H1

CASE

objektorienteret programmering

C#

Indledning:

Mål:

- at kunne arbejde med objektorienteret programmering (OOP) i C# herunder klasser og metoder, analysere og oversætte virkelighed (OOA), benytte de datastrukturer som er relevante for løsning af opgaven og at kunne arbejde sammen om en programmeringsopgave.
- at opnå et bredt kendskab til C#
- Udvide kendskabet til og bruge database normalisering, SQL, relationer, tabeller etc.
- at kunne lave færdig statisk klasse med databasekommunikation der kan bruges på de følgende moduler.
- at kunne arbejde med integration mellem database SQL og C#
- at kunne planlægge og følge op på sit arbejde. At være bevidst omkring forskellige løsninger og deres fordele og ulemper. At kunne diskutere på et fagligt niveau.
- at kunne formidle viden mundtligt og visuelt
- at afprøve versionsstyring og samarbejde om fælles løsninger

Forudsætninger:

Kendskab til SQL, design af database, tabeller og relationer, C# grundlæggende begreber og kodestrukturer og repræsentation (klasser, metoder, arv, indkapsling, lister, overload, override osv.)

Arbejdsform:

Gruppe med 2 max 3 medlemmer

Støtte og input undervejs:

læreren vil når det er nødvendigt lave relevante oplæg til hvor i processen I nu er kommet. Ud over det skal I bruge tidligere kode I har lavet, Google og hinanden.

Evaluering:

I skal fremlægge løsning enkeltvis over for læreren torsdag og fredag i anden uge af forløbet.

Opgaveoplæg/kravsspecifikation

Lærerene har valgt at I skal arbejde med et autoværksted, dets kunder og deres biler.

Der er følgende ønsker til systemet (kravsspecifikation):

1. Man skal kunne oprette en kunde, opdatere og slette en kunde
2. Man skal kunne få vist en kundeoversigt (herunder en sorteret form efter efternavn, eller sorteret biler)
3. Man skal kunne få vist en kundes biler.
4. Man skal kunne oprette, opdatere, slette og vise en bil (CRUD).
5. Hver bil skal have reg.nr., mærke, model, årgang, km, brændstoftype find selv på flere.
6. En kunde kan godt have flere biler.
7. En bil kan kun være ejet af én kunde
8. Både kunder og biler skal have en oprettelsesdato.
9. Man skal kunne oprette/rette/slette et værkstedsbesøg for en bil.
10. Det skal være muligt at vise en bils værkstedsophold/datoer.

TIP! I skal arbejde objektorienteret, dvs. en kunde er et objekt, en bil er et objekt osv. Som regel siger man at navneord er objekter og udsagnsord giver os

metoderne. (En menu kan også være et objekt, med metoder til at oprette nyt punkt etc.)

Al tilgang til databasen skal ske gennem en statisk klasse som kan håndtere databasekald som oprette, viser, (redigere) og slette poster (CRUD).

Det skal være muligt at udbygge dette system, så det til sidst minder om et fuldt funktionelt værksted, som I kan arbejde videre med og udbygge på efterfølgende skoleophold.

I koder i C# Console programmering. Selve databasen (etablering, tabeller, relationer, felter, navne definitioner) må gerne oprettes uden for C# miljøet i f.eks. SQL server management Studio. Her kan I også indlægge testdata.

Forslag til overordnet arbejdsplan:

- Plan over hvad I skal lave de enkelte dage.
- Design og etablering af databasen med testdata.
- Kodning og test af statisk klasse til databaseadministration.
- Analyse hvilke objekter og metoder skal benyttes, brug UML diagrammer hvor det giver mening. Hvilke klasser, hvilken beskyttelse (private osv.)
- Hvilke skærmbilleder får I brug for (brugergrænseflade), Wireframe?
- Hvilke datastrukturer skal benyttes (f.eks. lister, arrays osv.)
- Kodning i C#.
- Test og rettelser (kan systemet nemt udbygges med de foreliggende strukturer)
- Fremlæggelse (demo af system og forklaring af valgte løsninger). Brug diagrammer, C# kode, foto, video etc.

Følgende skal afleveres på ItsLearning onsdag i uge 2:

- Overordnet tidsplan
- Logbog over daglige aktiviteter og udfordringer
- UML diagrammer, f.eks. class diagram, E/R diagrammer
- Kildekode
- Test dokumentation for væsentlige krav

Værktøjer til UML diagrammer
www.draw.io
https://www.gliffy.com/uses/uml-software/