HO
GENT

H04 Array en ArrayList - Oefeningen

Table of Contents

1. Doelstellingen . . . 1
2. Oefeningen . . . 1
    2.1. Wat loopt fout? . . . 1
    2.2. Wat gebeurt er? . . . 2
    2.3. Gezinsuitgaven revised . . . 3
    2.4. Getallen hoger dan ondergrens . . . 4
    2.5. Hoogste en laagste omzet . . . 4
    2.6. Vier op een rij . . . 5
    2.7. Tankstation . . . 5
    2.8. Parking . . . 9
    2.9. Rekeningen 1 Dim . . . 12
    2.10. Wat loopt weer fout? . . . 14
    2.11. Gemiddelde getallen matrix . . . 14
    2.12. Rekeningen 2 Dim . . . 15
    2.13. Inkomsten en uitgaven . . . 16
    2.14. Unieke cijfers . . . 16
    2.15. Stoelendans . . . 17
    2.16. Tevredenheidsscores . . . 18
    2.17. Schermtijden . . . 19
    2.18. Rommelmarkt . . . 20
    2.19. Matrix transponeren . . . 24
    2.20. Puntenverwerking . . . 25

1. Doelstellingen

- Oefening 2.1: Declaratie en creatie van een eendimensionale array
- Oefening 2.2: Inzicht hebben in gebruik van eendimensionale array
- Oefening 2.3: Creatie, initialisatie en doorlopen van een eendimensionale array
- Oefening 2.4: Creatie, initialisatie en doorlopen van een eendimensionale array
- Oefening 2.5: Doorlopen van een vooraf door opsomming gedefinieerde eendimensionale array
- Oefening 2.6: Doorlopen van een eendimensionale array
- Oefening 2.7: Applicatie en domeinklasse ontwikkelen met eendimensionale array
- Oefening 2.8: Applicatie en domeinklasse ontwikkelen met eendimensionale array
- Oefening 2.9: Applicatie en domeinklasse ontwikkelen met eendimensionale array
- Oefening 2.10: Inzicht hebben in gebruik van tweedimensionale array
- Oefening 2.11: Doorlopen van een tweedimensionale array
- Oefening 2.12: Doorlopen van een tweedimensionale array van objecten
- Oefening 2.13: Creatie en gebruik van ArrayList
- Oefening 2.14: Creatie en gebruik van ArrayList
- Oefening 2.16: Gebruik van array en ArrayList
- Oefening 2.17: Gebruik van ArrayList
- Oefening 2.19: Werken met een tweedimensionale array
- Oefening 2.20: Werken met een ArrayList

2. Oefeningen

2.1. Wat loopt fout?

Onderstaande code snippets worden gebruikt om een eendimensionale array te declareren en te initialiseren. Jammer dat er enkele foutjes in de code zitten, zie jij wat fout is?

2.1.1. Code snippet 1

```
final int MAX = 3;
int table[MAX] = {1,2,3};
```

2.1.2. Code snippet 2

```
int max = 5;
int table1[] = new int[max];
table1[max] = 5;
```

```
max++;
double table2[] = new int[max];
```

2.2. Wat gebeurt er?

In onderstaande code snippets worden eenvoudige bewerkingen op eendimensionale arrays uitgevoerd. Kan jij voorspellen wat de inhoud is van de arrays na uitvoering van de stukjes code?

2.2.1. Code snippet 1

```
        int[] getallen = new int[10];
        getallen[9] = 90;
        getallen[0] = 100;
        getallen[1] = getallen[9 - 8];
        getallen[2] = getallen[0] * 2;
        getallen[3] = getallen[2] + getallen[0];
        getallen[4] = getallen[9] * 10;
        for (int i = 5; i <= 9; i++) {
            getallen[i] = i * 2;
        }

        // Array getallen:
            0   1   2   3   4   5   6   7   8   9
          -----------------------------------------
          |   |   |   |   |   |   |   |   |   |   |
          -----------------------------------------
```

2.2.2. Code snippet 2

```
        int[] getallen1 = new int[3];
        int[] getallen2 = getallen1;
        getallen1[1] = 100;
        getallen1 = new int[3];
        getallen2[1] = 500;

        // Array getallen1:
            0   1   2
          -------------
          |   |   |   |
          -------------

        // Array getallen2:
            0   1   2
          -------------
          |   |   |   |
          -------------
```

2.3. Gezinsuitgaven revised

Een programma laat toe dat een gebruiker voor elke maand van het jaar de gezinsuitgaven invoert (12 getallen). Na deze invoer wordt getoond voor welke maanden de gezinsuitgave, boven het gemiddelde waren.

Voorbeeld uitvoer

```
Geef de uitgaven in voor maand 1: 1255,14
Geef de uitgaven in voor maand 2: 1810,16
Geef de uitgaven in voor maand 3: 2000
Geef de uitgaven in voor maand 4: 1300
Geef de uitgaven in voor maand 5: 1500,99
Geef de uitgaven in voor maand 6: 1080
Geef de uitgaven in voor maand 7: 1310,10
Geef de uitgaven in voor maand 8: 1310,25
Geef de uitgaven in voor maand 9: 1500
Geef de uitgaven in voor maand 10: 1420
Geef de uitgaven in voor maand 11: 1240
Geef de uitgaven in voor maand 12: 1512
In de volgende maanden werd een omzet behaald groter dan het gemiddelde: 2 3 5 9 12
```

Voorbeeld uitvoer

```
Geef de uitgaven in voor maand 1: 1000
Geef de uitgaven in voor maand 2: 1000
Geef de uitgaven in voor maand 3: 1000
Geef de uitgaven in voor maand 4: 1000
Geef de uitgaven in voor maand 5: 1000
Geef de uitgaven in voor maand 6: 1000
Geef de uitgaven in voor maand 7: 1000
Geef de uitgaven in voor maand 8: 1000
Geef de uitgaven in voor maand 9: 1000
Geef de uitgaven in voor maand 10: 1000
Geef de uitgaven in voor maand 11: 1000
Geef de uitgaven in voor maand 12: 1000
Er zijn geen maanden met uitgaven hoger dan het gemiddelde!
```

In `Gezinsuitgaven.java` vind je de code die een student schreef vooraleer deze had gehoord van arrays. De invoer van de gezinsuitgave voor elke maand moet bijgehouden worden want pas na de bepaling van het gemiddelde (hiervoor heb je de 12 uitgaven nodig) kan er nagegaan worden in welke maanden de uitgave groter was dan dit gemiddelde (ook hier heb je de 12 uitgaven nodig). In de oplossing werd dan ook voor **elke maand een variabele** gedeclareerd.

Bekijk deze code! Je vindt de code in `H04start_Gezinsuitgaven`.

Begrijp dat deze aanpak niet zo opportuun is. Stel je maar eens voor dat we hetzelfde willen doen voor 52 weekuitgaven!

Herwerk dit programma gebruik makend van arrays.

Nadat je het programma herschreef kan je weer eens reflecteren over de wijzigingen die je in het herwerkte programma zou moeten aanbrengen om te werken met 52 weekuitgaven.

2.4. Getallen hoger dan ondergrens

Lees 10 gehele getallen in en plaats deze in een array. Lees vervolgens nog een getal `ondergrens` in. Druk alle getallen uit de array af, die strikt groter zijn dan `ondergrens`.

Voorbeeld uitvoer

```
Geef getal 1 op: 10
Geef getal 2 op: 50
Geef getal 3 op: 20
Geef getal 4 op: 0
Geef getal 5 op: -50
Geef getal 6 op: 5
Geef getal 7 op: 30
Geef getal 8 op: 2
Geef getal 9 op: 40
Geef getal 10 op: -5
Geef de ondergrens op: 8

Elementen van de array die strikt groter dan 8 zijn:
     Index    Waarde
         0        10
         1        50
         2        20
         6        30
         8        40
```

2.5. Hoogste en laagste omzet

Declareer en initialiseer in 1 stap een array met de omzet per maand van een bepaald warenhuis. De omzetwaarden zijn respectievelijk: 360, 2100, 450, 1450, 650, 780, 1200, 321, 560, 1850, 960, 420.

Druk vervolgens de omzet en de maand af voor de **maand met de laagste omzet**, en de **maand met de hoogste omzet**.

Het achtervoegsel voor maand 1 en 8 is `"ste"`, voor de andere maanden is dat `"de"`.

Voorbeeld uitvoer

```
Hoogste omzet van 2100 Euro werd behaald in 2de maand.
Laagste omzet van 321 Euro werd behaald in 8ste maand.
```

2.6. Vier op een rij

Vertrek van de code in `H04start_VierOpEenRij`

In deze applicatie wordt een `char[] karakters` op willekeurige manier gevuld met `'O'` en `'X'` karaktertjes. De bedoeling is dat je nagaat of de array `karakters` vier keer eenzelfde karakter op rij bevat. In de uitvoer toon je de inhoud van `karakters` en geef je een melding of er al dan niet vier op een rij gevonden werd.

Telkens je het programma uitvoert kan je een andere uitvoer verwachten, hier alvast enkele voorbeelden van uitvoer:

Voorbeeld uitvoer

```
De rij karakters: X O X X O X X O O O
Deze rij bevat geen vier op een rij...
```

Voorbeeld uitvoer

```
De rij karakters: X O X O X O X X X X
Deze rij bevat vier op een rij...
```

Voorbeeld uitvoer

```
De rij karakters: X O O O O O X O X X
Deze rij bevat vier op een rij...
```

2.7. Tankstation

Vertrek van de code in `H04start_Tankstation`.

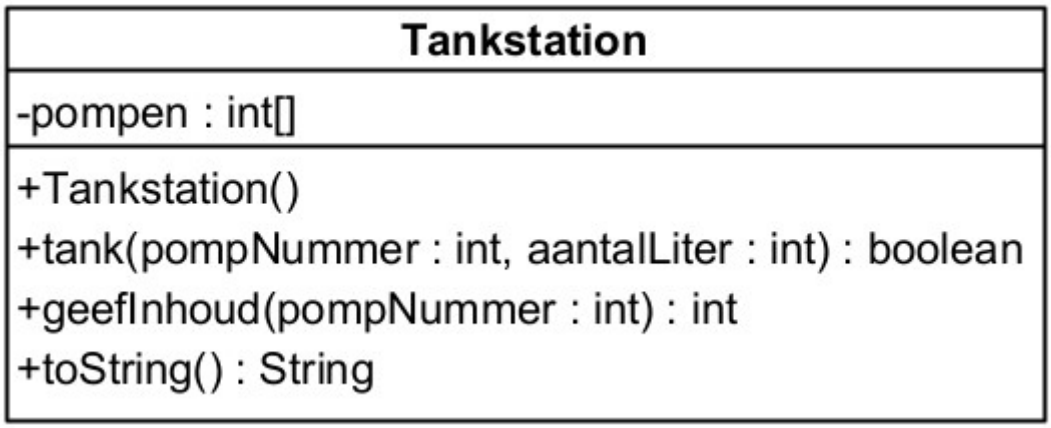
We gaan een applicatie ontwikkelen die toelaat dat een bediende van een tankstation tankbeurten ingeeft en het overzicht te zien krijgt van de brandstofvoorraad aan de 5 pompen (pomp 1 t.e.m. pomp 5) van het tankstation. De applicatie zal volgende interactie mogelijk maken:

Voorbeeld uitvoer

```
Aan welke pomp wil je tanken (1-5), druk 0 om te stoppen? 1
Hoeveel liter wens je te tanken (5 tot 80)? 60
Je hebt 60 liter getankt aan pomp 1

Overzicht pompen:
Pomp #1: 40 liter
Pomp #2: 100 liter
Pomp #3: 100 liter
Pomp #4: 100 liter
Pomp #5: 100 liter
```

```
Aan welke pomp wil je tanken (1-5), druk 0 om te stoppen? 3
Hoeveel liter wens je te tanken (5 tot 80)? 75
Je hebt 75 liter getankt aan pomp 3

Overzicht pompen:
Pomp #1: 40 liter
Pomp #2: 100 liter
Pomp #3: 25 liter
Pomp #4: 100 liter
Pomp #5: 100 liter

Aan welke pomp wil je tanken (1-5), druk 0 om te stoppen? 1
Hoeveel liter wens je te tanken (5 tot 80)? 50
Sorry, er is onvoldoende brandstof aan pomp 1

Overzicht pompen:
Pomp #1: 40 liter
Pomp #2: 100 liter
Pomp #3: 25 liter
Pomp #4: 100 liter
Pomp #5: 100 liter

Aan welke pomp wil je tanken (1-5), druk 0 om te stoppen? 0
Er werd in totaal 135 liter getankt in 2 tankbeurten.
```

2.7.1. Stap 1: implementatie domeinklasse `Tankstation`



Vervolledig de implementatie van de klasse `Tankstation`.

Hou rekening met onderstaande richtlijnen en test je klasse met de **unit testen** die je in het bestand `TankstationTest` vindt.

constructor `Tankstation()`

- er zijn `5` **pompen** in het tankstation
- elke pomp heeft `100` **liter** brandstofvoorraad

methode `boolean tank(int pompNummer, int aantalLiter)`

- voor een geldige tankbeurt moet
  - het `pompNummer` in het interval `[1, 5]` liggen
  - het `aantalLiter` moet in het interval `[5, 80]` liggen en mag **niet groter zijn dan de brandstofvoorraad** van de gewenste pomp
- bij een geldige tankbeurt wordt de brandstofvoorraad van de pomp verminderd met het gewenste aantal liter en retourneert de methode `true`
- bij een ongeldige tankbeurt blijft de brandstofvoorraad van de pomp ongewijzigd en retourneert de methode `false`

methode `int geefInhoud(int pompnummer)`

- retourneert de inhoud van de pomp indien een geldig pompnummer opgegeven is
- retourneert -1 voor een ongeldig pompnummer

methode `String toString()`

- maakt en retourneert een `String` die het overzicht van de **brandstofvoorraad van elke pomp** bevat; deze **methode is gegeven** en hoef je niet aan te passen. Je kan deze methode gebruiken om het overzicht van het tankstation te tonen in de `TankstationApp`

2.7.2. Stap 2: de applicatie `TankstationApp`

Vervolledig de klasse `TankstationApp` in de package `cui`. De gebruiker kan de gegevens voor opeenvolgende tankbeurten opgeven. Voor elke tankbeurt wordt het pompnummer en het gewenst aantal liter opgegeven.

Zorg voor

- **validatie** van het pompnummer in de methode `geefPompNummer`
  - in **voorbeeld uitvoer 01** wordt dit duidelijk geïllustreerd
- **validatie** van het gewenste aantal liter in de methode `geefAantalLiter`
  - in **voorbeeld uitvoer 02** wordt dit duidelijk geïllustreerd

Na elke tankbeurt krijgt de gebruiker te zien of de tankbeurt al dan niet succesvol was alsook het overzicht van het tankstation (maak hiervoor gebruik van de methode `toString` in de klasse `Tankstation`).

De gebruiker kan stoppen met het ingeven van tankbeurten door pompnummer `0` (dit is de **sentinel!**) in te geven. Op dat moment wordt het totaal aantal getankte liters getoond. Wanneer geen enkele tankbeurt plaatsvond wordt het overzicht van het tankstation (met volle pompen) getoond.

Voorbeeld uitvoer 01

```
Aan welke pomp wil je tanken (1-5), druk 0 om te stoppen? 6
```

```
We hebben geen pomp met nummer 6! Probeer opnieuw...
Aan welke pomp wil je tanken (1-5), druk 0 om te stoppen? -5
We hebben geen pomp met nummer -5! Probeer opnieuw...
Aan welke pomp wil je tanken (1-5), druk 0 om te stoppen? 3
Hoeveel liter wens je te tanken (5 tot 80)? 25
Je hebt 25 liter getankt aan pomp 3

Overzicht pompen:
Pomp #1: 100 liter
Pomp #2: 100 liter
Pomp #3: 75 liter
Pomp #4: 100 liter
Pomp #5: 100 liter

Aan welke pomp wil je tanken (1-5), druk 0 om te stoppen? 0
Er werd in totaal 25 liter getankt in 1 tankbeurt.
```

Voorbeeld uitvoer 02

```
Aan welke pomp wil je tanken (1-5), druk 0 om te stoppen? 1
Hoeveel liter wens je te tanken (5 tot 80)? 4
Je moet minstens 5 en kan maximaal 80 liter tanken...
Hoeveel liter wens je te tanken (5 tot 80)? 90
Je moet minstens 5 en kan maximaal 80 liter tanken...
Hoeveel liter wens je te tanken (5 tot 80)? 80
Je hebt 80 liter getankt aan pomp 1

Overzicht pompen:
Pomp #1: 20 liter
Pomp #2: 100 liter
Pomp #3: 100 liter
Pomp #4: 100 liter
Pomp #5: 100 liter

Aan welke pomp wil je tanken (1-5), druk 0 om te stoppen? 1
Hoeveel liter wens je te tanken (5 tot 80)? 50
Sorry, er is onvoldoende brandstof aan pomp 1

Overzicht pompen:
Pomp #1: 20 liter
Pomp #2: 100 liter
Pomp #3: 100 liter
Pomp #4: 100 liter
Pomp #5: 100 liter

Aan welke pomp wil je tanken (1-5), druk 0 om te stoppen? 3
Hoeveel liter wens je te tanken (5 tot 80)? 70
Je hebt 70 liter getankt aan pomp 3

Overzicht pompen:
```

```
Pomp #1: 20 liter
Pomp #2: 100 liter
Pomp #3: 30 liter
Pomp #4: 100 liter
Pomp #5: 100 liter

Aan welke pomp wil je tanken (1-5), druk 0 om te stoppen? 0
Er werd in totaal 150 liter getankt in 2 tankbeurten.
```

Voorbeeld uitvoer 03

```
Aan welke pomp wil je tanken (1-5), druk 0 om te stoppen? 0
Er waren geen tankbeurten:
Pomp #1: 100 liter
Pomp #2: 100 liter
Pomp #3: 100 liter
Pomp #4: 100 liter
Pomp #5: 100 liter
```

2.8. Parking

Vertrek van de code in `H04start_Parking`.

In deze applicatie gaan we simuleren hoe auto's parkeren op een parking met een beperkt aantal plaatsen. De parking is initieel leeg. Telkens een auto wil parkeren wordt de nummerplaat ingelezen. Enkel als er vrije plaatsen zijn in de parking gaat de slagboom open en kan de auto **parkeren**. Er kunnen parkeerplaatsen vrijkomen wanneer auto's de parking **verlaten**... De parkeerplaatsen zijn vanaf 0 oplopend genummerd.

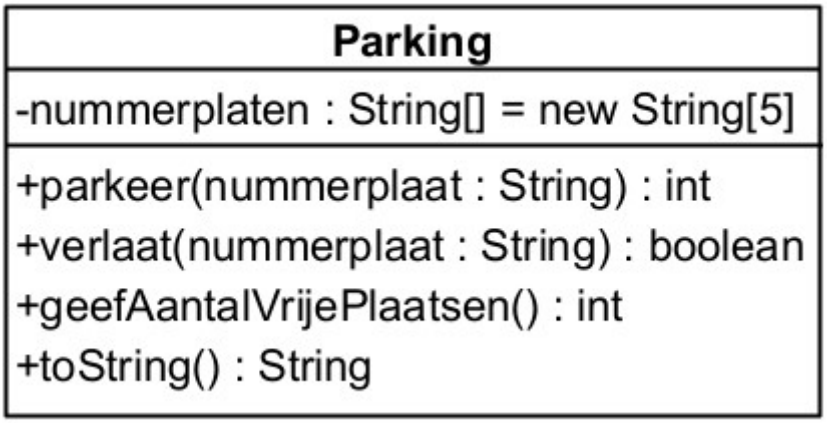
Hieronder zie je alvast hoe de interactie met de applicatie kan verlopen:

Voorbeeld uitvoer

```
1. Parkeer
2. Verlaat parking
3. Stop
Geef je keuze op: 1
Geef de nummerplaat in > RALLY
De auto staat geparkeerd op plaats 0

                 === PARKING (4 vrije plaatsen) ===
          0           1           2           3           4
      RALLY    --vrij--    --vrij--    --vrij--    --vrij--

1. Parkeer
2. Verlaat parking
3. Stop
Geef je keuze op: 1
Geef de nummerplaat in > 1-ABC-123
```

```
De auto staat geparkeerd op plaats 1

                  === PARKING (3 vrije plaatsen) ===
           0           1           2           3           4
       RALLY   1-ABC-123    --vrij--    --vrij--    --vrij--

1. Parkeer
2. Verlaat parking
3. Stop
Geef je keuze op: 1
Geef de nummerplaat in > MAMA
De auto staat geparkeerd op plaats 2

                  === PARKING (2 vrije plaatsen) ===
           0           1           2           3           4
       RALLY   1-ABC-123        MAMA    --vrij--    --vrij--

1. Parkeer
2. Verlaat parking
3. Stop
Geef je keuze op: 2
Geef de nummerplaat in > 1-ABC-123
De auto verliet de parking...

                  === PARKING (3 vrije plaatsen) ===
           0           1           2           3           4
       RALLY    --vrij--        MAMA    --vrij--    --vrij--

1. Parkeer
2. Verlaat parking
3. Stop
Geef je keuze op: 3
```

2.8.1. Stap 1: implementatie domeinklasse `Parking`



Vervolledig de implementatie van de klasse `Parking`.

De array `nummerplaten` wordt automatisch geïnitialiseerd bij declaratie: `private String nummerplaten`

`= new String[5]`. De array zal dus opgevuld zijn met `null`-s. Hou er rekening mee dat je op deze manier de vrije plaatsen herkent: een element dat `null` is duidt immers op een vrije plaats...

Hou rekening met onderstaande richtlijnen en test je klasse met de **unit testen** die je in het bestand `ParkingTest` vindt.

methode `int parkeer(String nummerplaat)`

- indien de nummerplaat niet geldig is retourneert de methode `-1`
  - een geldige nummerplaat is niet ´null´ en bevat 3 tot 5 karakters die niet allemaal spaties mogen zijn
- voor een geldige nummerplaat
  - als er plaats vrij is wordt de nummerplaat toegekend aan een vrije plaats en wordt de **index** van deze plaats geretourneerd
  - als er geen vrije plaatsen zijn retourneert de methode `-1`

methode `boolean verlaat(String nummerplaat)`

- retourneert `false` als er geen enkel element in de array `nummerplaten` gelijk is aan de gegeven `nummerplaat`, maak gebruik van de `equals`-methode om `String`-s te vergelijken!
- retourneert `true` als er wel een element gevonden wordt dat gelijk is aan de gegeven `nummerplaat`; dit bewuste element wordt gelijkgesteld aan `null`.

methode `String toString()`

- maakt en retourneert een `String` die het **overzicht van de parking** bevat; deze **methode is gegeven** en hoef je niet aan te passen.

2.8.2. Stap 2: de applicatie `ParkingApp`

Vervolledig de klasse `ParkingApp` in de package `cui`. De gebruiker kan via een menu een keuze maken.

methode `void parkeer(Parking parking)`

- vraag een nummerplaat op door de methode `geefNummerplaat` aan te roepen
- parkeer de auto; hiertoe roep je de methode `parkeer` aan (de gebruikte parking is de parameter `parking`).
- geef in de console weer of het parkeren al dan niet succesvol was

Voorbeeld uitvoer

```
// voorbeeld: de parking is niet vol...

Geef de nummerplaat in > RALLY
De auto staat geparkeerd op plaats 0
```

Voorbeeld uitvoer

```
// voorbeeld: de parking is vol...

Geef de nummerplaat in > PAPA
Sorry, de parking is vol, auto niet geparkeerd
```

methode `void verlaat(Parking parking)`

analoog aan parkeer methode

Voorbeeld uitvoer

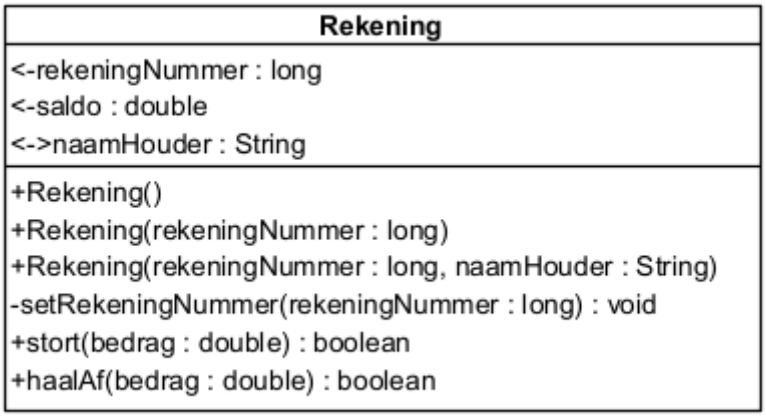
```
// voorbeeld: de auto was niet op de parking geparkeerd...

Geef de nummerplaat in > AZERTY
Deze auto stond niet op de parking
```

Voorbeeld uitvoer

```
// voorbeeld: de auto was op de parking geparkeerd...

Geef de nummerplaat in > RALLY
De auto verliet de parking...
```

2.9. Rekeningen 1 Dim

2.9.1. Stap 1: implementatie domeinklasse Rekening



Implementeer de klasse Rekening die je in oefening 4 van hoofdstuk 3 ontworpen hebt.

constructors

- het saldo wordt ingesteld op 0

- als geen rekeningnummer wordt opgegeven, dan wordt het rekeningnummer ingesteld op `123456789`
- als geen houder wordt opgegeven, dan wordt de houder ingesteld op `"onbekend"`

methode `boolean stort(double bedrag)`

- enkel een strikt positief bedrag kan gestort worden; in dit geval retourneert de methode `true` en wordt het `saldo` vermeerderd met het opgegeven bedrag
- voor een negatief bedrag retourneert de methode `false` en wordt het `saldo` niet aangepast

methode `boolean haalAf(double bedrag)`

- enkel een strikt positief bedrag dat niet groter is dan het saldo van de rekening kan afgehaald worden; in dit geval retourneert de methode true en wordt het saldo aangepast
- voor een negatief bedrag of een bedrag groter dan het saldo retourneert de methode `false` en wordt het `saldo` niet aangepast

Voeg in een package `testen` het gegeven bestand **`RekeningTest.java`** toe en gebruik de gegeven **unit testen** om je klasse op correctheid te testen. Je vindt dit bestand in `H04start_Rekening`.

2.9.2. Stap 2: de applicatie RekeningApplicatie

Maak een eendimensionale array van `Rekening`-objecten met volgende saldo's:

`100, 777.77, 287.15, -350, 399.99, 123.45, 987.65, -68.18, 413.26`.

De rekeningnummers zijn opeenvolgende getallen beginnend bij `0` en de houders bestaan telkens uit 1 letter, ook opeenvolgend beginnend bij `'A'`.

Declareer en initialiseer een array van `double`-s voor de gegeven saldo's en gebruik deze om op de `Rekening` objecten te maken

Doorloop na het aanmaken van de rekeningen de array met een **enhanced for** om op die manier alle gegevens van de rekeningen in een uitvoerstring te zetten en het gemiddelde saldo te berekenen.

Deze gegevens (de uitvoerstring en het gemiddelde) worden tenslotte getoond.

Voorbeeld uitvoer

```
Rekening 0 van A bevat 100,00 Euro
Rekening 1 van B bevat 777,77 Euro
Rekening 2 van C bevat 287,15 Euro
Rekening 3 van D bevat 0,00 Euro
Rekening 4 van E bevat 399,99 Euro
Rekening 5 van F bevat 123,45 Euro
Rekening 6 van G bevat 987,65 Euro
Rekening 7 van H bevat 0,00 Euro
```

```
Rekening 8 van I bevat 413,26 Euro

Het gemiddelde saldo van deze 9 rekeningen bedraagt 343,25 Euro
```

2.10. Wat loopt weer fout?

Een student wil met volgend stukje code een tweedimensionale array declareren en initialiseren. Elke element zou de som van de rij- en kolomindex moeten bevatten. Jammer dat er een foutje in de code sloop, zie jij waar dit gebeurde?

```
        final int MAX = 10;
        int[][] table = new int[MAX][MAX];
        for (int i = 0; i <= table.length; i++)
            for (int j = 0; j <= table[j].length; j++)
                table[i, j] = i + j;
```

2.11. Gemiddelde getallen matrix

In deze applicatie wordt de gebruiker gevraagd getallen in te voeren voor een 3 x 2 matrix. De applicatie toont vervolgens het gemiddelde van de elementen in de matrix.

Je kan vertrekken van de start code die verderop staat maar hieronder zie je alvast voorbeeld uitvoer:

Voorbeeld uitvoer

```
Geef getal op rij 1 en kolom 1: 10
Geef getal op rij 1 en kolom 2: 20,5
Geef getal op rij 2 en kolom 1: 25,47
Geef getal op rij 2 en kolom 2: 10
Geef getal op rij 3 en kolom 1: 10
Geef getal op rij 3 en kolom 2: 12,5

Het gemiddelde van alle getallen in de matrix is 14,7
```

Startcode: geef een invulling aan beide TODO's.

```
public class Startup {

    public static void main(String arg[]) {
        new Startup().berekenGemiddeldeVanGetallenInTweedimensionaleArray();
    }

    private void berekenGemiddeldeVanGetallenInTweedimensionaleArray() {
        double getallenMatrix[][] = new double[3][2]; // 3 rijen, 2 kolommen

        // TODO 1 - Maak gebruik van de onderstaande methode geefElement
```

```
        // om een waarde op te vragen en toe te kennen aan elk element van
        // getallenMatrix.


        // TODO 2 - Maak gebruik van een enhanced for lus om alle elementen van
        // getallenMatrix te overlopen. Maak de som van alle getallen en
        // tel hoeveel getallen er in getallenMatrix zitten zodat je het
        // gemiddelde kan berekenen en uitvoeren naar de console.

    }

    private double geefElement(int rij, int kolom) {
        Scanner invoer = new Scanner(System.in);
        System.out.printf("Geef getal op rij %d en kolom %d: ", rij, kolom);
        return invoer.nextDouble();
    }
```

2.12. Rekeningen 2 Dim

Schrijf een applicatie die een tweedimensionale array van `Rekening`-objecten maakt en opvult. Je kan de klasse `Rekening` uit een vorige oefening kopiëren.

Declareer de twee dimensionale array `rekeningen` van type `Rekening[][]`. Zorg dat `rekeningen` 3 rijen heeft.

Vraag aan de gebruiker uit hoeveel kolommen elke rij bestaat. Het aantal kolommen moet strikt positief zijn. Initialiseer elk element op een `Rekening`-object gebruik makend van de default constructor `Rekening()` en stort meteen een bedrag op de rekening. Kijk goed naar de voorbeeld uitvoer; je zal zien dat het bedrag bestaat uit een combinatie van de rij- en kolom index van de rekening.

Na het ingeven van het aantal kolommen voor elke rij wordt de inhoud van `rekeningen` rij poer rij getoond zoals je op de voorbeeld uitvoer kan zien.

Voorbeeld uitvoer

```
Geef aantal kolommen in voor rij 1: 3
Geef aantal kolommen in voor rij 2: 5
Geef aantal kolommen in voor rij 3: 2
11      12      13
21      22      23      24      25
31      32
```

Voorbeeld uitvoer

```
Geef aantal kolommen in voor rij 1: 5
Geef aantal kolommen in voor rij 2: 0
Rij 2 moet minstens 1 kolom hebben!
Geef aantal kolommen in voor rij 2: -8
```

```
Rij 2 moet minstens 1 kolom hebben!
Geef aantal kolommen in voor rij 2: 3
Geef aantal kolommen in voor rij 3: 4
11      12      13      14      15
21      22      23
31      32      33      34
```

2.13. Inkomsten en uitgaven

Een student wil zijn financiële toestand onder controle houden en bouwt hiervoor een kleine applicatie. De applicatie leest bedragen (komma getallen) in die zowel **positief (inkomsten)** als **negatief (uitgaven)** kunnen zijn. Positieve of negatieve bedragen van minder dan 1 Euro worden genegeerd en worden in de applicatie niet geregistreerd. De gebruiker signaleert het einde van de invoer met het ingeven van `0`.

De applicatie toont dan een overzicht van alle inkomsten en uitgaven, en toont eveneens het totaal van inkomsten en uitgaven.

Voorbeeld uitvoer

```
Geef bedrag op (0 om te stoppen) > 50
Geef bedrag op (0 om te stoppen) > 30
Geef bedrag op (0 om te stoppen) > -100
Geef bedrag op (0 om te stoppen) > 60
Geef bedrag op (0 om te stoppen) > -0,85
Geef bedrag op (0 om te stoppen) > 0,99
Geef bedrag op (0 om te stoppen) > 0

Inkomsten (totaal = 140,00 Euro):
50,00  30,00  60,00

Uitgaven (totaal = 100,00 Euro):
100,00
```

2.14. Unieke cijfers

De gebruiker geeft een geheel getal in van 8 betekenisvolle cijfers (niet startend met nulletjes). De invoer wordt gevalideerd en als de invoer niet geldig is krijgt de gebruiker de mogelijkheid om het getal opnieuw in te voeren. De applicatie haalt de individuele cijfers uit dit getal en houdt deze bij in een lijst. Let wel:

- de lijst mag geen dubbels bevatten
- de cijfers komen in de lijst voor in de volgorde waarin ze ook in het getal voorkomen

Uiteindelijk wordt de inhoud van de lijst getoond.

Voorbeeld uitvoer

```
Geef een getal van 8 betekenisvolle cijfers in > 150
Het getal moet 8 betekenisvolle cijfers bevatten...
Geef een getal van 8 betekenisvolle cijfers in > 123456789
Het getal moet 8 betekenisvolle cijfers bevatten...
Geef een getal van 8 betekenisvolle cijfers in > 00123456
Het getal moet 8 betekenisvolle cijfers bevatten...
Geef een getal van 8 betekenisvolle cijfers in > 85258258
In volgorde van voorkomen zijn de unieke cijfers in 85258258:
8 5 2
```

Voorbeeld uitvoer

```
Geef een getal van 8 betekenisvolle cijfers in > 21212121
In volgorde van voorkomen zijn de unieke cijfers in 21212121:
2 1
```

2.15. Stoelendans

Deze applicatie helpt een animator om het overzicht te bewaren bij een stoelendans. De 5 kinderen die deelnemen aan de stoelendans zijn "Mo", "To", "Bo", "Jo" en "Co". Bij elke ronde valt een kind af. De animator zal de naam van de afvaller opgeven en dan wordt getoond welke kinderen nog deelnemen aan de ronde die volgt. Wanneer een naam ingegeven wordt van een kind dat niet meespeelt wordt de animator daarop attent gemaakt en wordt er niet over gegaan naar de volgende ronde. De interactie met de applicatie verloopt als volgt:

Voorbeeld uitvoer

```
We starten de stoelendans met [Mo, To, Bo, Jo, Co]

Ronde 1 met [Mo, To, Bo, Jo, Co]
Geef de naam van de afvaller op: Jo

Ronde 2 met [Mo, To, Bo, Co]
Geef de naam van de afvaller op: An
Let op! An neemt niet deel aan de stoelendans...

Ronde 2 met [Mo, To, Bo, Co]
Geef de naam van de afvaller op: mo
Let op! mo neemt niet deel aan de stoelendans...

Ronde 2 met [Mo, To, Bo, Co]
Geef de naam van de afvaller op: Mo

Ronde 3 met [To, Bo, Co]
Geef de naam van de afvaller op: Co

Ronde 4 met [To, Bo]
```

```
Geef de naam van de afvaller op: Bo

Proficiat To! Je hebt de stoelendans gewonnen!
```

- maak gebruik van een `List<String> kinderen` om de namen van de kinderen bij te houden
  - de lijst bevat dus initieel 5 elementen
  - bij elke ronde wordt de afvaller verwijderd uit de lijst
  - wanneer de lijst nog 1 element bevat is de winnaar gekend!
- je kan de lijst van kinderen als volgt eenvoudig op het scherm zetten

```
// Om deze uitvoer te genereren
We starten de stoelendans met [Mo, To, Bo, Jo, Co]

// Maak je gebruik van volgend `printf` statement
System.out.printf("We starten de stoelendans met %s%n", kinderen);
```

2.16. Tevredenheidsscores

Vertrek van de code in `H04start_Tevredenheidsscores`.

Klanten kunnen voor de verschillende producten die aangeboden worden op een website een **tevredenheidsscore, i.e. een getal van 1 tot en met 5** ingeven. In deze applicatie krijg je een **lijst** met tevredenheidsscores en moet je een samenvattend overzicht geven van de gegeven scores. Deze samenvatting bevat het **totaal aantal gegeven scores**, en voor elke **score** het **aantal** keer dat ze werd gegeven.

De code om de lijst `scores` te maken en te vullen is al gegeven. De lijst heeft een willekeurige grootte (de lijst kan 10 tot 1000 scores bevatten), en wordt opgevuld met willekeurig gekozen scores. Telkens je de applicatie runt mag je dus een andere uitvoer verwachten.

- maak een array die 5 gehele getallen kan bevatten
- overloop de lijst `scores` en verhoog telkens het *juiste element* in de array met 1

Hier alvast enkele voorbeelden van de gewenste uitvoer.

Voorbeeld uitvoer

```
Samenvatting van 12 scores:
     score    aantal
         1         5
         2         2
         3         1
         4         1
```

```
        5        3
```

Voorbeeld uitvoer

```
Samenvatting van 951 scores:
     score    aantal
         1       164
         2       217
         3       207
         4       197
         5       166
```

2.17. Schermtijden

Een leerkracht merkt dat veel van zijn leerlingen te kampen hebben met een telefoonverslaving en beslist een challenge te organiseren: de leerlingen registreren **10 dagen lang hun dagelijkse schermtijd** (in minuten) met de bedoeling dat de schermtijd elke dag wat **verminderd**. Uiteraard slaagden niet alle leerlingen erin om hun schermtijd elke dag te doen dalen.

Onze applicatie gaat voor de geregistreerde tijden van 1 leerling de **langste reeks van opeenvolgende dagen met dalende schermtijd** bepalen. De applicatie toont eveneens de dag waarop de daling zich inzette en de totale tijdsafname.

```
int[] tijden = { 800, 750, 700, 720, 700, 680, 650, 620, 630, 620 };
```

Voorbeeld uitvoer

```
De langste reeks van opeenvolgende dagen dat je schermtijd daalde overspande 4
dagen.
Deze daling zette zich in van dag 4 op dag 5.
In deze periode daalde je schermtijd met 100 minuten.
```

Je kan de verschillende arrays uit dit document kopiëren en je code testen op correcte uitvoer. Voor leerlingen die de challenge tot een goed einde brachten, maar ook voor leerlingen die geen enkele daling konden realiseren is er afwijkende uitvoer.

```
int[] tijden = { 570, 575, 573, 560, 600, 610, 590, 580, 570, 560 };
```

Voorbeeld uitvoer

```
De langste reeks van opeenvolgende dagen dat je schermtijd daalde overspande 4
dagen.
```

```
Deze daling zette zich in van dag 6 op dag 7.
In deze periode daalde je schermtijd met 50 minuten.
```

```
int[] tijden = { 800, 750, 700, 680, 700, 680, 650, 700, 630, 620 };
```

Voorbeeld uitvoer

```
De langste reeks van opeenvolgende dagen dat je schermtijd daalde overspande 3
dagen.
Deze daling zette zich in van dag 1 op dag 2.
In deze periode daalde je schermtijd met 120 minuten.
```

```
int[] tijden = { 900, 850, 800, 790, 789, 780, 750, 740, 730, 630 };
```

Voorbeeld uitvoer

```
Proficiat! Challenge completed! Elke dag ging je schermtijd omlaag.
Uiteindelijk is je schermtijd met 270 minuten gedaald!
```

```
int[] tijden = { 850, 851, 860, 866, 902, 910, 920, 930, 931, 940 };
```

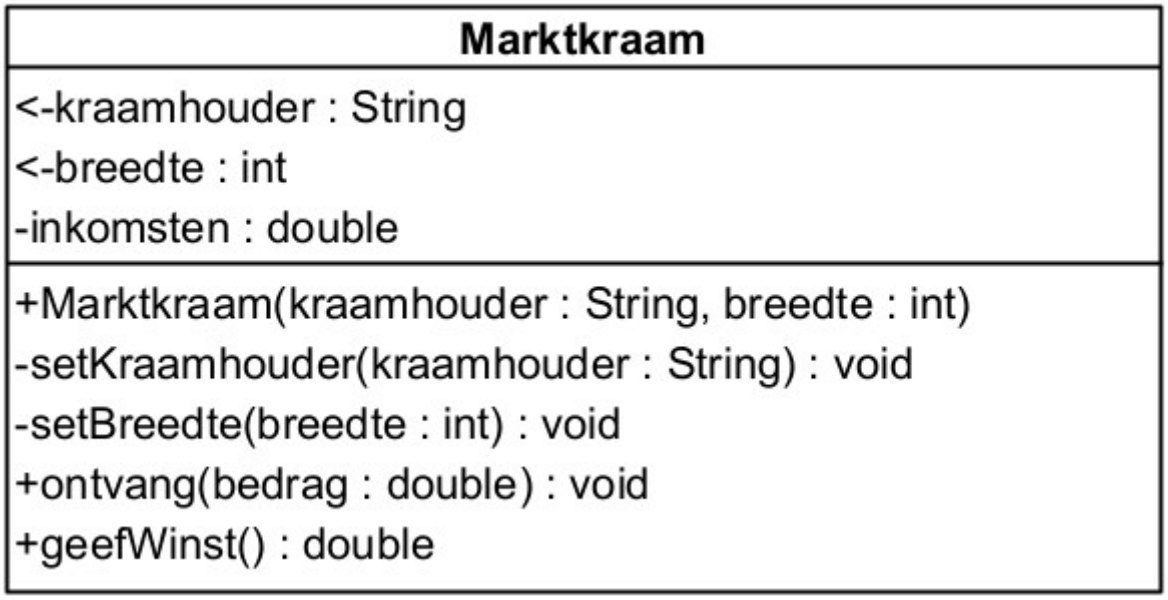
Voorbeeld uitvoer

```
Wat een ontgoocheling, je schermtijd ging enkel omhoog ipv omlaag!
```

2.18. Rommelmarkt

In deze applicatie gaan we een rommelmarkt simuleren. Op de rommelmarkt staan een aantal kraampjes. Van elk kraampje is de **kraamhouder** en de **breedte** (aantal meter) gekend. Telkens een aankoop gebeurt aan een kraampje wordt dit geregistreerd en op het einde geeft de applicatie een overzicht van de winsten en/of verliezen die geboekt zijn aan elk kraampje.

2.18.1. De domeinklasse `Marktkraam`



Hou rekening met onderstaande richtlijnen en test je klasse met de **unit testen** die je in het bestand `MarktkraamTest` vindt. Zie `H04start_Rommelmarkt`.

setters

Implementeer volgende regels in de gepaste **setters**.

- `kraamhouder` moet ingevuld zijn en mag niet uit enkele en alleen spaties bestaan; wanneer een ongeldige waarde wordt opgegeven wordt de `kraamhouder` ingesteld op `"anoniem"`
- `breedte` moet in het interval [2,10] liggen; wanneer een ongeldige waarde wordt opgegeven wordt de `breedte` ingesteld op `10`

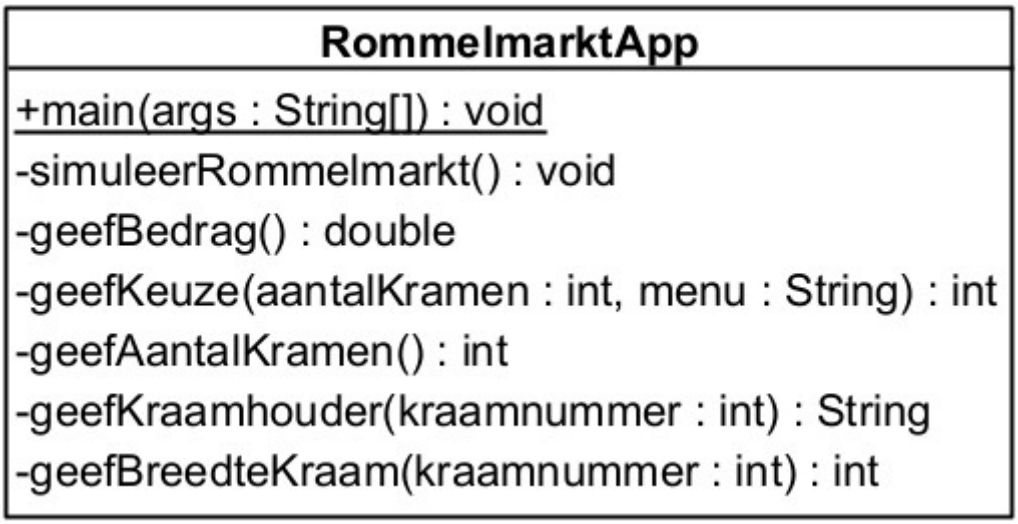
methode `void voegToeAanInkomsten(double bedrag)`

- als het bedrag positief is wordt het bedrag opgeteld bij `inkomsten`; met een negatief bedrag gebeurt niets

methode `double berekenWinst()`

- retourneert de winst die als volgt berekend kan worden:
  - de *kostprijs* van een marktkraam bedraagt 3 Euro/meter voor kramen met een lengte in het interval [2, 5] en 2.5 Euro voor bredere kramen
  - de winst is het verschil tussen de `inkomsten` en de *kostprijs*

2.18.2. De cui klasse `RommelmarktApp`



methode `int geefAantalKramen()`

De gebruiker kan ingeven hoeveel marktkramen er op de rommelmarkt staan. Dit aantal moet strikt positief zijn en dit wordt gevalideerd.

```
Geef het aantal kramen dat op de rommelmarkt staat op: -6
Het aantal moet groter dan 0 zijn, probeer opnieuw...
Geef het aantal kramen dat op de rommelmarkt staat op: 0
Het aantal moet groter dan 0 zijn, probeer opnieuw...
Geef het aantal kramen dat op de rommelmarkt staat op: 3

// de methode retourneert 3
```

methode `String geefKraamhouder(int kraamnummer)`

De gebruiker kan de houder van kraam `kraamnummer` ingeven, deze mag niet leeg zijn...

```
Geef de naam van de kraamhouder voor kraam 1 op:    // gebruiker tikt spaties
De naam van de kraamhouder moet je opgeven...
Geef de naam van de kraamhouder voor kraam 1 op: Jan

// de methode retourneert "Jan"
```

methode `geefBreedteKraam(int kraamnummer)`

De gebruiker kan de breedte van kraam `kraamnummer` ingeven, deze moet in het interval [2, 10] liggen...

```
Geef de  breedte van kraam 1 op: -5
De breedte moet tussen 2 en 10 meter liggen...
Geef de  breedte van kraam 1 op: 12
De breedte moet tussen 2 en 10 meter liggen...
Geef de  breedte van kraam 1 op: 2
```

```
 // de methode retourneert 2
```

methode `int geefKeuze(int aantalKramen, String menu)`

Het `menu` wordt aangeleverd via de parameter, alsook het `aantalKramen`. Je moet het `menu` tonen en vragen aan de gebruiker om zijn keuze in te voeren of 0 in te geven om te stoppen, de keuze moet dus in het interval [0, `aantalKramen`] liggen...

Voorbeeld uitvoer

```
Aan welk kraam wil je iets kopen?
1. Kraam van Jan.
2. Kraam van Sofie.
3. Kraam van Ayla.

Kies je kraam (of geef 0 om te stoppen): -2
Dit is geen geldige keuze! Probeer opnieuw...

Aan welk kraam wil je iets kopen?
1. Kraam van Jan.
2. Kraam van Sofie.
3. Kraam van Ayla.

Kies je kraam (of geef 0 om te stoppen): 4
Dit is geen geldige keuze! Probeer opnieuw...

Aan welk kraam wil je iets kopen?
1. Kraam van Jan.
2. Kraam van Sofie.
3. Kraam van Ayla.

Kies je kraam (of geef 0 om te stoppen): 3

 // de methode retourneert 3
```

methode - `double geefBedrag()`

De gebruiker heeft het bedrag dat deze spendeert aan een marktkraam in, dit moet strikt positief zijn...

Voorbeeld uitvoer

```
Geef op voor hoeveel Euro je kocht: -0,5
De prijs moet strikt positief zijn...
Geef op voor hoeveel Euro je kocht: 25,99

 // de methode retourneert 25.99
```

Stap 06 - `simuleerRommelmarkt()`

Deze methode wordt vanuit de `main` methode aangeroepen.

1. vraag naar het aantal marktkramen en initialiseer een array `Marktkraam[] markt` met de gepaste grootte
2. vraag je naar de naam en de breedte voor ***elk*** kraam en maak telkens een nieuw `Marktkraam` object die je op de juiste plaats in de array `markt` stopt
3. overloop de array en bouw een String die het menu bevat, in ons voorbeeld bevat deze String dus

   ```
   "1. Kraam van Jan.\n2. Kraam van Sofie.\n3. Kraam van Ayla."
   ```

4. laat de gebruiker herhaaldelijk kiezen aan welk kraam deze iets wenst te kopen, vraag voor welk bedrag de aankoop is en registreer dit bij het juiste marktkraam a.d.h.v. de `Marktkraam` methode `ontvang(bedrag)`
5. wanneer de gebruiker aangegeven heeft dat hij wenst te stoppen wordt het overzicht getoond:

Voorbeeld uitvoer

```
Kraamhouder Jan maakte 6,00 Euro verlies
Kraamhouder Sofie maakte 37,00 Euro winst
Kraamhouder Ayla maakte 136,74 Euro winst
```

2.19. Matrix transponeren

De getransponeerde matrix van een matrix bekom je door de **rijen en kolommen** van de oorspronkelijke matrix te **wisselen**. Schrijf een applicatie die van een matrix een getransponeerde matrix maakt.

vertrek van de code in `H04start_MatrixTransponeren`

Vul de gegeven code aan. De `matrix` die je moet transponeren is aangemaakt met een willekeurig aantal rijen en kolommen (beide zullen tussen 1 en 5 liggen) en opgevuld met willekeurige getallen (tussen 0 en 100). Zorg dat de lokale variabele `getransponeerdeMatrix` het resultaat bevat. Telkens je het programma runt mag je een andere uitvoer verwachten. Enkel voorbeeldjes:

Voorbeeld uitvoer

```
Matrix:
   53   58   78
   30   76   11
   20   59   17
   42   43   95

Getransponeerde matrix:
   53   30   20   42
```

```
   58   76   59   43
   78   11   17   95
```

Voorbeeld uitvoer

```
Matrix:
   28   32

Getransponeerde matrix:
   28
   32
```

Voorbeeld uitvoer

```
Matrix:
   90   78
   10   51

Getransponeerde matrix:
   90   10
   78   51
```

2.20. Puntenverwerking

Voor deze applicatie wordt vertrokken van een klaslijst met de namen van alle studenten. De lector kan voor elke student een resultaat voor de mid-term test ingeven. Dit is een getal tussen 0 en 20. Het programma hoeft deze resultaten niet bij te houden en zal enkel een lijst van geslaagde en niet geslaagde studenten maken. Een student is geslaagd als deze een resultaat heeft groter of gelijk aan 10. Let op de uitvoer wanneer alle studenten, of net geen enkele student, slaagt…

Voorbeeld uitvoer

```
Geef resultaat van Imani (0-20) > -6
Een geldig resultaat ligt tussen 0 en 20!
Geef resultaat van Imani (0-20) > 66
Een geldig resultaat ligt tussen 0 en 20!
Geef resultaat van Imani (0-20) > 20
Geef resultaat van Tanaka (0-20) > 5
Geef resultaat van Ayana (0-20) > 4
Geef resultaat van Yaro (0-20) > 12
Geef resultaat van Diara (0-20) > 14
Geef resultaat van Lente (0-20) > 55
Een geldig resultaat ligt tussen 0 en 20!
Geef resultaat van Lente (0-20) > 18
Geef resultaat van Jin (0-20) > 3
Geef resultaat van Sem (0-20) > 0
Geef resultaat van Zaur (0-20) > 20
Geef resultaat van Nyasha (0-20) > 19
```

```
Volgende studenten zijn geslaagd:
Imani
Yaro
Diara
Lente
Zaur
Nyasha
Volgende studenten zijn niet geslaagd:
Tanaka
Ayana
Jin
Sem
```

Voorbeeld uitvoer

```
Geef resultaat van Imani (0-20) > 3
Geef resultaat van Tanaka (0-20) > 5
Geef resultaat van Ayana (0-20) > 2
Geef resultaat van Yaro (0-20) > 1
Geef resultaat van Diara (0-20) > 0
Geef resultaat van Lente (0-20) > 0
Geef resultaat van Jin (0-20) > 5
Geef resultaat van Sem (0-20) > 9
Geef resultaat van Zaur (0-20) > 8
Geef resultaat van Nyasha (0-20) > 4
Geen enkel student slaagde...
Volgende studenten zijn niet geslaagd:
Imani
Tanaka
Ayana
Yaro
Diara
Lente
Jin
Sem
Zaur
Nyasha
```

Voorbeeld uitvoer

```
Geef resultaat van Imani (0-20) > 10
Geef resultaat van Tanaka (0-20) > 15
Geef resultaat van Ayana (0-20) > 18
Geef resultaat van Yaro (0-20) > 20
Geef resultaat van Diara (0-20) > 18
Geef resultaat van Lente (0-20) > 11
Geef resultaat van Jin (0-20) > 10
Geef resultaat van Sem (0-20) > 12
Geef resultaat van Zaur (0-20) > 13
Geef resultaat van Nyasha (0-20) > 20
```

```
Volgende studenten zijn geslaagd:
Imani
Tanaka
Ayana
Yaro
Diara
Lente
Jin
Sem
Zaur
Nyasha
Er waren geen niet geslaagden
```