Programmeren2 Practicum

Week 6a

HashMap

We gaan verder met het bespreken van een andere Java Library. Deze week gaat het over de HashMap. Dit iseen flink practicum, dus ga snel aan de slag!

Eerst even aandacht voor

In een practicum hebben we eerst een gezamenlijk deel. Even opletten wat de docent te vertellen heeft dus. We behandelen eerst:

- `HashMap`: Wat is het? Wat kan het?

SIMPEL

[1] ➔Bestudeer kort de beschrijving van de HashMap op de site van de Java 8 API. (http://docs.oracle.com/javase/8/docs/api/). Bestudeer met name de methods: `put`, en `get`.

Wanneer James Bond een van zijn geheime contactpersonen tegen komt gebruikt hij een lijst van codewoorden om te controleren of de ander wel echt is wie hij zegt te zijn, of dat James misschien met een vijandige spion te maken heeft.

Omdat James een dagje ouder wordt heeft Q voor hem een programma gemaakt (natuurlijk beschermd met allerlei slimme uitvindingen zodat alleen James Bond het kan gebruiken). Met dit programma kan James een codewoord invullen en dan krijgt hij automatisch het correcte antwoord te zien. Hij hoeft dan niet zelf alle codewoorden met hun antwoorden te onthouden.

We kunnen dit uitstekend implementeren met behulp van een HashMap. Immers, we willen een sleutel (het code woord dat James vraagt) koppelen aan een of andere waarde (het juiste antwoord).

[2] ➔Implementeer de CodeWordChecker zoals hiernaast staat.
Gebruik hiervoor de methods `put` en `get` van de class HashMap.

CodeWordChecker
`HashMap<String, String> words`
`void addWord(String codeWord, String respons)` `void showRespons(String codeWord)`

- Maak (dus) eerst een klasse CodeWordChecker volgens de klassenstructuur hiernaast / boven. En zorg dat de methodes werken.
  - Als ik met `addWord` een codewoord wil toevoegen dat al in de lijst staat dan moet de bestaande betekenis van het codewoord overschreven worden EN hiervan een waarschuwing in de console wordt afgedrukt.
  - De method `showRespons` () print het juiste antwoord op de console af. Als het codewoord niet in de lijst voorkomt dan moet ook een waarschuwing worden afgedrukt.

- Vergeet ook zeker niet een constructor toe te voegen waarin je de instance variabele words instantieert.

- Maak vervolgens een klasse MainClass met de `main()` method.
  - Gebruik in je main method onderstaande code om de CodeWordChecker te vullen.

```
CodeWordChecker cwc = new CodeWordChecker();

cwc.addWord("panda","potato");
cwc.addWord("pencil","earwax");
cwc.addWord("java","maniac");
cwc.addWord("student","superman");
cwc.addWord("rice","guitar");
cwc.addWord("humbug","bazinga");
// If all is well, the next statement should give a warning
cwc.addWord("java","bazinga");
```

Gebruik nu zelf een aantal keer de method `showRespons` om je programma te testen. Gebruik ook een keer een codewoord dat niet in je lijstje staat. Voorbeeld:

```
cwc.showRespons("java");
cwc.showRespons("pencil");
cwc.showRespons("Java"); // Let op hoofdletter!
```

[3] ➔ Bestudeer de beschrijving van de HashMap op de site van de Java 8 API. (http://docs.oracle.com/javase/8/docs/api/). Bestudeer met name de methods: `containsKey`, en `remove`.

[4] ➔ Maak de volgende wijzigingen / toevoegingen in bovenstaand project:

- Als je de method `containsKey` nog niet gebruikt had in je code, pas die dan nu toe in de methodes `addWord` en `showRespons` om te checken of het codewoord al bestaat in de hashmap.
- Voeg een method `removeCodePair(String codeWord)` toe die gebruik maakt van de `remove` functie van HashMap.

BASIS

Maak de BASIS opdrachten in de les als je al klaar bent met de SIMPEL opdracht, deze helemaal begrijpt en wel toe bent aan iets meer uitdaging.

[1] ➔ Bestudeer de beschrijving van de HashMap op de site van de Java 8 API. (http://docs.oracle.com/javase/8/docs/api/). Bestudeer met name de methods: `get`, `put`, `containsKey`, en `remove`. (Als je dit al niet gedaan hebt bij de simpel opdrachten.)

De politie wil een programma waarin ze bij kan houden in welke auto's overtredingen zijn begaan. Hiervoor koppelen ze kentekens aan overtredingen. Het kenteken wordt opgeslagen als `String` en daaraan gekoppeld is een beschrijving van de overtreding (ook een `String`).

[2] ➔ Maak een programma met de volgende onderdelen:

- Een MainClass met een (nu nog lege) `main`() method.

- Een TrafficTicketSystem klasse volgens onderstaand schema.
  - Lees ook de extra eisen & uitleg daar onder!

TrafficTicketSystem
`HashMap<String, String> carTickets`

`void addTicket(String numberPlate, String description)`
`void showTicket(String numberPlate)`
`void payTickets(String numberPlate)`
`void showAllTickets()`

addTicket : Voegt een nummerplaat met de beschrijving van de overtreding toe (aan carTickets).
Check eerst of de nummerplaat niet al aanwezig is! In dat geval moet je de beschrijving van deze overtreding achter de beschrijving plakken die er al staat!
Dit doe je door :

1. te checken of de nummerplaat al in de HashMap staat.
2. Zo ja: dan vraag je de beschrijving op die bij die nummerplaat hoort.
3. Daar plak je de nieuwe beschrijving achter. (*String concatinating*)
4. Dan voeg je de nummerplaat met beschrijving weer toe in de HashMap.
   (De HashMap class zal zelf de oude beschrijving overschrijven met de nieuwe.)

showTicket : Als de meegegeven nummerplaat een bekeuring aan zich gebonden heeft dan wordt de beschrijving in de console afgedrukt. Als dit niet zo is dan wordt een bericht gegeven dat deze auto geen bekeuring heeft.

payTickets : Alle bekeuringen op de meegegeven nummerplaat worden betaald. Verwijder de nummerplaat uit het systeem.

showAllTickets :Druk in de console alle kentekens met hun bekeuringen af.

[3] ➔ Breid de main() methode uit:

- Maak een TrafficTicketSystem object aan.
- Voeg een aantal nummerplaten met bekeuringen aan dit object toe.
- Test je code door minimaal alle methods van TrafficTicketSystem een keer te gebruiken.

De applicatie uit opgave 2 en 3 is eigenlijk maar onhandig. Ik wil namelijk niet alleen een beschrijving van een bekeuring toevoegen aan een kenteken, maar nog veel meer informatie. Namelijk wat voor soort auto bij dit kenteken hoort, wie de eigenaar is, etc. etc.
Dit is zonder problemen mogelijk. Zowel de key als de value van een HashMap mogen elk voorkomend type Java Class zijn.

[4] ➔ Schrijf een class Car (volgens het ontwerp hier naast). Hierin kan je alle relevante informatie kwijt. Wat nog beter is: we kunnen auto's zonder boetes gewoon in ons systeem laten staan.

Car

`String ownerName`
`String carType`
`String color`
`ArrayList<String> tickets`

`void printCarInfo()`
`boolean hasOpenTickets()`
`void payTickets()`

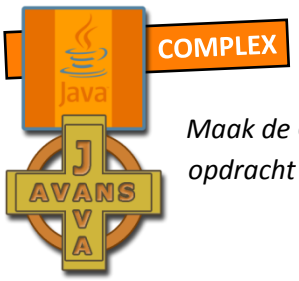
- Zorg ervoor dat de class Car een functie heeft waarmee we kunnen checken of er op die auto boetes openstaan.
- De method printCarInfo drukt, naast de standaard informatie uit de instance variables, ook informatie af over alle overtredingen als er met deze auto overtredingen zijn gepleegd.
- De constructor van `Car` heeft als parameters waarden voor de `ownerName`, `CarType` en `color`.
- Car moet nog meer methodes krijgen zoals alle getters en setters, addTicket(String ticket).

[5] ➔ Pas de class `TrafficTicketSystem` zo aan dat de instance variable `carTickets` nu een `HashMap <String, Car>` wordt. Pas alle methods zo aan dat ze weer werken met de nieuwe implementatie van de `HashMap`.

- De method `showTickets()` van het nieuwe `TrafficTicketSystem` drukt alle informatie af van de auto wanneer met deze auto een overtreding is begaan.
- Voeg ook een methode `addCar(String numberplate, Car car)` toe aan de klasse TrafficTicketSystem, zodat je de HashMap kan vullen.

[6] ➔ Pas ook eventueel je `main` methode aan en test weer of alles werkt.

- Voeg eerst een aantal auto's toe d.m.v. de nieuwe `addCar(String numberplate, Car car)` methode.
- Vervolgens kan je een aantal Tickets toevoegen op kenteken, waarbij je weer alles test (bestaande kentekens, niet-bestaande kentekens, meerdere tickets op dezelfde kentekens, enz.)



Maak de COMPLEX opdrachten in de les als je al klaar bent met de SIMPEL of BASIS opdracht en die te makkelijk vind of als je nog tijd over hebt.

FileLetterCounter
`<Your choice>`
`void countLetters(String fileName)` `void printResults()`

[1] ➔ Je kan maar op een paar manieren echt met HashMaps aan de gang. Veel moeilijker dan de basis-opdrachten wordt het niet. Vandaar dat deze complexe opdracht er iets extra's bij pakt, namelijk file-IO. Maak een class FileLetterCounter. Deze class kan een tekstbestand openen en een overzicht maken van hoeveel keer een bepaalde letter voorkomt in het bestand.

Er zitten hier best een paar valkuilen in. Bijvoorbeeld: je kan geen int of char in een HashMap zetten. In een HashMap moeten altijd objecten staan. (Zoek hiervoor eens op de term autoboxing.)