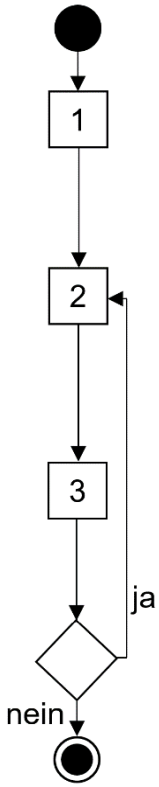
k-Means-Algorithmus

Einführung:

Bei KI-Systemen unterscheidet man zwischen dem wissensbasierten Ansatz und dem datenbasierten Ansatz (auch *maschinelles Lernen*). Teilbereiche des *maschinellen Lernens* sind u.a. das *Reinforcement Learning*, das *Supervised Learning* und das *Unsupervised Learning*. Beim *Unsupervised Learning* sind die Daten *ungelabelt* und der Algorithmus teilt diese in Gruppen, die sog. *Cluster*, ein. Ein typischer Algorithmus aus dem Bereich des *Unsupervised Learning* ist der *k-Means-Algorithmus*, um den es in dieser Unterrichtseinheit geht.

Bei der Anwendung des *k-Means-Algorithmus* gehst du wie folgt vor:



Initialisierung *(Initialize)*	Wähle zufällig *k* Datenpunkte als Clusterzentren aus.
Zuweisung *(Assign)*	Weise jedem Datenpunkt das Clusterzentrum mit der geringsten Entfernung zu.
Aktualisierung Clusterzenten *(Update)*	Berechne für jedes Clusterzentrum jeweils die Mittelwerte der x- und y-Koordinaten aller Datenpunkte eines Clusters. Dies sind die x- und y-Koordinate des aktualisierten Clusterzentrums.
Änderung?	Haben sich die Koordinaten der Clusterzentren geändert?

Arbeitsauftrag 1 (Clustering):

Teile mit Hilfe des *k-Means-Algorithmus* die Datenpunkte **A** bis **H** in **zwei** Cluster! Fülle dazu die folgenden Lücken aus. Die Lösungen sind farbig eingetragen.

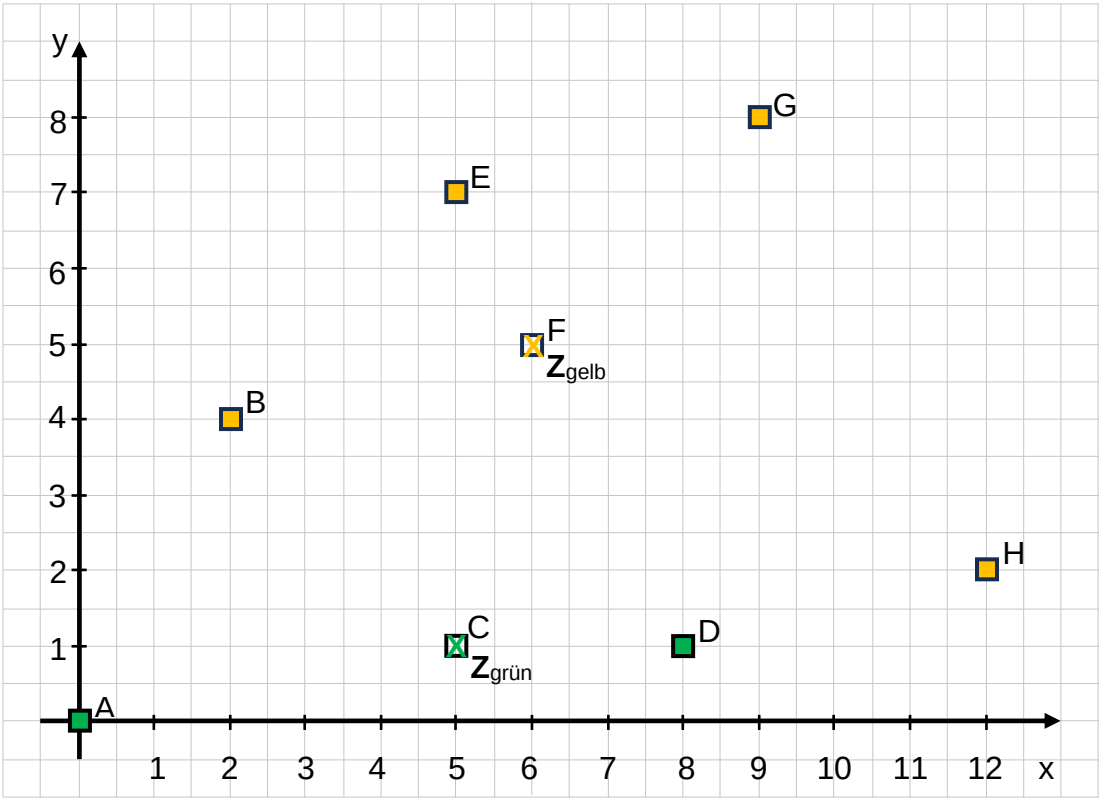
1 Initialisierung

k = 2

Wir gehen davon aus, dass per Zufall als Clusterzentrum $\mathbf{Z}_{grün}$ der Punkt **C** und als Clusterzentrum $\mathbf{Z}_{gelb}$ der Punkt **F** ausgewählt wurde. Trage diese als grünes bzw. gelbes „X“ im folgenden Koordinatensystem ein.

Andres D., Joachim S., Hennecke M, Universität Würzburg

1. Durchlauf



2 **Zuweisung**

Messe alle Entfernungen zwischen den Datenpunkten **A** bis **H** und den Clusterzentren $Z_{grün}$ und Z_{gelb} und trage diese in die Tabelle ein.

	A (0|0)	**B** (2|4)	**C** (5|1)	**D** (8|1)	**E** (5|7)	**F** (6|5)	**G** (9|8)	**H** (12|2)
$Z_{grün}$ (5|1)	5,1	4,2	0	3	6	4,1	8,1	7,1
Z_{gelb} (6|5)	7,8	4,1	4,1	4,5	2,2	0	4,2	6,7
$Z_{grün}$? Z_{gelb}?	$Z_{grün}$	Z_{gelb}	$Z_{grün}$	$Z_{grün}$	Z_{gelb}	Z_{gelb}	Z_{gelb}	Z_{gelb}

Das Cluster zum Clusterzentrum $Z_{grün}$ enthält die Datenpunkte **A, C, D**

Das Cluster zum Clusterzentrum Z_{gelb} enthält die Datenpunkte **B, E, F, G, H**

Markiere im vorherigen Koordinatensystem die Datenpunkte in der Farbe des zugehörigen Clusterzentrums.

Andres D., Joachim S., Hennecke M, Universität Würzburg

3 **Aktualisierung der Clusterzentren**

Berechne die neuen Koordinaten von $\mathbf{Z}_{grün}$!

x = (0 + 5 + 8) : 3 = 4,3

y = (0 + 1 + 1) : 3 = 0,7

Berechne die neuen Koordinaten von $\mathbf{Z}_{gelb}$!

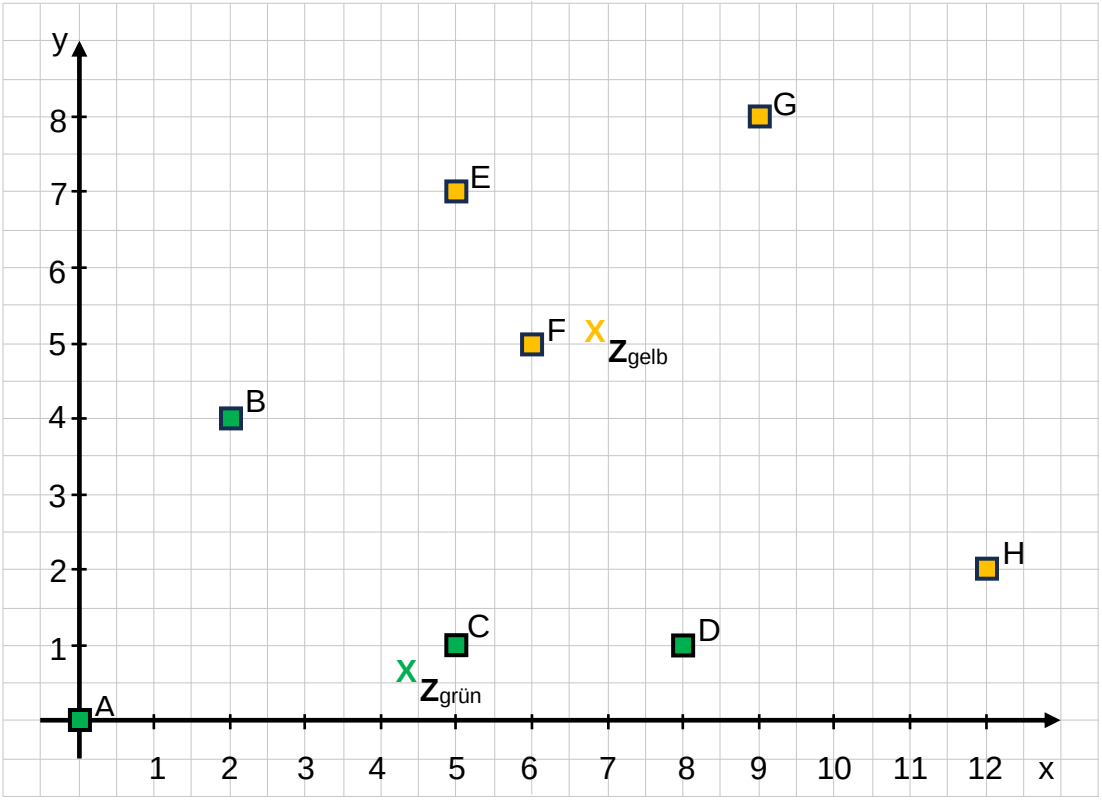
x = (2 + 5 + 6 + 9 + 12) : 5 = 6,8

y = (4 + 7 + 5 + 8 + 2) : 5 = 5,2

Haben sich die Koordinaten der Clusterzentren geändert? ☒ ja ☐ nein

2. Durchlauf

Trage die aktualisierten Clusterzentren als grünes bzw. gelbes „X“ im folgenden Koordinatensystem ein.



Andres D., Joachim S., Hennecke M, Universität Würzburg

2 **Zuweisung** zu den aktualisierten Clusterzentren

	A (0|0)	**B** (2|4)	**C** (5|1)	**D** (8|1)	**E** (5|7)	**F** (6|5)	**G** (9|8)	**H** (12|2)
$\mathbf{Z}_{grün}$(4,3|0,7)	4,4	4,0	0,8	3,7	6,3	4,6	8,7	7,8
$\mathbf{Z}_{gelb}$(6,8|5,2)	8,6	4,9	4,6	4,4	2,5	0,8	3,6	6,1
$\mathbf{Z}_{grün}$? $\mathbf{Z}_{gelb}$?	$\mathbf{Z}_{grün}$	$\mathbf{Z}_{grün}$	$\mathbf{Z}_{grün}$	$\mathbf{Z}_{grün}$	$\mathbf{Z}_{gelb}$	$\mathbf{Z}_{gelb}$	$\mathbf{Z}_{gelb}$	$\mathbf{Z}_{gelb}$

Das Cluster zum Clusterzentrum $\mathbf{Z}_{grün}$ enthält die Datenpunkte **A, B, C, D**

Das Cluster zum Clusterzentrum $\mathbf{Z}_{gelb}$ enthält die Datenpunkte **E, F, G, H**

Markiere im vorherigen Koordinatensystem die Datenpunkte in der Farbe des zugehörigen Clusterzentrums.

3 **Aktualisierung der Clusterzentren**

Berechne die neuen Koordinaten von $\mathbf{Z}_{grün}$!

x = (0 + 2 + 5 + 8) : 4 = 3,8

y = (0 + 4 + 1 + 1) : 4 = 1,5

Berechne die neuen Koordinaten von $\mathbf{Z}_{gelb}$!

x = (5 + 6 + 9 + 12) : 4 = 8

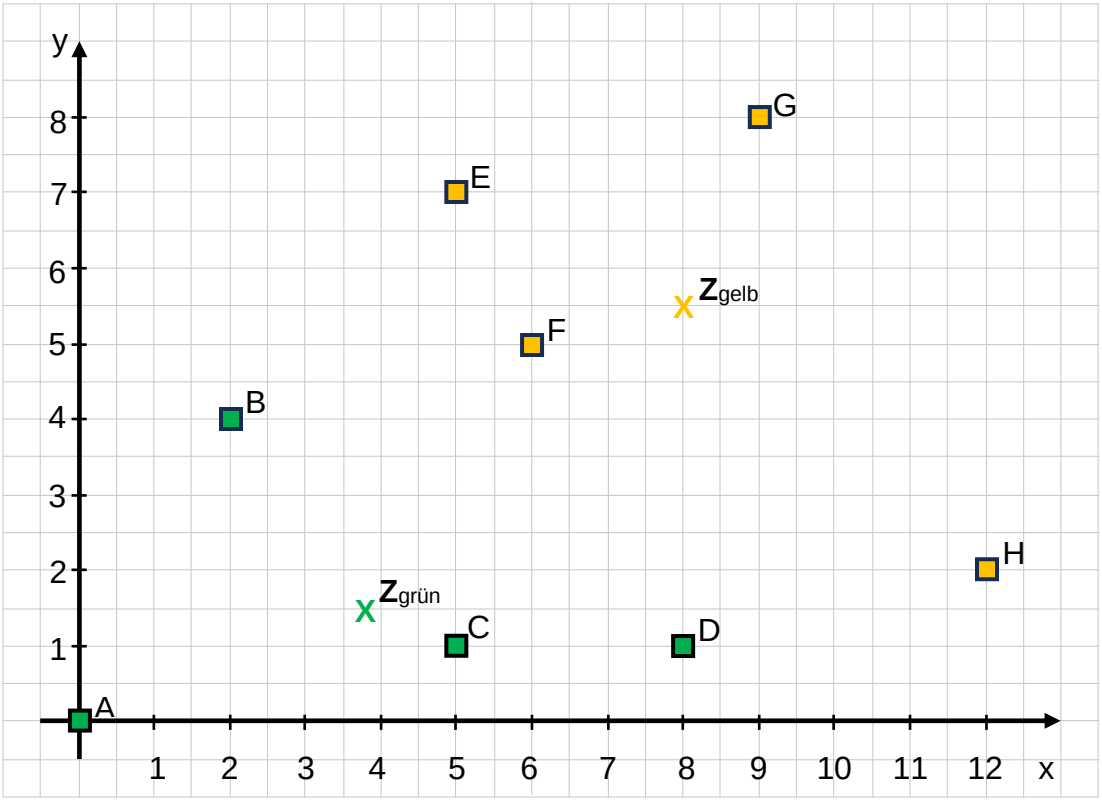
y = (7 + 5 + 8 + 2) : 4 = 5,5

Haben sich die Koordinaten der Clusterzentren geändert? ☒ ja ☐ nein

Andres D., Joachim S., Hennecke M, Universität Würzburg

3. Durchlauf

Trage die aktualisierten Clusterzentren als grünes bzw. gelbes „X“ im folgenden Koordinatensystem ein.



2 **Zuweisung** zu den aktualisierten Clusterzentren

	A (0|0)	**B** (2|4)	**C** (5|1)	**D** (8|1)	**E** (5|7)	**F** (6|5)	**G** (9|8)	**H** (12|2)
$\mathbf{Z}_{grün}$(3,8|1,5)	4,1	3,1	1,3	4,2	5,6	4,1	8,3	8,2
$\mathbf{Z}_{gelb}$(8|5,5)	9,7	6,2	5,4	4,5	3,4	2,1	2,7	5,3
$\mathbf{Z}_{grün}$? $\mathbf{Z}_{gelb}$?	$\mathbf{Z}_{grün}$	$\mathbf{Z}_{grün}$	$\mathbf{Z}_{grün}$	$\mathbf{Z}_{grün}$	$\mathbf{Z}_{gelb}$	$\mathbf{Z}_{gelb}$	$\mathbf{Z}_{gelb}$	$\mathbf{Z}_{gelb}$

Das Cluster zum Clusterzentrum $\mathbf{Z}_{grün}$ enthält die Datenpunkte **A, B, C, D**

Das Cluster zum Clusterzentrum $\mathbf{Z}_{gelb}$ enthält die Datenpunkte **E, F, G, H**

Markiere im vorherigen Koordinatensystem die Datenpunkte in der Farbe des zugehörigen Clusterzentrums.

Andres D., Joachim S., Hennecke M, Universität Würzburg

3 **Aktualisierung der Clusterzentren**

Berechne die neuen Koordinaten von $\mathbf{Z}_{grün}$!

x = (0 + 2 + 5 + 8) : 4 = 3,8

y = (0 + 4 + 1 + 1) : 4 = 1,5

Berechne die neuen Koordinaten von $\mathbf{Z}_{gelb}$!

x = (5 + 6 + 9 + 12) : 4 = 8

y = (7 + 5 + 8 + 2) : 4 = 5,5

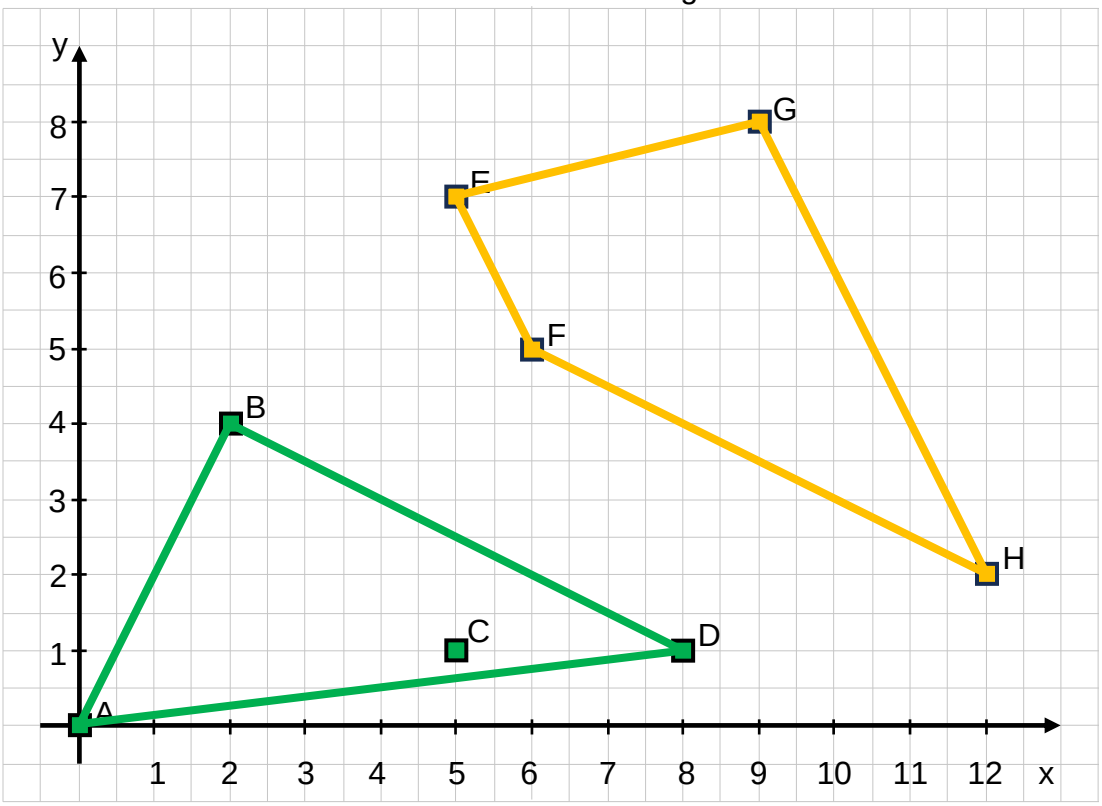
Haben sich die Koordinaten der Clusterzentren geändert? ☐ ja ☒ nein

Ergebnis

Die Datenpunkte sind final den zwei Clusterzentren $\mathbf{Z}_{grün}$ (3,8|1,5) und $\mathbf{Z}_{gelb}$ (8|5,5) zugewiesen.

Arbeitsauftrag 2 (Cluster-Hülle):

Zeichne die Cluster-Hüllen ein! Tipp: Verbinde zunächst alle Datenpunkte eines Clusters und zeichne dann nur die äußersten Linien farbig nach.

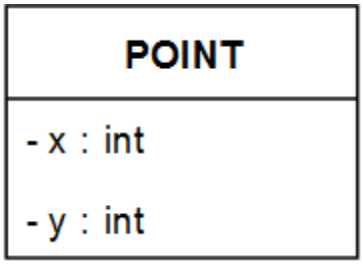


Andres D., Joachim S., Hennecke M, Universität Würzburg

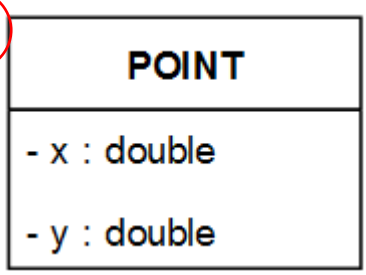
Arbeitsauftrag 3 (Modellierung):

a) Für jeden Datenpunkt müssen die x- und y-Koordinaten gespeichert werden. Diskutiere: Welches Klassendiagramm würdest du dafür bevorzugen?

a.

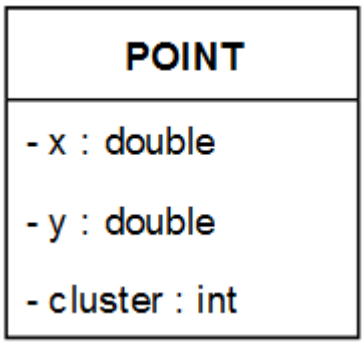


b.

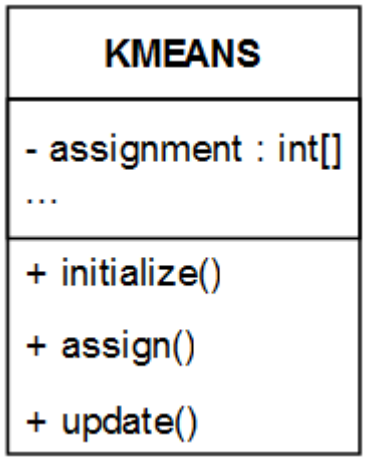


b) Für jeden Datenpunkt muss es möglich sein, dessen Zuweisung zu einem Cluster zu speichern. Welche der drei untenstehenden Möglichkeiten ermöglicht eine effektive Zuweisung, die keine Prinzipien der Objektorientierung verletzt?

a. Speicherung in der Klasse POINT



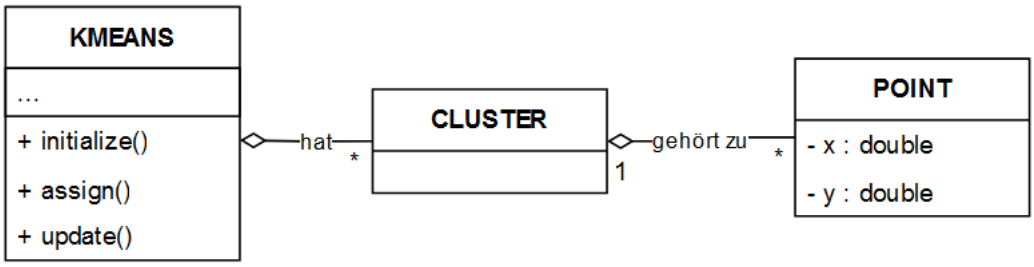
b. Speicherung in der Klasse KMEANS als Array der Länge #Eingabedaten.



Die Clusterzuordnung ist lediglich ein „Hilfswert“ für den *k-Means-Algorithmus*. Somit sollte dieser auch innerhalb der Klasse KMEANS verwaltet werden.

Andres D., Joachim S., Hennecke M, Universität Würzburg

c. Speicherung in separater Klasse CLUSTER, die die zum Cluster zugehörigen Punkte referenziert.



c) Die Datenpunkte der Eingabedaten müssen in einer Datenstruktur gehalten werden. Überlege, welche Datenstrukturen du bereits kennst. Für welche würdest du dich entscheiden?

Je nach Vorwissen kennen Schülerinnen und Schüler bereits die Datenstruktur des Stapels, der Warteschlange, der Liste, des Baumes und des Graphen. Für die gegebene Problematik bietet sich die Liste als Datenstruktur an.

Da die Daten nach einmaligem hinzufügen allerdings nie aus der Liste entnommen oder anderweitig bearbeitet werden, muss nicht die Struktur einer vollständig implementierten Liste – wie der einfach verketteten Liste – genutzt werden, die Daten können auch lediglich in einem Array gehalten werden. Diese Lösung erzeugt den geringsten Overhead für das Programm.

Andres D., Joachim S., Hennecke M, Universität Würzburg

Arbeitsauftrag 4 (Pseudocode):

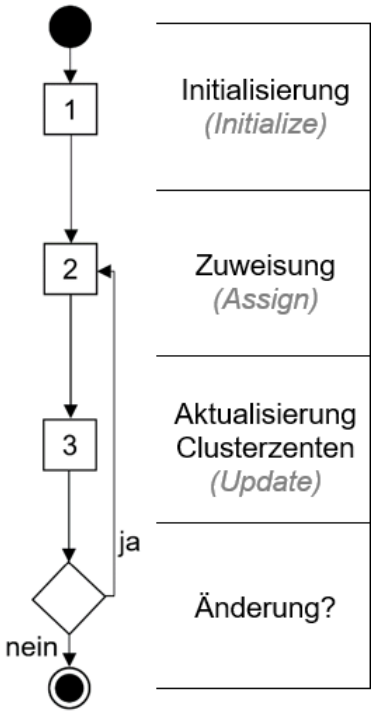
Ergänze den Pseudocode für folgende Methoden

```
def initialize()

    Wiederhole k mal:

            Wähle einen zufälligen Punkt aus der Liste data

            aus.

            Erstelle an dessen Koordinaten ein Clusterzentrum.


def assign()

    Für jeden Punkt in der Liste data:

            Speichere die Zugehörigkeit zu dem Zentrum mit

            minimalem Abstand.


def update()

    Für alle Zentren in der Liste centers:

            Berechne jeweils den Mittelwert der x-

            y- Koordinaten aller Punkte, die zu diesem

            Zentrum gehören.

            Setze die x- und y- Koordinaten des aktuellen

            Zentrums auf die berechneten Mittelwerte.
```


Andres D., Joachim S., Hennecke M, Universität Würzburg

Arbeitsauftrag 5 (Hilfestellung fürs Programmieren):

Ordne die folgenden Ablaufschritte den einzelnen Phasen des Algorithmus zu. Schreibe die Schrittnummer in die Kreise.



1

Eine zufällige Ganzzahl zwischen 0 und k-1 erzeugen

```
Random random = new Random();
int index = random.nextInt(k);
```

3

Den Mittelwert der Zahlen x_1, ..., x_n berechnen.

$$\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i = \frac{x_1 + x_2 + \ldots + x_n}{n}$$

2

Den euklidischen Abstand zwischen zwei Punkten $P_1 = (x_1, y_1)$ und $P_2 = (x_2, y_2)$ berechnen.

$$d(P_1, P_2) = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$$

$\sqrt{a}$ = Math.sqrt(double a)

a^2 = Math.pow(double a, 2)

3

Den größtmöglichen Wert einer Gleitkommazahl bekommen.

d_{max} = Double.MAX_VALUE

Andres D., Joachim S., Hennecke M, Universität Würzburg

Arbeitsauftrag 6 (Implementierung in *BlueJ*):

Öffne das *BlueJ*-Projekt „kMeans_BlueJ_Vorlage“. Dort findest du sieben Klassen. Deine Aufgabe ist es die fehlenden Methoden der Klasse KMEANS zu ergänzen. Sie entsprechen den einzelnen Schritten des *k-Means-Algorithmus*. Der vollständige Code befindet sich auch unter **https://go.uniwue.de/ki** in dem Projekt kMeans_BlueJ_Loesung.

a) Ergänze die Methode `initialize()` der Klasse KMEANS. Achte darauf, dass du das Zentrum als einen **neuen** Punkt anlegst, da dessen x- und y-Koordinaten im Verlauf des Algorithmus geändert werden.

```
public void initialize(){
    //Variante 1: Wähle k zufällige Punkte aus den Startpunkten als Anfangs"mittelwerte"
    Random random = new Random();
    for(int i = 0; i < k; i++){
        int index = random.nextInt(data.length);
        POINT zentrum = new POINT(data[index].getX(), data[index].getY());
        centers[i] = zentrum;
    }
}
```

Problem: Bei wenigen Datenpunkten ist es wahrscheinlich, dass ein Datenpunkt mehrmals als Zentrum ausgewählt wird.

```
public void initialize(){
    //Variante 2: Wähle k zufällige, verschiedene Punkte aus den Startpunkten als Anfangs"mittelwerte"
    Random random = new Random();
    int i = 0;
    while(i<k){
        int rNeu = random.nextInt(data.length);
        POINT zentrum = new POINT(data[rNeu].getX(), data[rNeu].getY());
        if(!contains(centers, zentrum)){
            centers[i] = zentrum;
            i++;
        }
    }
}
```

Methode `contains` ist in dem Projekt kmeans_BlueJ_Vorlage bereits implementiert.

b) Ergänze die Methode `assign()`. Das Array `assignment` ist zu Beginn komplett mit „-1“ gefüllt und muss von dir auf die Clusterzahl (= Index des entsprechenden Zentrums in `centers`) des nächsten Zentrums gesetzt werden.

Hinweis: Die Methode `getDistance(POINT p)` *der Klasse POINT berechnet den euklidischen Abstand zwischen dem Punkt selbst und einem anderen Punkt p.*

```
public void assign(){
    //Zuordnung: Jeder Punkt wird zu dem Clusterzentrum geordnet, dem es am nächsten ist
    //Den Abstand berechnet die Methode getDistance(POINT p) der Klasse POINT
    for(int i = 0; i < data.length; i++){
        double minabs = Double.MAX_VALUE;
        for(int j = 0; j < centers.length; j++){
            double abstand = data[i].getDistance(centers[j]);
            if(abstand < minabs){
                minabs = abstand;
                assignment[i] = j; //der Index des enstsprechnenden Zentrums = Clusternummer des Zentrums wird als Zuordnung gespeichert
            }
        }
    }
}
```


Andres D., Joachim S., Hennecke M, Universität Würzburg

c) Ergänze die Methode `update()` und bestimme die Laufzeit der von dir implementierten Lösung. Zusatzaufgabe: Kannst du die Laufzeit noch verbessern?

```
public void update(){
    //Mittelpunkt von jedem Cluster berechnen. Dieser Punkt wird das neue Zentrum des Clusters
    for(int i = 0; i < centers.length; i++){ //gehe durch alle bisherigen Zentren
        double meansX = 0;
        double meansY = 0;
        int count = 0; //wie viele Punkte pro Cluster
        //Mittelwertsberechnung für die x- und y-Koordinate
        for(int j = 0; j < data.length; j++){
            if(assignment[j] == i){
                meansX += data[j].getX();
                meansY += data[j].getY();
                count++;
            }
        }
        meansX = meansX/count;
        meansY = meansY/count;
        centers[i].setX(meansX);
        centers[i].setY(meansY);
    }
}
```

Laufzeit O(nk) mit n = data.length und k = centers.length

Zusatzaufgabe:

```
public void update(){
    POINT[] centersNeu = new POINT[centers.length];
    int[] anzahl = new int[centers.length];
    for(int i = 0; i < centers.length; i++){
        centersNeu[i] = new POINT(0,0);
    }
    for(int i = 0; i < data.length; i++){
        centersNeu[assignment[i]].setX(centersNeu[assignment[i]].getX() + data[i].getX());
        centersNeu[assignment[i]].setY(centersNeu[assignment[i]].getY() + data[i].getY());
        anzahl[assignment[i]] = anzahl[assignment[i]] + 1;
    }
    for(int i = 0; i < centersNeu.length; i++){
        centersNeu[i].setX(centersNeu[i].getX()/anzahl[i]);
        centersNeu[i].setY(centersNeu[i].getY()/anzahl[i]);
    }
    centers = centersNeu;
}
```

Laufzeit O(n+k) mit n = data.length und k = centers.length

Testen: Zum Testen deiner Methoden kannst du entweder den Konstruktor der Klasse GUI ohne Parameter aufrufen oder ein Objekt der Klasse DATAHANDLER erstellen und dieses zusammen mit deinem gewünschten *k* dem Konstruktor der Klasse GUI übergeben. Mittels der aufgehenden Visualisierung siehst du die Auswirkungen deines Codes.

Andres D., Joachim S., Hennecke M, Universität Würzburg