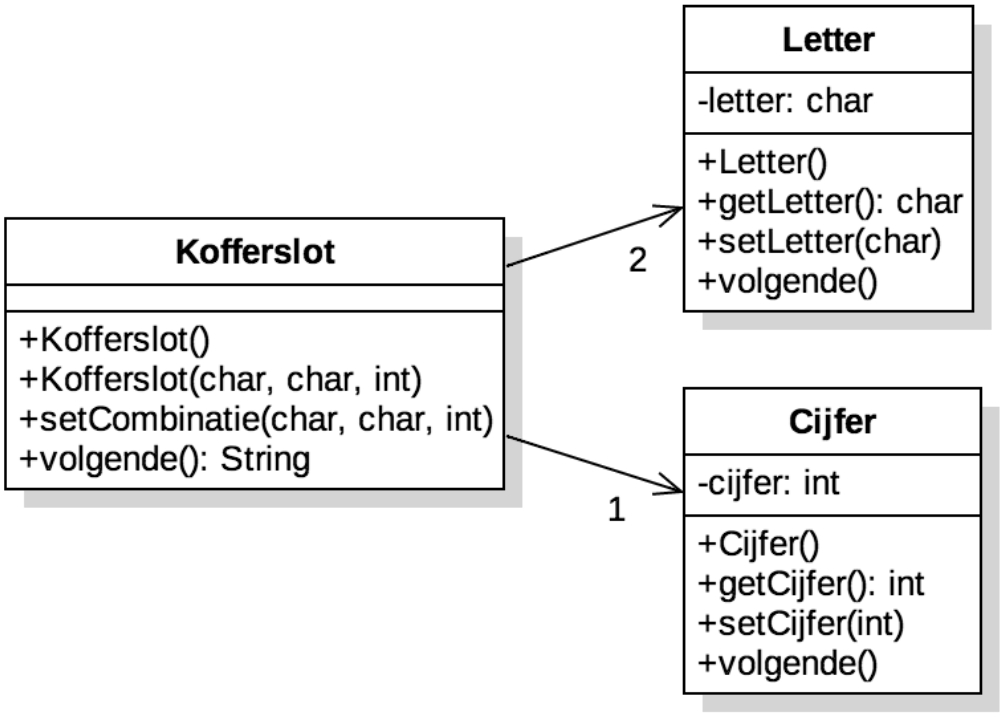
OOP1 – Oefenopdracht 3a

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 ga je achtereenvolgens vier klassen maken: `Letter`, `Cijfer`, `Kofferslot`, `Main`.

Opdracht A: klasse `Letter`

- Zoals te zien is in het klassendiagram heeft de class `Letter` een private attribuut genaamd “letter”. Het type daarvan is `char`. Maak deze.
- Maak de default constructor `Letter()`. Deze constructor initialiseert de letter op ‘A’.
- Maak een getter-methode `getLetter()`. Deze methode geeft de waarde van het attribuut “letter” uit de klasse terug.
- Maak een setter-methode `setLetter(char letter)`. Deze initialiseert het attribuut “letter” met de letter die als parameter wordt meegegeven.
- Maak een methode `volgende()`. Deze verandert het attribuut “letter”, door naar de volgende letter te gaan. Dus na ‘A’ komt ‘B’, na ‘B’ komt ‘C’,… et cetera. Na ‘Z’ komt weer ‘A’.

Opdracht B: klasse `Cijfer`

- Zoals te zien is in het klassendiagram heeft de class `Cijfer` een private attribuut genaamd “cijfer”. Het type daarvan is `int`. Maak deze.
- Maak de default constructor `Cijfer()`. Deze constructor initialiseert het cijfer op 0.
- Maak een getter-methode `getCijfer()`. Deze methode geeft de waarde van het attribuut “cijfer” uit de klasse terug.
- Maak een setter-methode `setCijfer(int cijfer)`. Deze initialiseert het attribuut “cijfer” met het cijfer die als parameter wordt meegegeven.
- Maak een methode `volgende()`. Deze verandert het attribuut “cijfer”, door naar het volgende cijfer te gaan. Dus na 0 komt 1, na 1 komt 2,… et cetera. Na 9 komt weer 0.

Opdracht C: klasse Kofferslot

- Maak twee attributen:
  - Een private attribuut genaamd “letterlijst”. Het type is een array van klasse `Letter` (de eerste klasse die je hierboven hebt gemaakt).
  - Een private attribuut genaamd “cijfer”. Het type hiervan is `Cijfer` (de tweede klasse die je hierboven hebt gemaakt).
- Maak twee constructors:
  - De default constructor `Kofferslot()`. Deze doet twee dingen:
    - Maak een letterlijst aan en zet daar twee objecten van type `Letter` in. Gebruik om de objecten aan te maken de default constructor van `Letter` die je hierboven hebt gemaakt.
    - Maak een cijfer aan. Gebruik hiervoor de default constructor van `Cijfer` die je hierboven hebt gemaakt.
  - De constructor `Kofferslot(char firstLetter, char secondletter, int cijfer)`. Deze doet twee dingen:
    - Maak een letterlijst aan en zet daar twee objecten van type `Letter` in. Vervolgens dienen deze te worden geïnitialiseerd met de waarden die zijn meegegeven in de parameters letter1 en letter2. Gebruik hiervoor de setter-methode van klasse `Letter` die je hierboven hebt gemaakt.
    - Maak een cijfer aan. Initialiseer deze met de waarde die is meegegeven in de parameter cijfer. Gebruik hiervoor de setter-methode van klasse `Cijfer` die je hierboven hebt gemaakt.
- Maak een methode `volgende()`. Deze methode verandert de attributen letterlijst en cijfer op de volgende manier.

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

  Gebruik hierbij de methode `volgende()` van de klasse `Letter` en de methode `volgende()` van de klasse `Cijfer`.

Opdracht D: klasse Main

- In `Main` ga je je code testen.
- 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 number*s (ICC#5).
- Gebruik javadoc tags: `@author`, `@param` en `@return` (ICC #6).
- Voeg waar nodig commentaar toe die inzicht geven in je code (ICC#7).
- Denk aan *encapsulation* (ICC #9).