Exercise 1

Indledning:

Formålet med denne øvelse er at repetere nogle af de grundlæggende begreber i C++: Klasser og objekter, klasserelationen komposition samt tekststrenge (*strings*) i C++.

Opgaven består dels af spørgsmål, som du skal besvare, gerne i samarbejde med en medstuderende, dels af opgaver i UML og C++.

Gennem løsningen af opgaverne i denne øvelse vil du genopfriske ovenstående begreber og anvende dem i UML og C++.

Til øvelsen hører et sæt af source code filer (.cpp og .h-filer). Du skal bruge disse filer som udgangspunkt til at løse opgaven. Det gør du lettest ved at lave en MS Visual Studio Solution, f. eks. med navnet ”Ex1”, og så lave et C++ Console Project for hver delopgave, f. eks. med navnene ”Ex1.1”, ”Ex1.2”, etc.

Exercise 1.1: Klassen `Date`

Til denne øvelse skal du anvende filerne `Date.h` og `Date.cpp`.

a) Diskutér følgende med en medstuderende:
   1. Hvad gør de enkelte metoder?
   2. Hvorfor er nogle af metoderne erklæret for `const`, mens andre ikke er?
   3. Hvad kendetegner generelt `const`-metoder?

b) Tegn et UML klassediagram for klassen `Date`.

c) Test klassen `Date` med programmet i filen `test_Date.cpp`. Læg godt mærke til den systematiske måde at teste på – her er nøje overvejet hvilke *testinputs* der er nødvendige for at klassen er testen kvalitativt.

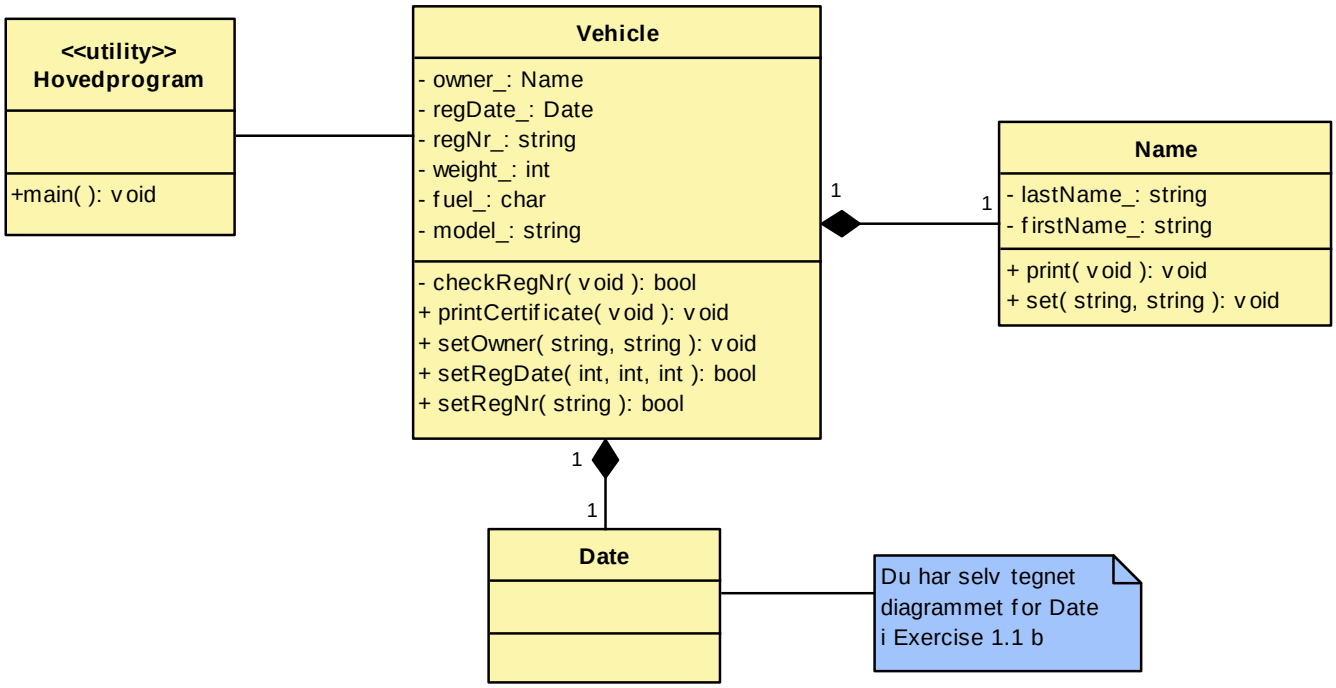
Exercise 1.2: Klassen `Name`

Til denne øvelse skal du anvende filerne `Name.h` og `Name.cpp`.

a) Skriv et C++ program hvori du tester klassen `Name`.
   Overveje nøje dine testinput og dit forventede testresultat, og sammenlign det med det faktiske resultat. Lad dig inspirere af test-filen til `Date` klassen.

Exercise 1.3: Klassen `Vehicle`

I forbindelse med udarbejdelsen af et program til Centralregisteret for Motorkøretøjer har man fundet det hensigtsmæssigt, at definere klassen `Vehicle`. Denne klasse gør brug af klasserne `Date` og `Name` som vist på UML klassediagrammet herunder



Til denne øvelse skal du anvende de ufærdige filer `Vehicle.h` og `Vehicle.cpp`.

a) Diskutér med en medstuderende hvad relationen mellem klasserne `Vehicle` og `Date` *hedder* og hvad den *betyder*?

b) Skriv filen `Vehicle.h` færdig. Tag udgangspunkt i UML klassediagrammet ovenfor, og husk at tage stilling til om nogle metoder skal erklæres `const`.

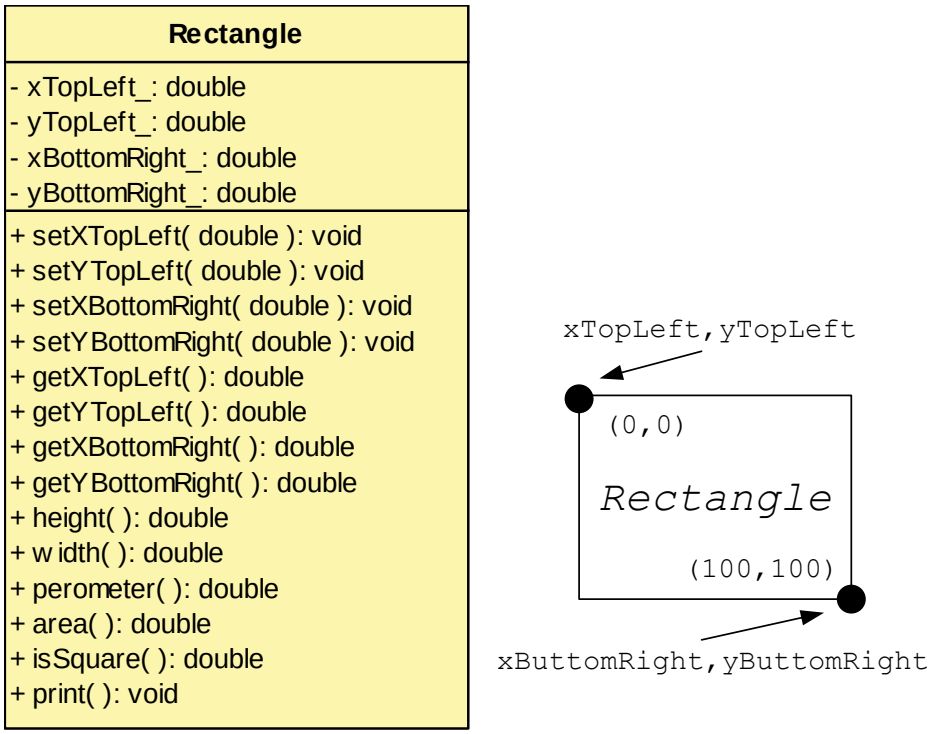
c) Skriv filen `Vehicle.cpp` så den implementerer klassen `Vehicle`.

d) Diskutér med en medstuderende hvorfor constructoren til klassen `Vehicle` *ikke* har behov for at initialisere medlemsvariablene `owner_` og `regDate_`? (Hvis I er i tvivl så prøv at oprette et objekt af klassen `Vehicle` og kald derefter metoden `printCertificate()` – hvordan fik `owner_` og `regDate_` deres værdier?)

e) Skriv et program der tester *hele* klassen `Vehicle`. Husk at overveje dine testinput og dit forventede testresultat, og sammenlign det med det faktiske resultat. Brug systematisk test som i test-filen til `Date` klassen.

Exercise 1.4: Klassen `Rectangle`

Herunder ser du UML klassediagrammet for klassen `Rectangle`, der modellerer et almindeligt rektangel. Rektanglet er beskrevet ved (x, y)-koordinaterne for det *øvre venstre hjørne* og det *nedre højre hjørne*:

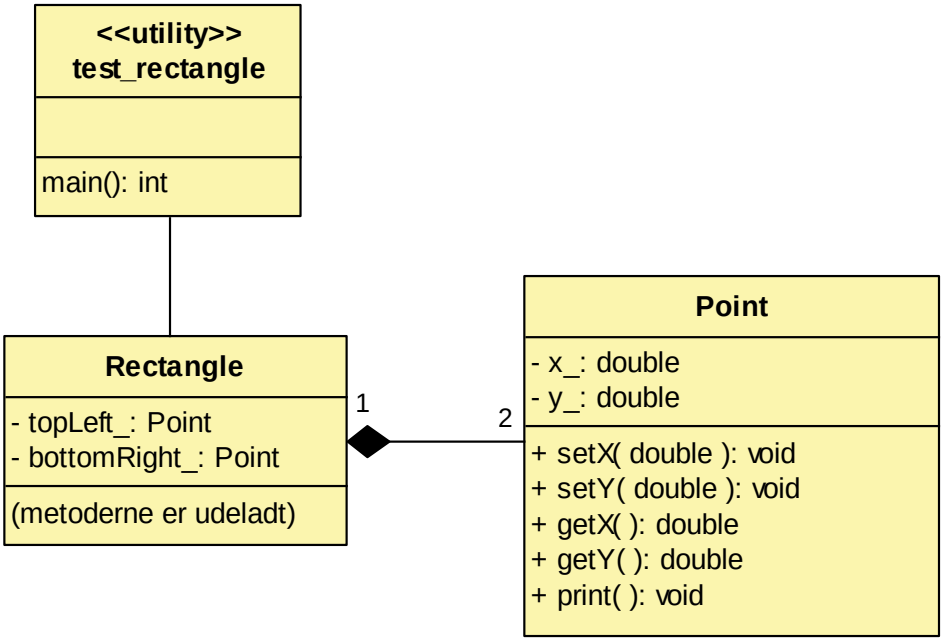


Klassen er implementeret i filerne `Rectangle.h/.cpp` som du finder sammen med denne øvelse.

a) Test klassen `Rectangle` med programmet i filen `test_Rectangle.cpp`.

Exercise 1.5: Klassen `Point`

På næste side ser du UML klassediagrammet for klassen `Point`. Et punkt er beskrevet ved koordinaterne x og y. Klassen er implementeret i filerne `Point.h/.cpp` som du finder du sammen med denne øvelse.
Du skal nu ***ændre*** klassen `Rectangle` så den anvender klassen `Point` til at bestemme hjørner, som illustreret i klassediagrammet (bemærk at metoderne i klassen `Rectangle` ikke er vist her).



Dvs.

- Koordinaterne `xTopLeft_`, og `yTopLeft_` skal erstattes med objektet `topLeft_` af typen `Point`.
- Koordinaterne `xBottomRight_` og `yBottomRight_` skal erstattes med objektet `buttomRight_` af typen `Point`.
- Efter at du har gjort det, er det selvfølgelig nødvendigt, at du laver ændringer i nogle af metoderne i klassen `Rectangle`.

b) Diskuter med en medstuderende, hvilke metoder i klassen `Rectangle` der skal ændres.

c) Indfør de nødvendige rettelserne i implementeringen af klassen `Rectangle`.

d) Test din nye udgave af klassen `Rectangle` – igen med filen `test_Rectangle.cpp`.