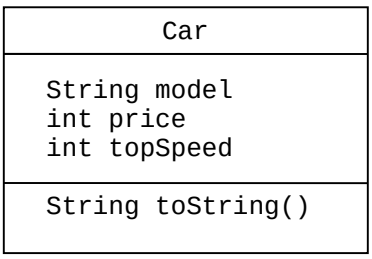
Car



1. Opret en klasse, *Car*, der repræsenterer en bil; klassen *Car* 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 *Car*, f.eks.

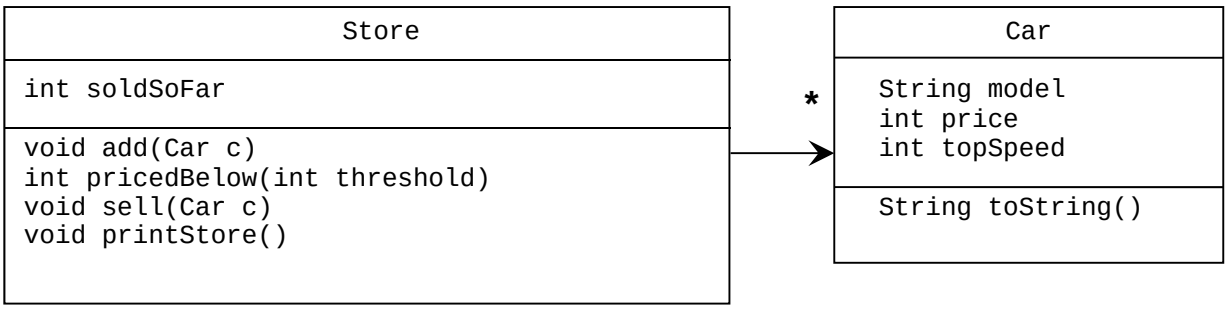
   ```
   "Ford Mustang, 455000, 250"
   ```

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

3. Opret fem velvalgte *Car*-objekter i *exam*-metoden, via objektreferencer *c1*, *c2*, *c3*, *c4* og *c*5, og udskriv disse vha. *toString*-metoden.

Tilkald tilsynsførende og demonstrer det du har lavet indtil nu.

4. Opret en ny klasse, *Store*, der repræsenterer en butik med et antal biler. Klassen *Store*, og dens relation til klassen *Car*, er specificeret i følgende UML-diagram:



5. Programmér metoden *add,* der tilføjer *Car*-objektet *c* til *Store*-objektet.

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

7. Programmér metoden *pricedBelow*. Metoden skal returnere antallet af biler, der har en pris under *threshold*. Udvid *Car*-klassen med de nødvendige get-metoder.

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

Tilkald tilsynsførende og demonstrer det du har lavet indtil nu.

9. Programmér metoden *printStore*. Metoden skal udskrive *soldSoFar* efterfulgt af alle biler i butikken sorteret efter pris (lavest til højest). Hvis to biler har samme pris, sorteres efter topfart (lavest til højest). Tilføj de nødvendige get-metoder. Afprøv *printStore* i *exam*-metoden.

10. Programmér metoden *sell*. Metoden skal fjerne den solgte bil fra *Store* og opdatere *soldSoFar* ved at addere den solgte bils pris. Afprøv *sell* i *exam*-metoden ved at kalde *printStore* før og efter salget.

Tilkald tilsynsførende og demonstrer din færdige løsning.