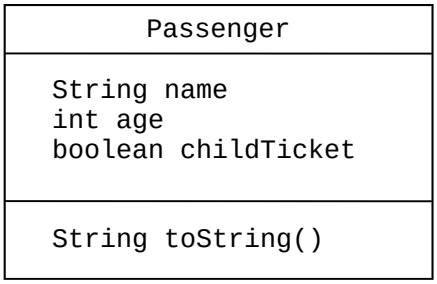
Passenger



1. Opret en klasse, *Passenger*, der repræsenterer en passager. 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 *Passenger* på formen

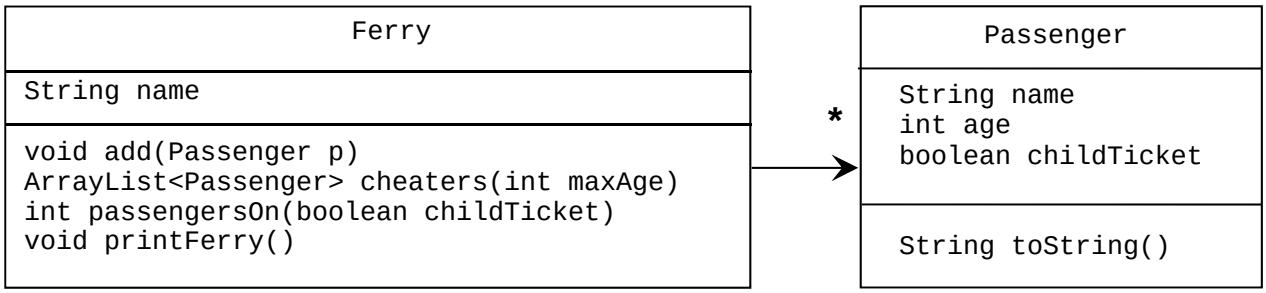
   ```
   ”Anna is 17 and travels as a child”
   ”David is 37 and travels as an adult”
   ```

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

3. Opret fem velvalgte *Passenger*-objekter i *exam*-metoden, via objektreferencer *p1, p2, p3, p4* og *p5,* og udskriv disse vha. *toString*-metoden.

Tilkald tilsynsførende og demonstrer det du har lavet indtil nu.

4. Opret en ny klasse, *Ferry*, der repræsenterer en færge med passagerer. Klassen *Ferry*, og dens relation til klassen *Passenger,* er specificeret i følgende UML-diagram:



5. Programmér metoden *add*, der tilføjer *Passenger*-objektet *p* til *Ferry*-objektet.

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

7. Programmér metoden *cheaters*. Metoden skal returnere alle passagerer, der er ældre end den angivne aldersgrænse (maxAge) og rejser på børnebillet. Udvid *Passenger*-klassen med de nødvendige get-metoder.

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

Tilkald tilsynsførende og demonstrer det du har lavet indtil nu.

9. Programmér metoden *passengersOn*. Metoden skal returnere antallet af passagerer på den type billet, som parameteren angiver. Afprøv *passengersOn* i *exam*-metoden.

10. Programmér metoden *printFerry*. Metoden skal returnere færgens navn efterfulgt af alle passagerer sorteret efter alder (lavest til højest). Hvis to har samme alder sorteres efter navn. Afprøv *printFerry* i *exam*-metoden.

Tilkald tilsynsførende og demonstrer din færdige løsning.