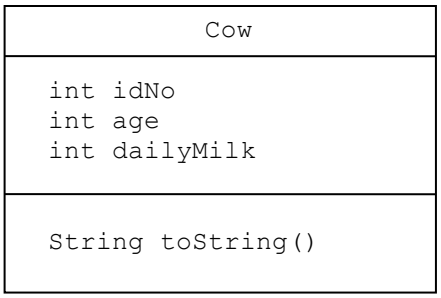
Cow



1. Opret en klasse, *Cow*, der repræsenterer en ko. Klassen er specificeret i UML-diagrammet til højre. De tre feltvariabler skal initialiseres i en konstruktør (via parametre af passende type). Metoden *toString* skal returnere en streng-repræsentation for en *Cow* på formen

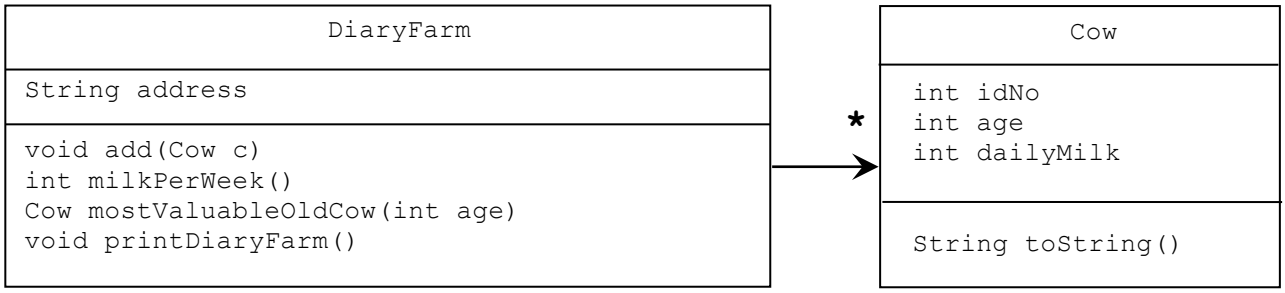
   ```
   "Cow 23419 is 6 years, 18 liter milk per day"
   ```

2. Lav en *Driver*-klasse med en *exam*-metode. Metoden skal være static, have returtype void og være uden parametre.

3. Opret fem velvalgte *Cow*-objekter i *exam*-metoden, via objektreferencer *c1,c2, c3, c4* og *c5,* og udskriv disse vha. *toString*-metoden.

Tilkald tilsynsførende og demonstrer det du har lavet indtil nu.

4. Opret en ny klasse, *DiaryFarm*, der repræsenterer en bondegård med køer. Klassen *DiaryFarm*, og dens relation til klassen *Cow*, er specificeret i følgende UML-diagram:



5. Programmér metoden *add,* der tilføjer *Cow*-objektet *c* til *DiaryFarm*-objektet.

6. Opret et objekt af typen *DiaryFarm* i *exam*-metoden i *Driver*-klassen og knyt de allerede oprettede *Cow*-objekter hertil.

7. Programmér metoden *milkPerWeek*. Metoden skal returnere køernes samlede mælkeproduktion pr. uge. Udvid *Cow*-klassen med de nødvendige get-metoder.

8. Afprøv metoden *milkPerWeek* i *exam*-metoden i *Driver*-klassen.

Tilkald tilsynsførende og demonstrer det du har lavet indtil nu.

9. Programmér metoden *mostValuableOldCow*. Metoden skal returnere den ko, der har størst mælkeproduktion blandt de køer, hvis alder er større end eller lig den angivne værdi. Hvis der ikke findes en sådan ko returneres null. Afprøv *mostValuableOldCow* i *exam*-metoden.

10. Programmér metoden *printDiaryFarm*. Metoden skal udskrive bondegårdens adresse efterfulgt af alle køer sorteret efter alder (højest til lavest). Hvis to har samme alder sorteres efter mælkeproduktion (højest til lavest). Afprøv *printDiaryFarm* i *exam*-metoden.

Tilkald tilsynsførende og demonstrer din færdige løsning.