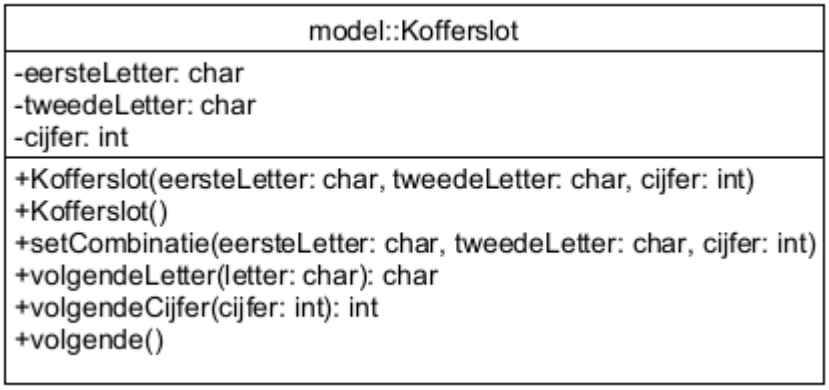
MAKE IT WORK

Opdracht Encapsulation-3 Kofferslot

Inleiding

De firma *Luggage & Co* rust haar reiskoffers uit met een kofferslot dat bestaat uit twee letters en één cijfer. Mogelijke combinaties zijn bijvoorbeeld BE7, KS2 en ML9.

Zie het volgende klassendiagram.



In deze opdracht ga je het Kofferslot simuleren. Daartoe maak je twee klassen: `Kofferslot` en `KofferslotLauncher`.
Opmerking vooraf: Je kunt met de all-args constructor een Kofferslot object aanmaken op basis van een letter-letter-cijfer-combinatie. Je kunt ook een bestaand object instellen op een nieuwe combinatie. Daar is de methode `setCombinatie(letter, letter, cijfer)` voor. Omdat de code van de constructor en de methode hetzelfde is, maakt de constructor gebruik van de methode. Je schrijft daardoor geen code twee keer. Zie de beschrijving van de opdracht hieronder.

Opdracht A: klasse Kofferslot

- Maak drie attributen:
  - Een private attribuut genaamd “eersteLetter”. Het type is `char`.
  - Een private attribuut genaamd “tweedeLetter”. Het type is `char`.
  - Een private attribuut genaamd “cijfer”. Het type hiervan is `int`.
- Maak twee constructors:
  - De default constructor `Kofferslot()`. Deze gebruikt “A” als defaultwaarde voor een letter en 0 als defaultwaarde voor een cijfer. Gebruik constructor chaining.
  - De constructor `Kofferslot(char eersteLetter, char tweedeLetter, int cijfer)`. Deze roept de methode `setCombinatie` aan. Zie het volgende onderdeel.
- Maak een methode `setCombinatie(char eersteLetter, char tweedeLetter, int cijfer)`. Deze methode geeft de klasse-attributen de waarden die zijn meegegeven in de parameters `eersteLetter`, `tweedeLetter` en `cijfer`. In feite combineert de methode de drie afzonderlijke setter-methode van de klasse-attributen. Je kunt daarom gebruik maken van de drie setter-methodes.
- Maak een methode `volgendeLetter(char letter)`. Deze methode geeft de volgende letter terug, de letter na de meegegeven letter. Dus volgendeLetter(‘A’) geeft ‘B’, volgendeLetter(‘B’) geeft ‘C’,… et cetera. Let wel: volgendeLetter(‘Z’) geeft weer ‘A’.

- Maak een methode `volgendeCijfer(int cijfer)`. Deze methode geeft het volgende cijfer terug. Deze werkt vergelijkbaar als de volgendeLetter() methode, maar dan met cijfers en na 9 komt dus 0.
- Maak een methode `volgende()`. Deze methode maakt gebruik van de methodes `volgendeLetter()` en `volgendeCijfer()` en verandert de attributen `eersteLetter`, `tweedeLetter` en `cijfer` op de onderstaande manier. De methode zet dus de combinatie van een Kofferslot object op de volgende combinatie.

AA0	AA1	AA2	AA3	AA4	…	AA8	AA9
AB0	AB1	AB2	…			AB8	AB9
…							
AZ0	AZ1	…					AZ9
BA0	BA1	…					BA9
…							
ZZ0	ZZ1	…					ZZ9
AA0							

Opdracht B: klasse KofferslotLauncher

In `KofferslotLauncher` ga je je code testen.

- Test of de default constructor van de klasse `Kofferslot` werkt.
- Test of the all-args constructor van de klasse `Kofferslot` werkt.
- Test of de methode `volgende()` van de klasse `Kofferslot` correct werkt.
  Test de volgende slotcombinaties:
  - AA0 volgende moet zijn: AA1
  - BR9 volgende moet zijn: BS0
  - DZ9 volgende moet zijn: EA0
  - ZZ9 volgende moet zijn: AA0

Richtlijnen bij coderen (zie ook HBO-ICT code conventions [ICC])

- Zorg dat je naam en het doel van het programma bovenin staan (ICC #1).
- Gebruik de juiste inspringing (*indentation*) bij de lay-out (ICC #2) .
- Let op juist gebruik hoofdletters en kleine letters (ICC #3).
- Gebruik goede namen (ICC #4).
- Vermijd *magic numbers* (ICC#5).
- Voeg waar nodig commentaar toe die inzicht geven in je code (ICC#7).
- Denk aan *encapsulation* (ICC #9).