Kursträff 7: Parprogrammeringsuppgifter

Ta hjälp av kursboken och av slidesen till Föreläsning 6 för att svara på frågorna nedan.

Kom ihåg att diskutera igenom frågorna så att ni är säkra på att ni förstår.

A. Diskussionsfrågor

1. Diskutera följande nyckelord och vad de används till
    a. `internal`
    b. `override`
    c. `interface`
    d. `base`
2. Vad är en klassmetod (static method), och hur skiljer den sig från en vanlig metod?
3. Vad är en generisk klass och hur skapar man en sådan?
4. Vad innebär polymorfism?
5. Vad är accessor-metoder och när används de?
6. Vad innebär det att implementera ett interface?

B. Programmeringsuppgifter

7. Skapa en ny klass, `Dog`
    a. Den ska ärva klassen `Animal` och implementerar gränssnittet `IEater` från koden nedan
        - `EatFood` metoden ska sätta `isHungry` till false
    b. Klassen ska också ha en sträng-egenskap som heter `Name`. Den ska vara public men med en protected set-accessor.
    c. Dog ska ha två överlagringar av konstruktorn:
        - `Dog()`
        - `Dog(string Name)`
    d. Båda konstruktoror-överlagringarna ska använda `base` för att sätta `isHungry` till `true`.

```
public class Animal {
      protected bool IsHungry { get; set; }

      public Animal(bool isHungry) {
            this.IsHungry = isHungry;
      }
}


interface IEater {
      void EatFood();
}
```

8. Skapa en ny klass, `DogPound`, som ärver från en `List` av `Dog`.
    a. Klassen ska ha en metod som heter `GetDogsByName`
        - Metoden ska ta emot en parameter, `string name`
        - Den returnerar en `List<Dog>` som bara innehåller de hundar som har namnet `name`.
    b. Den ska också ha en metod, `Feed`:
        - Metoden ska anropa `EatFood` på alla hundar.

C. Design patterns

Gå igenom slidesen om design patterns från föreläsning kursträff 6.

9. Genomför och diskutera hur följande ändringar förhåller sig till principerna:
    a. Skapa en factory method på `Dog` som heter `FromName` som tar emot en sträng, `name`. Metoden ska skapa och returnera en ny `Dog` med namnet `name`.
    b. Skapa ett nytt interface, `INameable`, som definierar egenskapen Name.
        - Implementera den i klassen `Dog`
    c. Gör om `DogPound` till en generisk klass `AnimalCollection<T>` som ärver från `List<T>` och som har constraintet `where T : IEater, INameable`
        - Diskutera constraintet. Vad innebär det?
        - Ändra `GetDogsByName` till `GetAnimalsByName`
            - Den nya metoden ska returnera en `List<INameable>`
10. Diskutera vilka fler ändringar man skulle kunna tänka sig att genomföra för göra koden tydligare eller lättare att ändra i framtiden
    a. Till exempel implementerar många samlingar i System.Collections.Generic gränssnittet `IEnumerable<T>`, och man brukar ofta i metoder returnera och ta emot parametrar som IEnumerable<T> istället för till exempel List<T>. T.ex. om man ska returnera en lista av strängar så returnerar man `IEnumerable<string>`, så är det upp till koden som använder metoden att bestämma vilken typ av samling den ska använda.
        - Hur skulle man kunna applicera det i `AnimalCollection<T>`?
    b. Borde `FromName` definieras på `INameable` och implementeras i `Dog` istället?
        - Vad borde i så fall `FromName` returnera för typ?