Færdighedsprøve

i Fagene

Software Construction

og

Software Design

Datamatiker 1. semester

14. juni 2023

Domaine beskrivelse

FDF k19 Vanløse har fået bygget en bålhytte i baghaven ved Vanløse Kirke, der bruges til at bage snobrod og andre ting over bål. Da bålhytten er meget efterspurgt af de enkelte børnegrupper i kredsen, ønsker lederne et system kaldet "**Bålhyttebooking**" der kan holde styr på, hvornår de enkelte børnegrupper har booket hytten. Som udgangspunkt kan man max. booke bålhytten 2 gange pr. sæson, men hvis den er ledig kan den bookes. Børnemøderne er om torsdagen, hvor hytten kan bookes på hele lige klokkeslet (12:00, 14:00, etc.)

I systemet skal der være en klasse **børnegruppe**, hvis instanser/objekter, hver især kan indeholde oplysninger en børnegruppe.

Klassen ***Boernegruppe*** har således følgende properties:

Property	**Datatype**	**Beskrivelse**
ID	string	Gruppens ID, som skal være unik
Navn	string	Gruppens navn
Aldersgruppe	string	Kan antage en af streng-værdierne: "pusling", "tumling" , "pilt", "væbner" eller "seniorvæbner"
AntalDeltagere	int	Antallet af personer i gruppen

I systemet skal der ligeledes være en klasse ***Reservation*****,** som skal have følgende properties:

Property	Datatype	Beskrivelse
ID	int	Reservationens ID, som skal være unik
Tidspunkt	DateTime	Det tidspunkt, hvor man ønsker at reservere hytten.
Boernegruppe	Boernegruppe	En reference til et Boernegruppe object

I systemet skal der også være en klasse ***Reservationer*****,** som skal have følgende properties:

Property	Datatype	Beskrivelse
ID	int	Unikt ID, som er året for sæsonen.

Note: Du behøver ikke at implementere sikring af, at de ovennævnte ID properties er unikke før det bliver krævet i de efterfølgende opgaver.

Opgave 1. Opret Klassen BoerneGruppe.

- Opret et nyt ASP.Net Core console Application projekt, **`Baalhyttebooking`**
- Implementer klassen **`BoerneGruppe`** med instansfelter, properties og passende konstruktører.
- Implementer **`ToString()`** metoden på klassen.
- Test klassen ved at oprette nogle instanser af den i **`Main`** metoden (i C# filen program.cs) og skriv dem ud til konsollen med **`Console.WriteLine`**

Hints:

- *Du når frem til starten af opgave 1 ved at følge Opgavevejledningen*
- *Gør klassen public*

Opgave 2. Opret Klassen Reservation.

- Implementer klassen **`Reservation`** med instansfelter, properties og passende konstruktører.
- Implementer **`ToString()`** metoden på klassen.
- Test klassen ved at oprette nogle instanser af den i **`Main`** metoden og skriv dem ud til konsollen med **`Console.WriteLine`**

Hints:

- *Gør klassen public*

Opgave 3. Opret klassen Reservationer

`Reservationer` klassen skal indeholde en dictionary (af typen **`Dictionary<int,Reservation>`**) af **`Reservation`**s objekter. **`ID`** property'en bruges som key. Denne dictionary skal instantieres i konstruktøren.

- Implementer klassen **`Reservationer`** med instansfelter, properties og passende konstruktører.

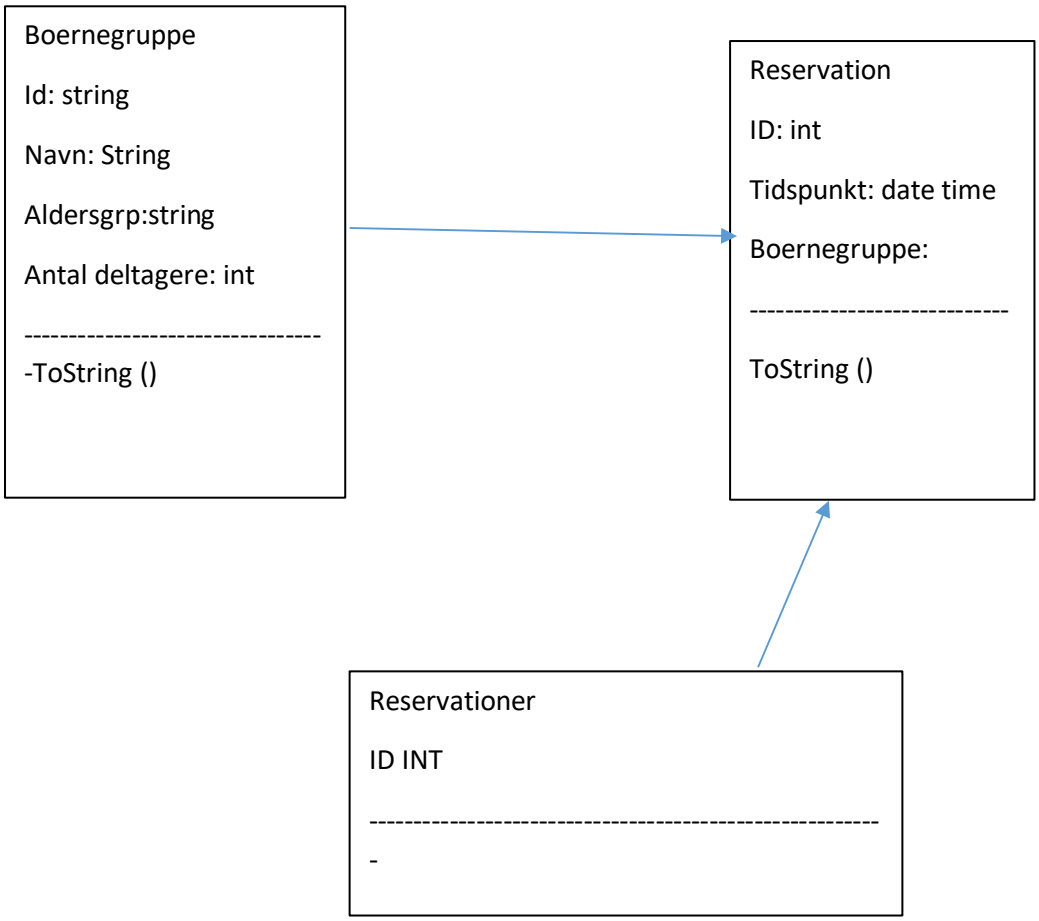
Hints:

- *Gør klassen public*

Opgave 4: Lav et sekvensdiagram

Tegn et sekvensdiagram for et kald af metoden **`RegistrerReservation`** på klassen **`Reservationer`**. Udgangspunktet for sekvensdiagrammet er **`Main`** metoden og sekvensdiagrammet skal kun vise et enkelt kald af **`RegistrerReservation`** og vise, hvorledes der oprettes 2 objekter hver af typerne **`Reservation`** og **`BoerneGruppe`** og at Reservations objektet tilføjes til **`Reservationer`**.

Opgave 5: Aggregering



På Klassediagrammet ovenover er klasserne sammensat ved hjælp af en Aggregering.

- Du bedes redegøre for om relationen er den rigtige. Hvilke andre typer af relationer kender du.

Opgave 6: Opret CRUD metoder på klassen Reservationer

Implementer følgende CRUD-funktioner:

- **`RegistrerReservation(Reservation reservation)`** metoden på klassen **`Reservationer`**. Metoden skal tilføje **`reservation`**en.
- **`PrintReservationer()`** metoden på klassen **`Reservationer`**, som skriver alle reservationer ud på skærmen på vha. en **`foreach`** løkke.

- **`void FjernReservation(Reservation reservation)`** metoden på klassen **`Reservationer`**. Metoden skal fjerne **`reservation`**en.
- Test metoderne ved oprette et **`Reservationer`** objekt i **`Main`** metoden og derefter kalde dem.

Opgave 7: Valideringsfunktioner

I **`Reservationer`** klassen implementeres følgene medlemsmetoder:

- **`public int AntalReservationer(BoerneGruppe bGruppe)`**
  Funktionen skal returnere det antal reservationer, der er foretaget for en børnegruppe, som parameteren **`bGruppe`** refererer til **`.`**

- **`public bool ReservationLedig(Reservation reservation)`**
  Funktionen skal retunere **`false,`** hvis reservationstidspunktet allerede er optaget (findes i dictionary). Ellers returneres værdien **`true.`** Parameteren **`reservation`** refererer til den pågældende reservation.

- **`ReservationsTidspunktOK(Reservation reservation)`**
  Funktionen skal retunere værdien **`false`** i følgende tilfælde:

Kriterie	Hint
Reservations år skal være lig med Reservationern ID	
Reservationsdagen skal være torsdag	Sammenlign reservationens ugedag med **`DayOfWeek.Thursday`**
Reservationens tidspunkt på dagen skal være 12:00, 14:00, 16:00, 18:00 eller 20:00	Du kan checke om reservations timetal findes i en liste som denne: **`new List<int>() {12,14,16,18,20}`**
Reservationens tidspunkt mht. minutter, sekunder og millesekunder skal alle være 0	

  Ellers returnerer funktionen værdien **`true.`** Parameteren **`reservation`** refererer til den pågældende reservation.

Opgave 8: Validerings funktioner (forsat)

Lav en samlet funktion i **`Reservationer`** klassen, der validerer en reservation:

- **`public bool ReservationOK(Reservation reservation)`**
  Skal returnere **`false`** i følgende tilfælde:

Kriterie	Hint
Reservationer skal have en reference til et **`Boernegruppe`** object	Check på **`Boernegruppe==null`**
Antal reservationer for en børnegruppe må ikke være højere end 2	Anvendt funktionen **`AntalReservationer`** til at udføre dette check.
Tidspunktet for reservationen må ikke være optaget.	Anvendt funktionen **`ReservationLedig`** til at udføre dette check.
Tidspunktet for reservationen skal være gyldigt.	Anvendt funktionen **`ReservationsTidspunktOK`** til at udføre dette check.

  Ellers skal funktionen returnere værdien **`true.`**

  Funktionen skal skive en passende meddelelse til brugeren ud på konsolen i hver enkelt tilfælde, hvor der returneres false.

- Test **`ReservationOK`** metoden i **`Main`** metoden**.**

Opgave 9: Exception handling

Indfør følgende i **`Reservationer`** klassen

- Ovennævnte metode **`ReservationOK`** skal i stedet at returnere **`false`** smide (**`throw`**) en exception af typn **`Exception`**. Genanvend din meddelelse til brugeren, når du istantierer din **`Exception.`**
- Test den ændrede udgave af **`ReservationOK`** metoden med **`try-catch`** i **`Main`** metoden**.**
- Integrer ovennævnte valideringsfunktioner i **`RegistrerReservation`** metoden

Opgave 10. Lav dine egne user stories

Lav et par user stories, der relaterer sig til **`Baalhyttebooking.`**

Du skal samtidig redegøre for INVEST kriterierne.

Opgave 11. Implementér dine egne user stories

Implementer de user stories, du definerede i opgaven ovenover.

Opgave 12. Indfør Enumeration type

Indfør en passende Enumeration type for **`Aldersgruppe`** property'en stedet for **`string`**.