjerne et **Car** objekt fra listen (overvej parametre og return type).

    d. Test klassen ved at oprette en instans af **CarRegister** i Program.cs og skriv **Car** objekterne ud til konsollen med Console.WriteLine.

Formateringen af out er ikke af betydning.

Iteration 2
I denne iteration skal der modelleres, designes og implementeres den del af systemet, som omhandler oprettelse af en reparation, **AutoRepair**, og dennes tilknytning til et **Car** objekte. Til en bil kan der forekomme mange reparationer.

Klassen **AutoRepair**

Property / Attributter	***Type***	***Beskrivelse***
Id	int	Unikt id der identificerer reparationen, eks. **101**
Description	string	Mekanikerens beskrivelse af skaden. Eks. **Bremseklodser skiftet på højre og venstre forhjul**"
Price	double	Prisen i kroner på reparationen, eks. **50300,50**

G. Udvid og opdater Design-klassediagrammet.

H. Implementer klassen **AutoRepair**
   a. Klassen skal indeholde relevante instans felter, properties og konstruktør(er) samt en ToString() metode.

   b. Test klassen ved at oprette nogle instanser af den i Program.cs og skriv dem ud til konsollen med Console.WriteLine.

Hver bil skal indeholde et *Dictionary* af objekter af typen **AutoRepair**, med denne type definition **Dictionary<int, AutoRepair>.**

Id i klassen **AutoRepair** anvendes som key i dictionary'et.

I. Udvid implementationen af klassen **Car,** så der til et Car objekt kan tilføjes objekter af typen **AutoRepair**.

   a. I klassen Car skal der implementeres en metode **AddAutoRepair**.

   b. Overvej returtype og hvilke parametre der er nødvendige.

c. Udvid ToString metoden i Car klassen, så den kan returnere alle reparationer tilknyttet bilen.

d. Test klassen ved at tilføje nogle instanser af **AutoRepair** til et **Car** objekt og skriv dem ud til konsollen med Console.WriteLine i program.cs.

J. Tegn et Design-sekvensdiagram der viser, hvorledes der oprettes et nyt **Car** objekt, og der tilknyttes et nyt objekt af typen **AutoRepair** til **Car** objektet, som til sidst indsættes i **CarRegister.**

a. Sekvensdiagrammet skal tegnes med udgangspunkt i program.cs. Se koden nedenfor:

```
Car c3 = new Car("FT23455", 2020, "Ford", "Fuga", 25000);
AutoRepair repair4 = new AutoRepair("Motor skiftet", 27050);
c3.AddRepair(repair4);
carRegister.AddCar(c3);
```

K. Udvid implementationen af klassen **CarRegister**.

a. Tilføj metoden **AddRepairToCar**, der kan tilføje et **AutoRepair** objekt til et eksisterende **Car** objekt. Køretøjet skal identificeres ud fra dets registreringsnummer. Signaturen for metoden er:

public void AddRepairToCar(string regNr, AutoRepair newAutoRepair)

b. Test metoden **AddRepairToCar** ved at oprette en ny instans af **AutoRepair** og tilføje objektet til et allerede eksistende **Car** objekt i **CarRegister**. Efterfølgende kan **CarRegister** udskrives l konsollen med Console.WriteLine.

Iteration 3
I denne iteration skal der laves en metode, der kan udskrive en samlet pris på alle reparationer der har været på en given bil gennem tiden. Ligeledes skal der laves fejl håndtering på **Car** objekter.

L. Opdater klassen **Car** i klassediagrammet med en metode **TotalAutoRepairCost**. Metoden skal returnere den samlede indtjening der har været på bilen.
    a. Metoden skal have følgende signatur:

    public double TotalAutoRepairCost()

    b. Test metoden ved at kalde den i Program.cs.

M. Opdater klassen **Car**, således at der kastes en exception af typen *ArgumentException*, hvis der forsøges oprettet et **Car** objekt med et registreringsnummer der ikke er på 7 karakterer.

N. Vis hvorledes du kan teste det i Program.cs.

Iteration 4
I denne iteration skal systemet udvides med klasser til at registrere ejeren (**Owner**) af bilen og mekanikeren (**Mechanic)** der udfører reparationen.

Klassen **Owner**

Property	***Type***	***Beskrivelse***
Id	int	Unikt id der identificerer reparationen, eks. 101
Name	string	Persons navn
Phone	string	Telefonnummer
Title	string	Ejerens titel, eks. “hr” eller “fru”

Klassen **Mechanic**

Property	***Type***	***Beskrivelse***
Id	int	Unikt id der identificerer reparationen, eks. 101
Name	string	Persons navn
Phone	string	Telefonnummer
Educa-tion	string	Hvilken bil type, som mekanikeren er special uddannet i . Eks. “Toyota”.

O. Udvid og opdater design-klassediagrammet med klasserne **Owner** og **Mechanic**.

P. Overvej om det er muligt at anvende arv (hint det er tilladt at indføre en ny klasse).

Q. Implementer klassen **Owner** med de nødvendige instans felter, properties og konstruktør(er).

R. Implementer relationen mellem **Car** og **Owner** og vis hvorledes man kan udskrive et **Car** objektet med alle reparationer og oplysninger om bilens ejer.

S. Implementere relationen mellem **AutoRepair** og **Mechanic**

EKSTRA: *Kun hvis du blev hurtigt færdig og keder dig* 😊

Iteration 5
I denne iteration skal der implementeres en metode, der finder den samlede indtjening på alle reparationer på alle bilerne, der er udført gennem tiden.

T. Definer selv metoden og hvilken klasse den skal placeres i, dens returtype og parameterliste.

U. Opdater de nødvendige diagrammer.