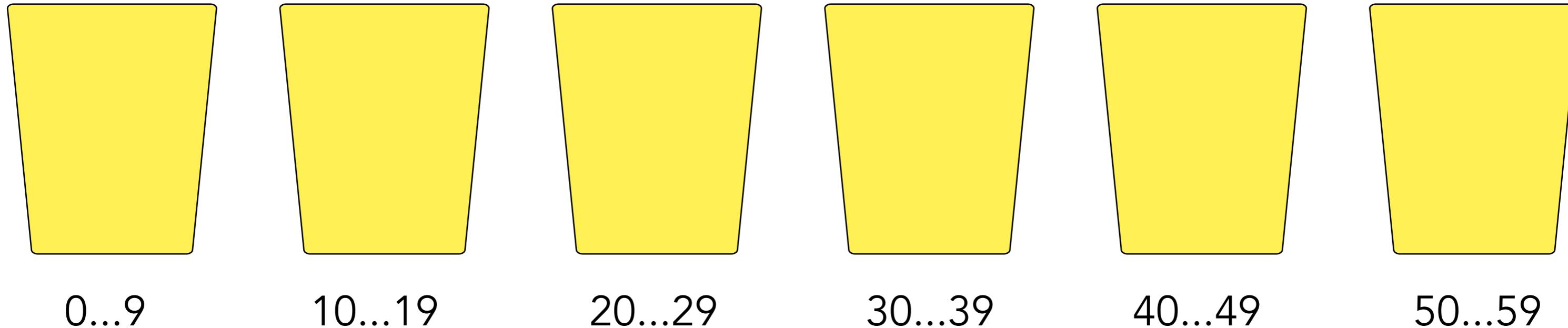
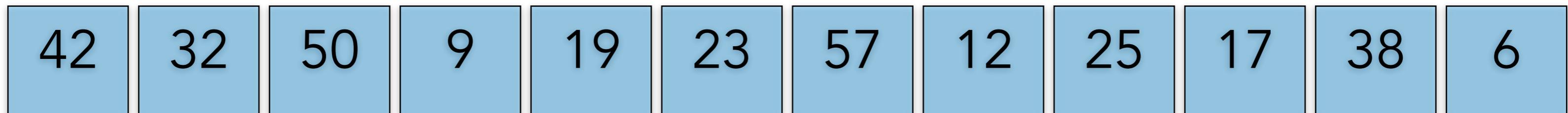


5. Sortieren und Suchen

5.6 Der Bucketsort

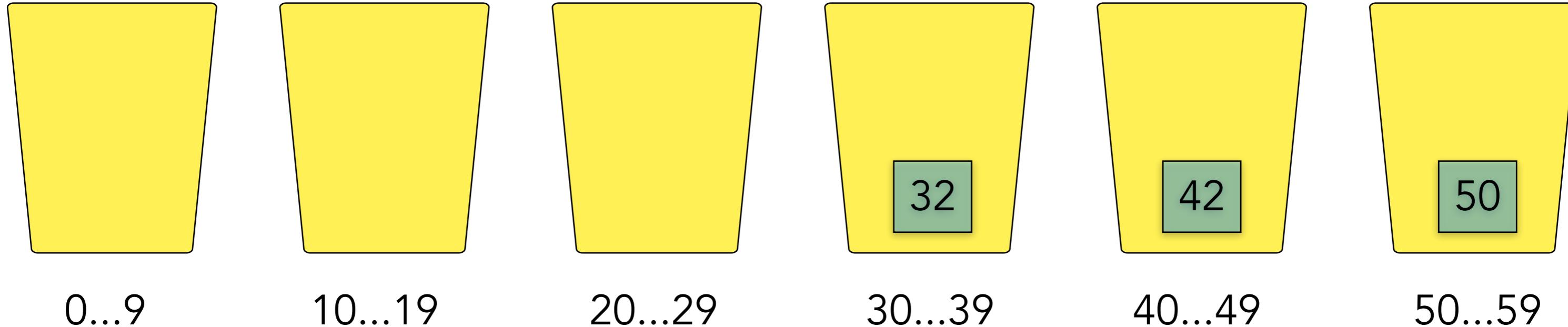
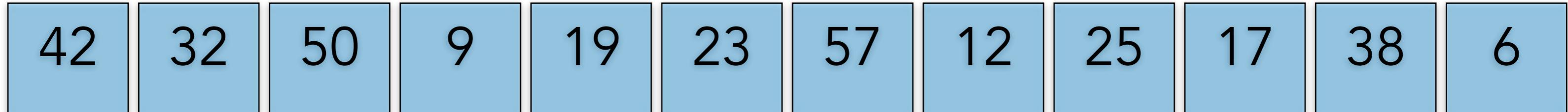
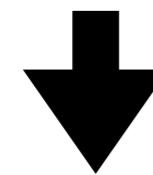
Einführung in die Objektorientierte Programmierung (OOP)

5.6 Bucket sort



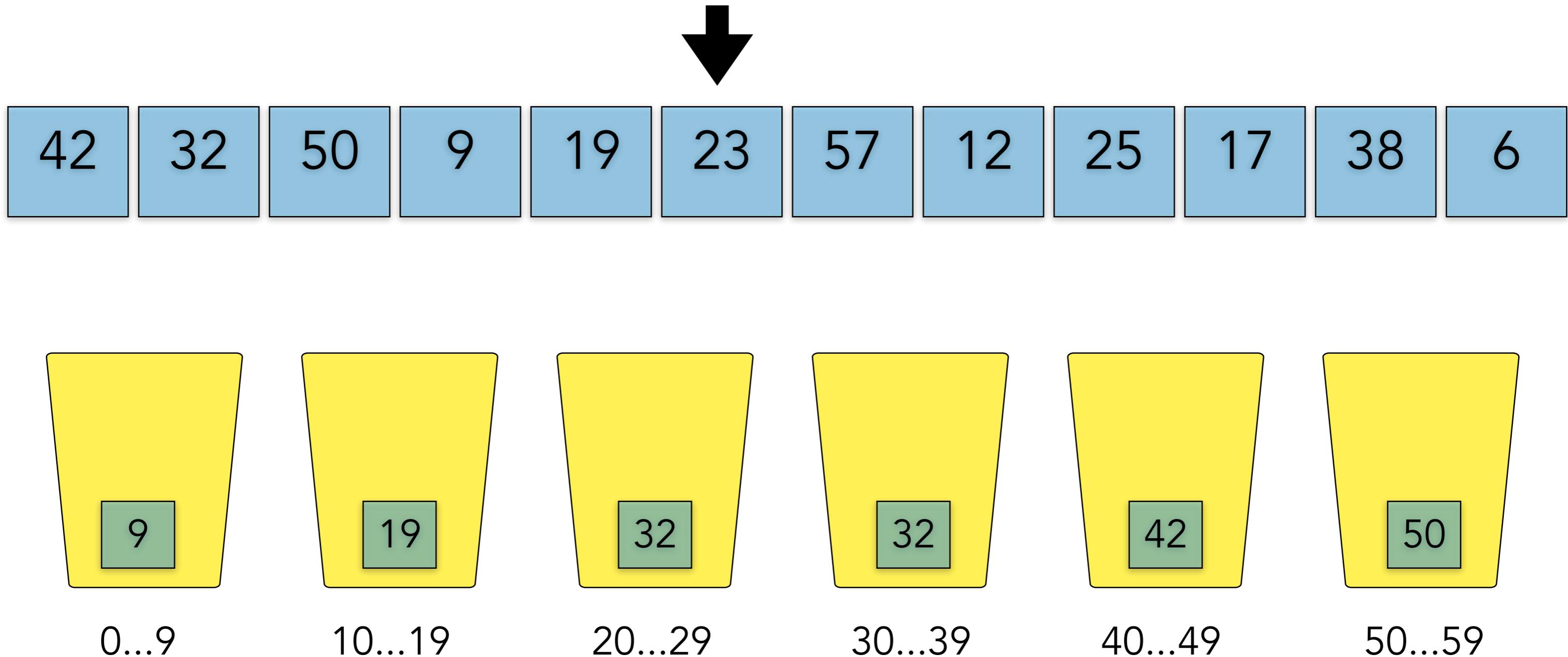
Einführung in die Objektorientierte Programmierung (OOP)

5.6 Bucket sort



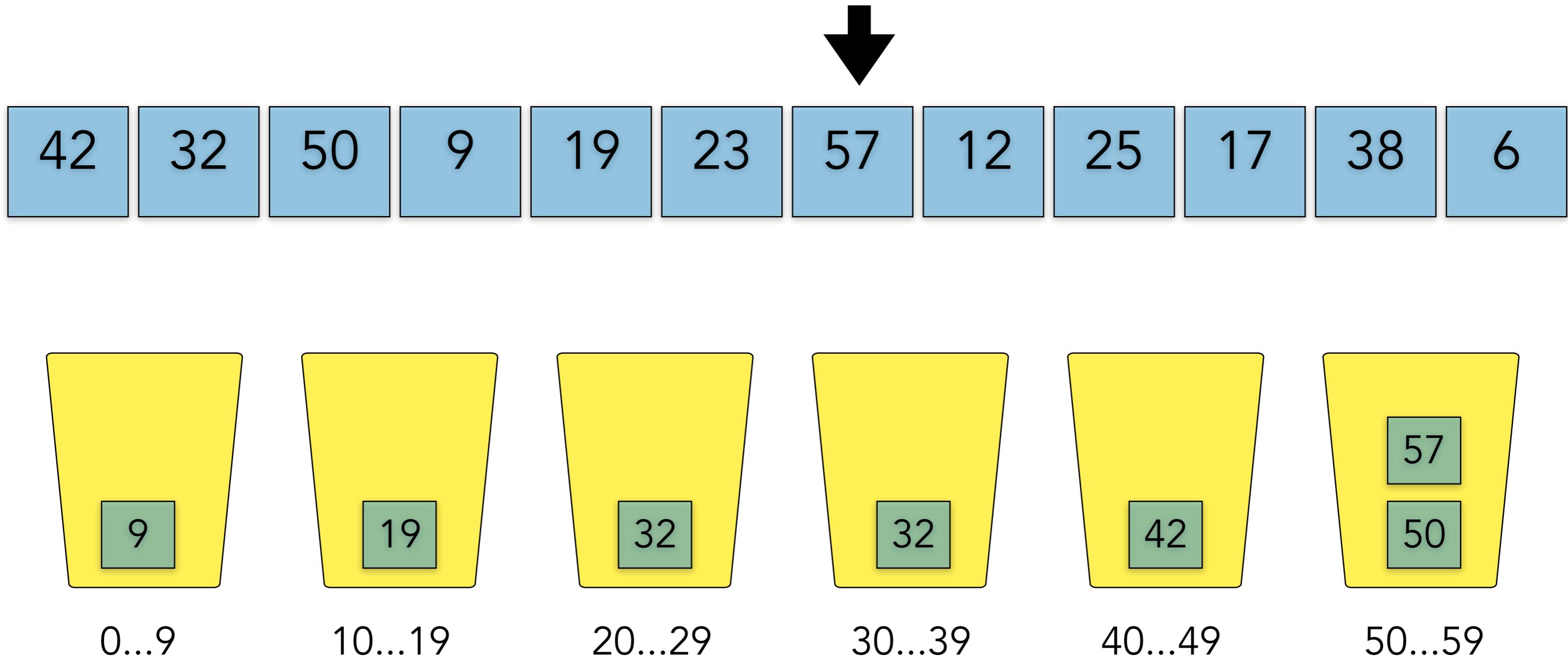
Einführung in die Objektorientierte Programmierung (OOP)

5.6 Bucket sort



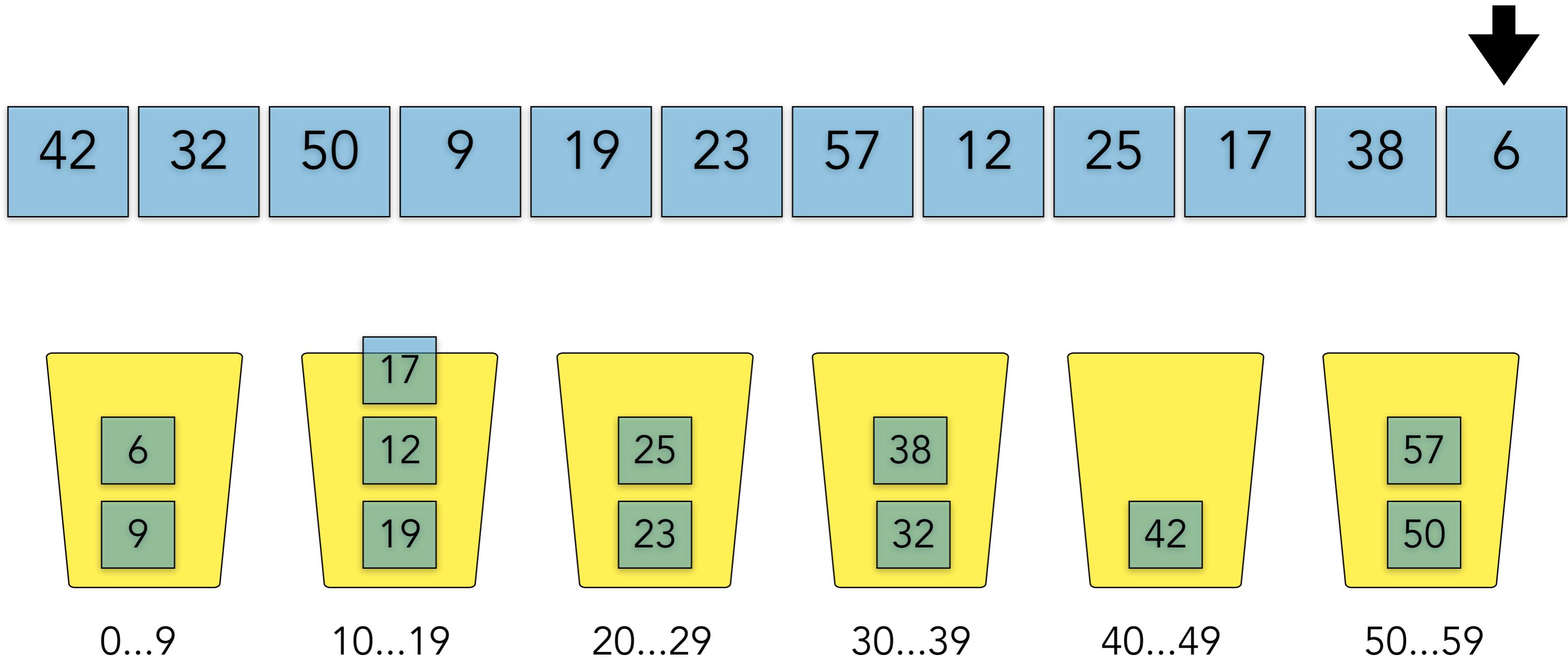
Einführung in die Objektorientierte Programmierung (OOP)

5.6 Bucket sort



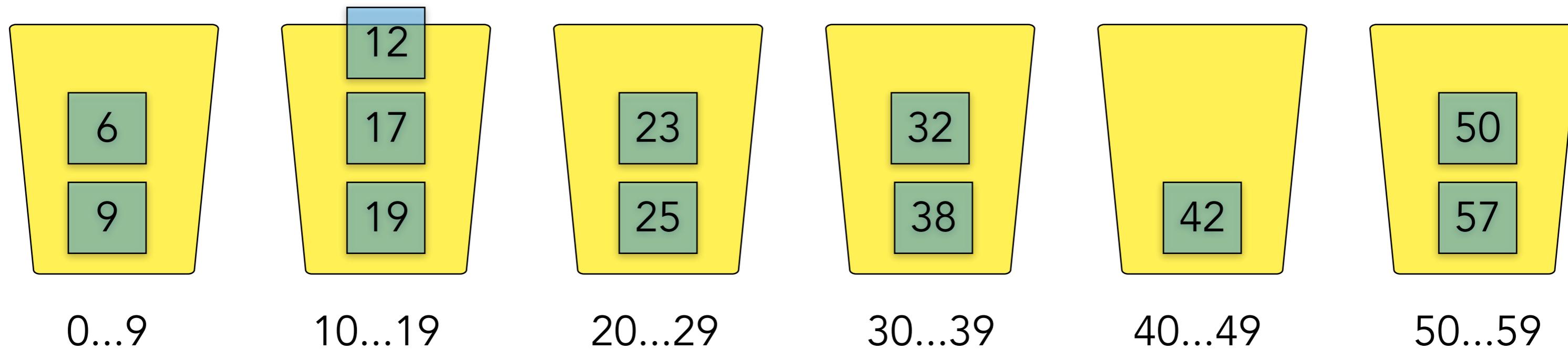
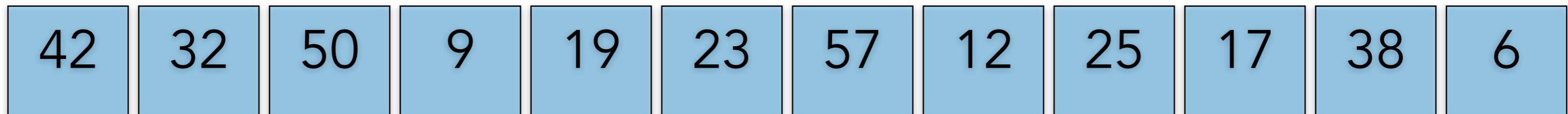
Einführung in die Objektorientierte Programmierung (OOP)

5.6 Bucketsort



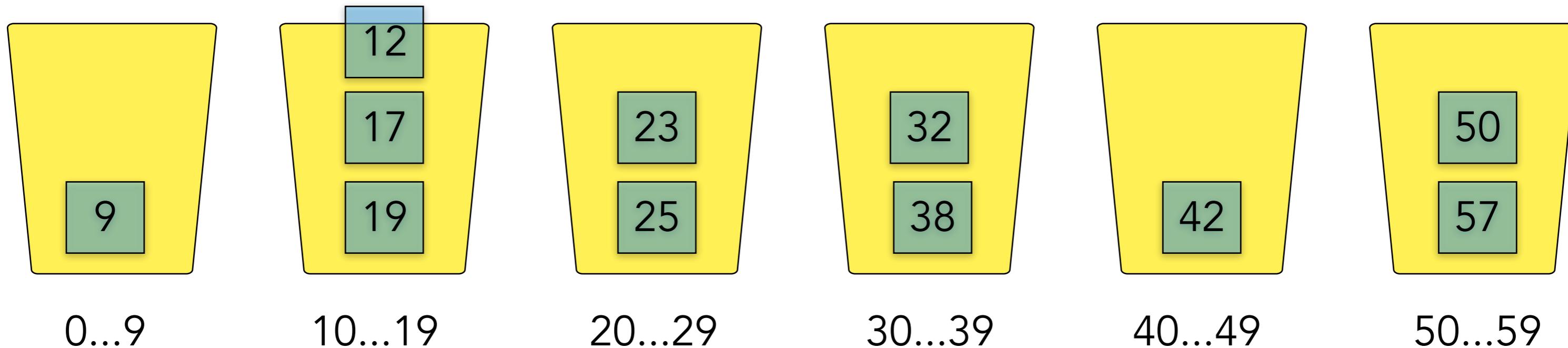
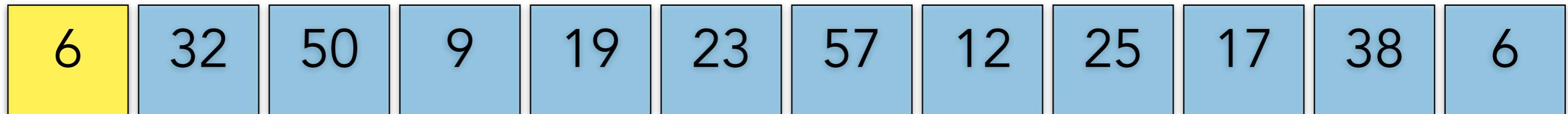
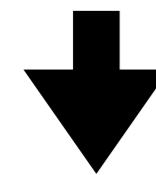
Einführung in die Objektorientierte Programmierung (OOP)

5.6 Bucket sort



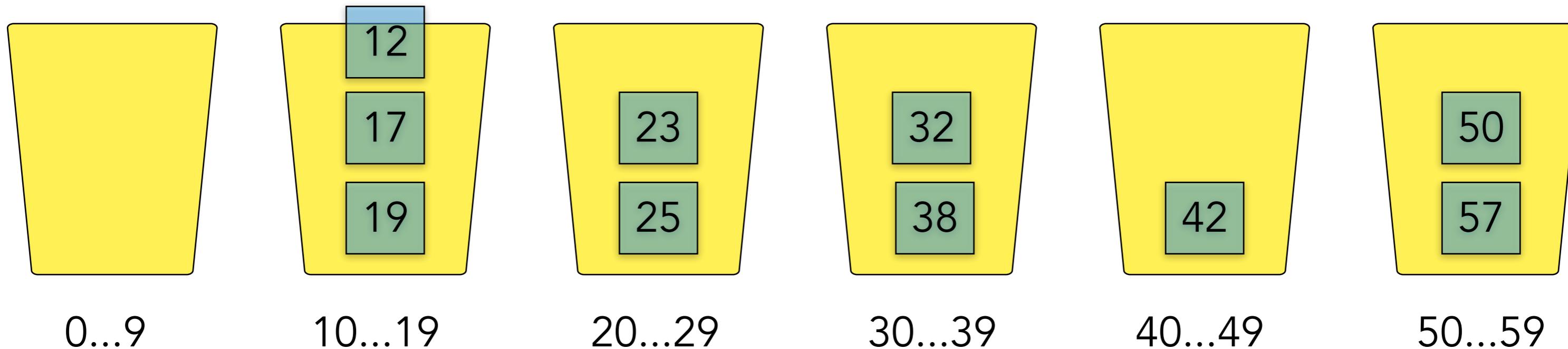
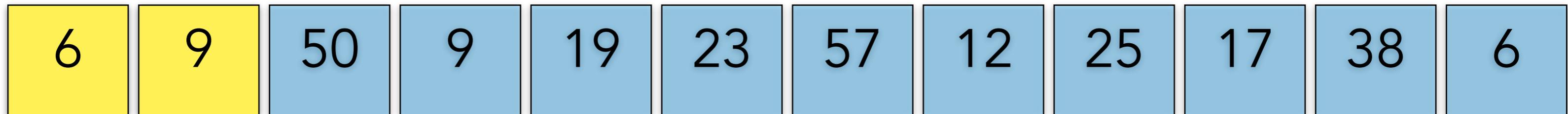
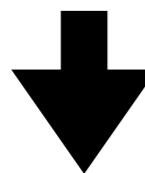
Einführung in die Objektorientierte Programmierung (OOP)

5.6 Bucket sort



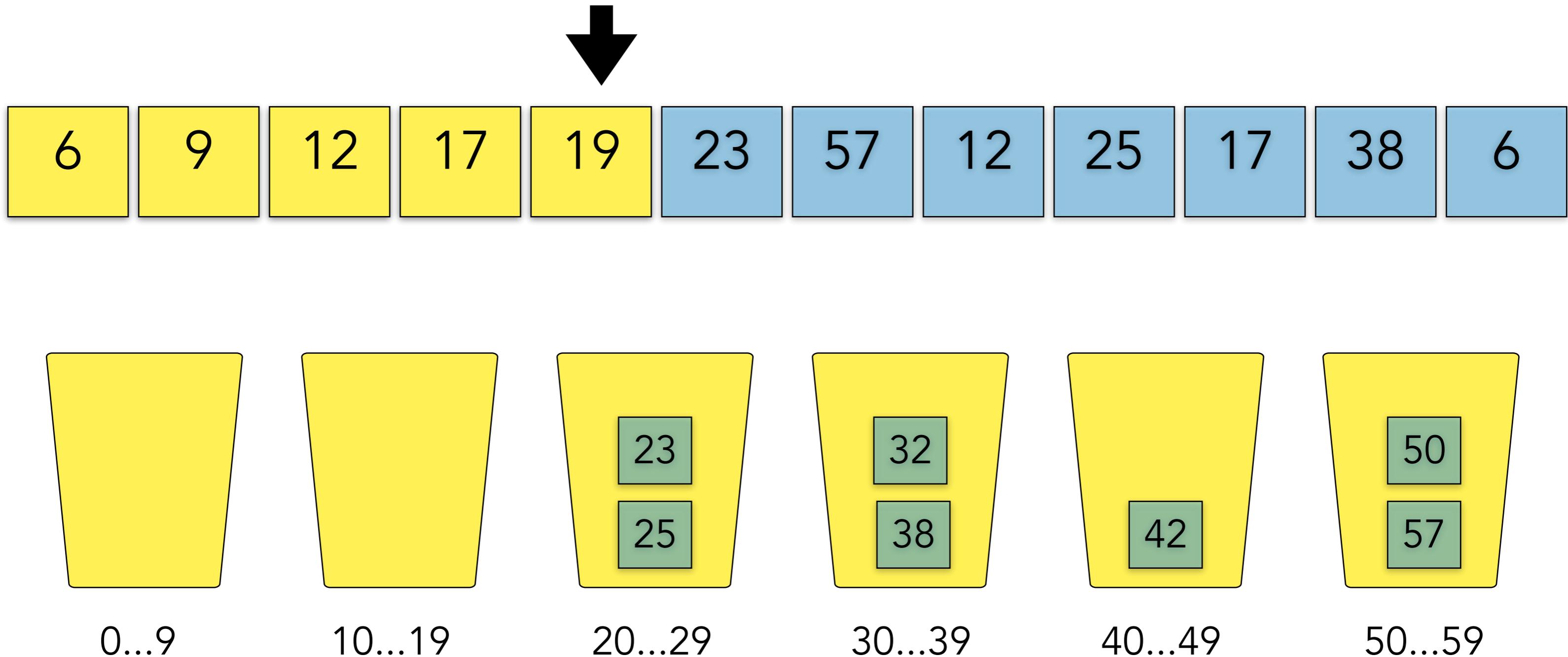
Einführung in die Objektorientierte Programmierung (OOP)

5.6 Bucket sort



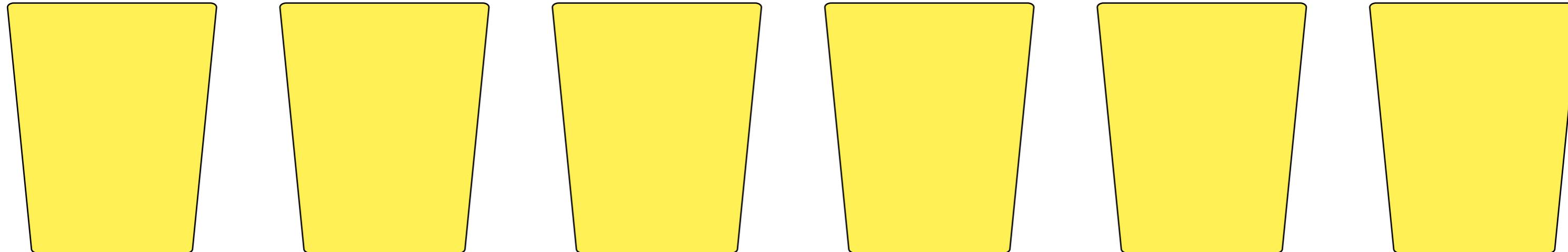
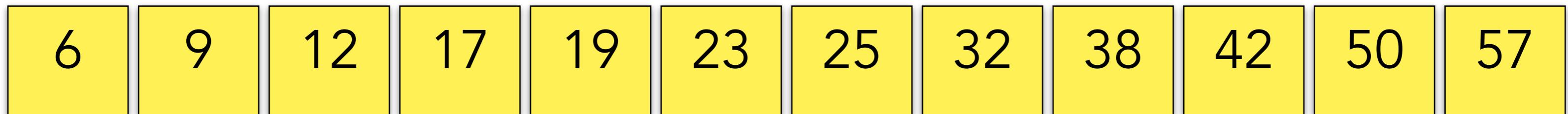
Einführung in die Objektorientierte Programmierung (OOP)

5.6 Bucket sort



Einführung in die Objektorientierte Programmierung (OOP)

5.6 Bucket sort



0..9

10..19

20..29

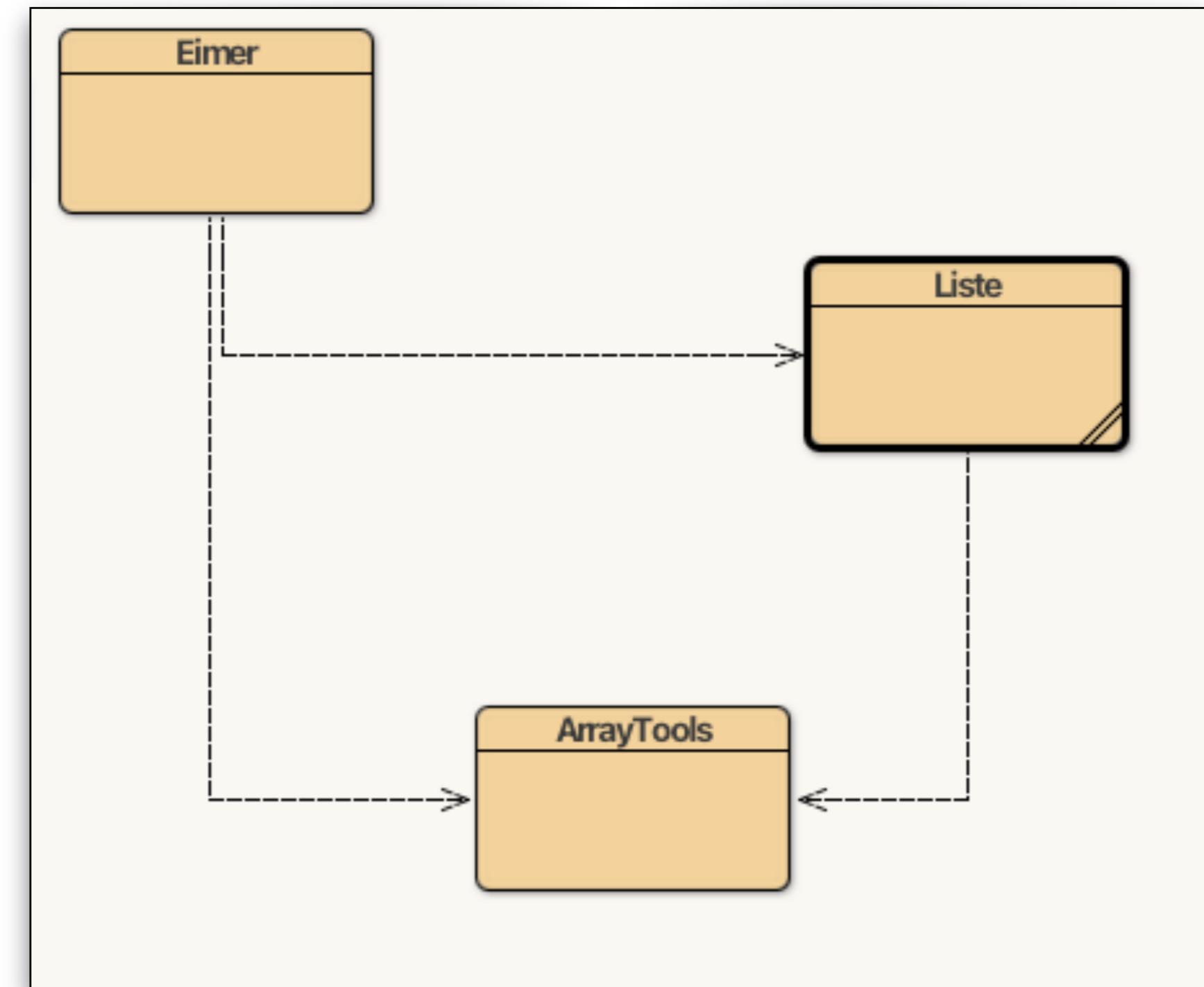
30..39

40..49

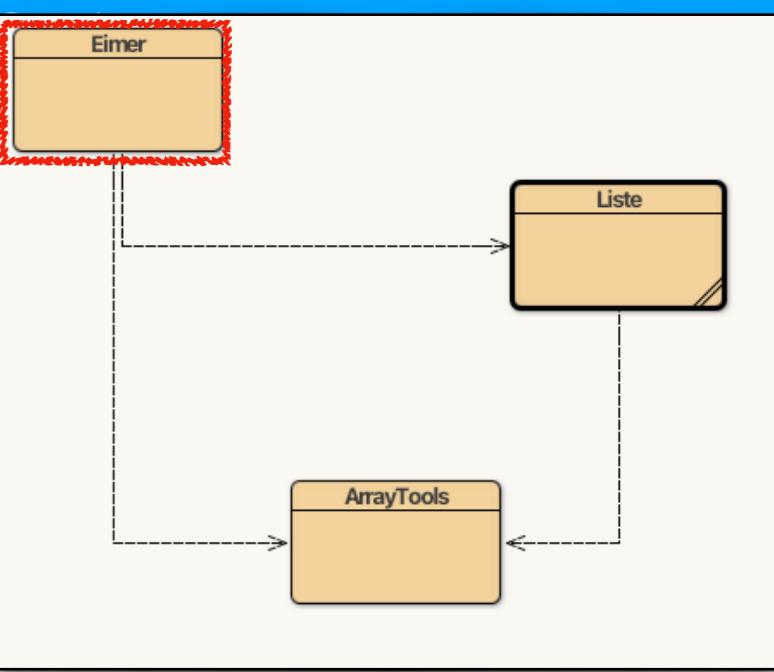
50..59

Einführung in die Objektorientierte Programmierung (OOP)

5.6 Bucket sort

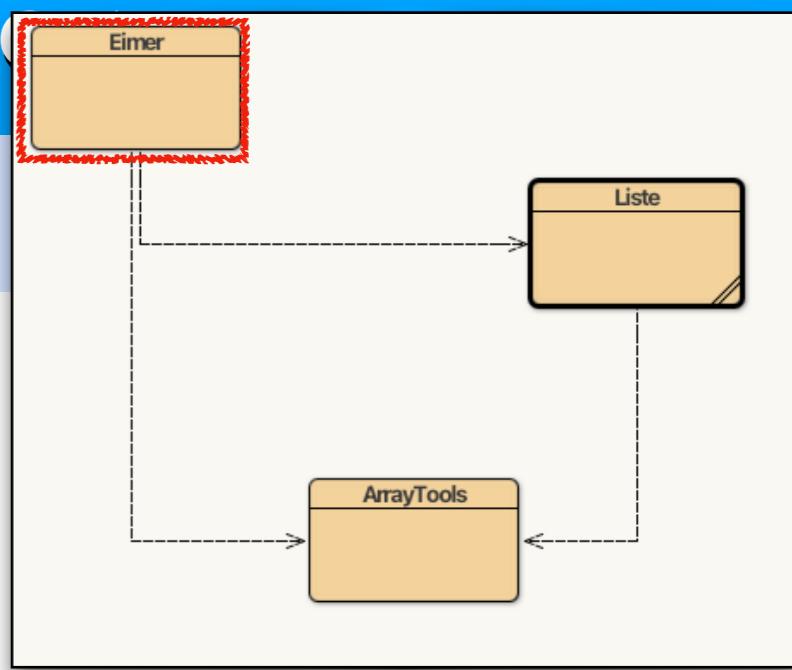


5.6 Bucket sort



```
1 public class Eimer
2 {
3     private final int MAX = 100; ←
4     private final int MAXWERT = 80; ←
5
6     private int[] zahlen; ←
7 }
```

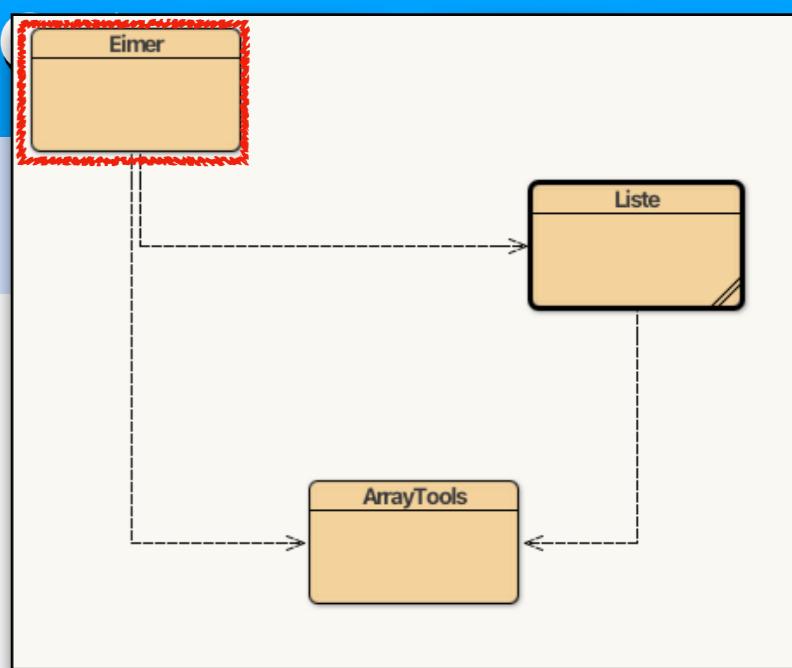
5.6 Bucket sort



```
1 public class Eimer  
2 {  
3     private final int MAX = 100; ← Größe des zu erzeugenden Arrays  
4     private final int MAXWERT = 80; ← größte enthaltene Zahl in diesem Array  
5     private int[] zahlen; ← das zu sortierende Array  
6 }  
7 }
```

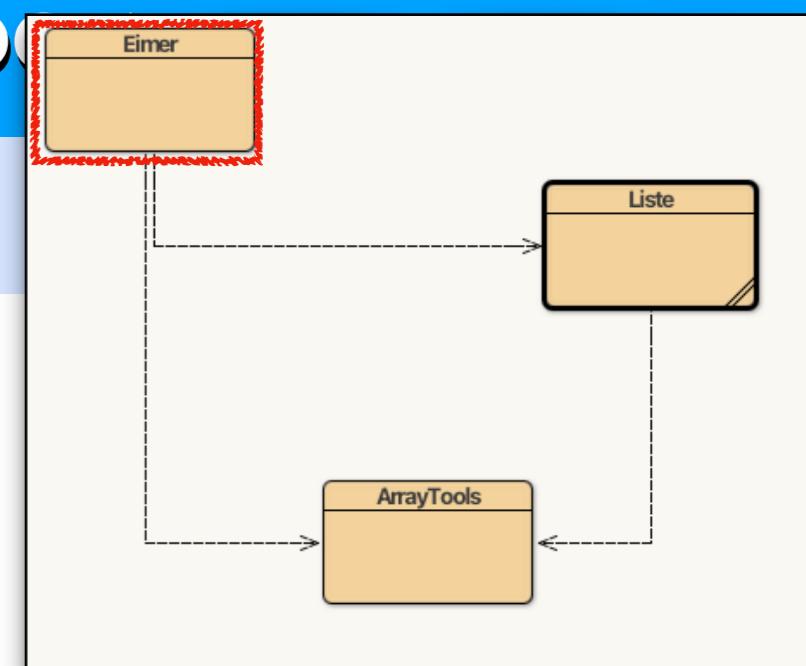
Größe des zu erzeugenden Arrays
größte enthaltene Zahl in diesem Array
das zu sortierende Array

5.6 Bucket sort



```
8 public Eimer()
9 {
10     zahlen = ArrayTools.erzeugeArray(MAX);
11     ArrayTools.fuelleArrayMitZufallszahlen(zahlen, MAX, 1, MAXWERT);
12
13     System.out.println("Anzeigen des unsortierten Arrays");
14     ArrayTools.zeigeArray(zahlen, MAX);
15
16     bucketSortMitListe();
17
18     System.out.println("\nAnzeigen des sortierten Arrays");
19     ArrayTools.zeigeArray(zahlen, MAX);
20 }
```

5.6 Bucket sort



das Array wird erzeugt

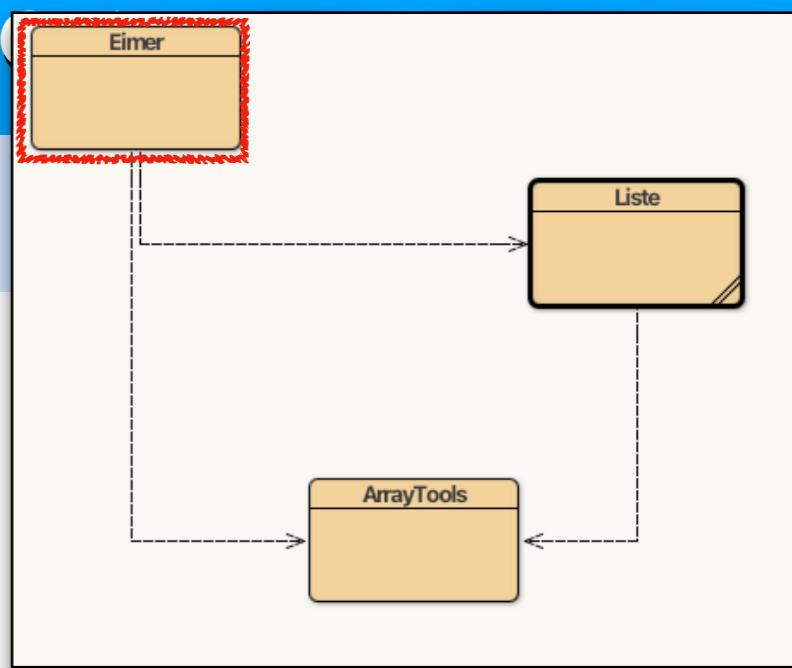
es wird komplett...

...mit Zufallszahlen
im Bereich
[1...MAXWERT]
gefüllt

Hier wird das Array
zur Kontrolle
angezeigt.

```
8 public Eimer()
9 {
10     zahlen = ArrayTools.erzeugeArray(MAX);
11     ArrayTools.fuelleArrayMitZufallszahlen(zahlen, MAX, 1, MAXWERT); ←
12
13     System.out.println("Anzeigen des unsortierten Arrays");
14     ArrayTools.zeigeArray(zahlen, MAX); ←
15
16     bucketSortMitListe();
17
18     System.out.println("\nAnzeigen des sortierten Arrays");
19     ArrayTools.zeigeArray(zahlen, MAX);
20 }
```

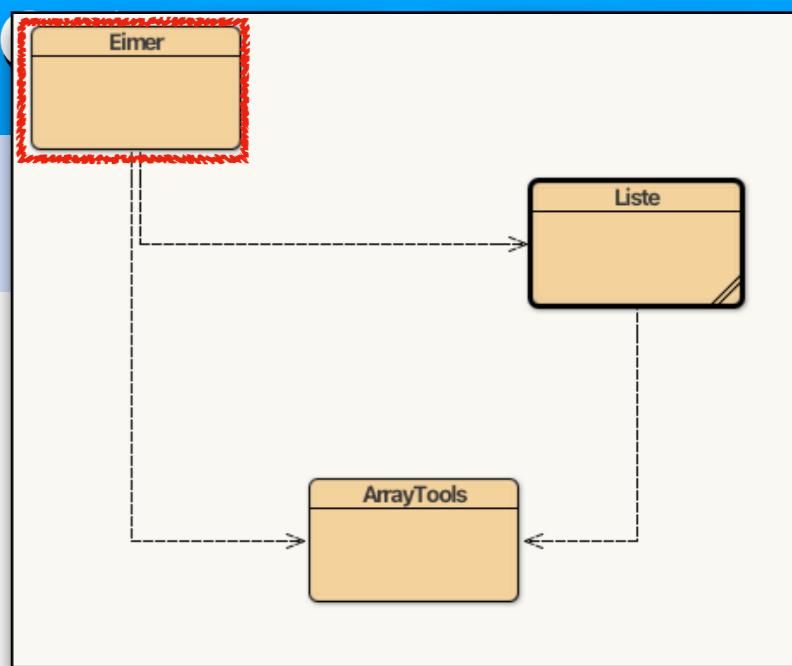
5.6 Bucket sort



```
8 public Eimer()
9 {
10     zahlen = ArrayTools.erzeugeArray(MAX);
11     ArrayTools.fuelleArrayMitZufallszahlen(zahlen, MAX, 1, MAXWERT);
12
13     System.out.println("Anzeigen des unsortierten Arrays");
14     ArrayTools.zeigeArray(zahlen, MAX);
15
16     bucketSortMitListe(); ←
17
18     System.out.println("\nAnzeigen des sortierten Arrays");
19     ArrayTools.zeigeArray(zahlen, MAX); ←
20 }
```

The code snippet is a Java class definition for **Eimer**. It starts with a constructor that initializes an array **zahlen** using **ArrayTools.erzeugeArray** with parameter **MAX**. It then fills the array with random numbers using **ArrayTools.fuelleArrayMitZufallszahlen** with parameters **zahlen**, **MAX**, **1**, and **MAXWERT**. It prints the unsorted array and shows it using **ArrayTools.zeigeArray**. It then calls the **bucketSortMitListe** method. After sorting, it prints the sorted array and shows it again using **ArrayTools.zeigeArray**.

5.6 Bucket sort



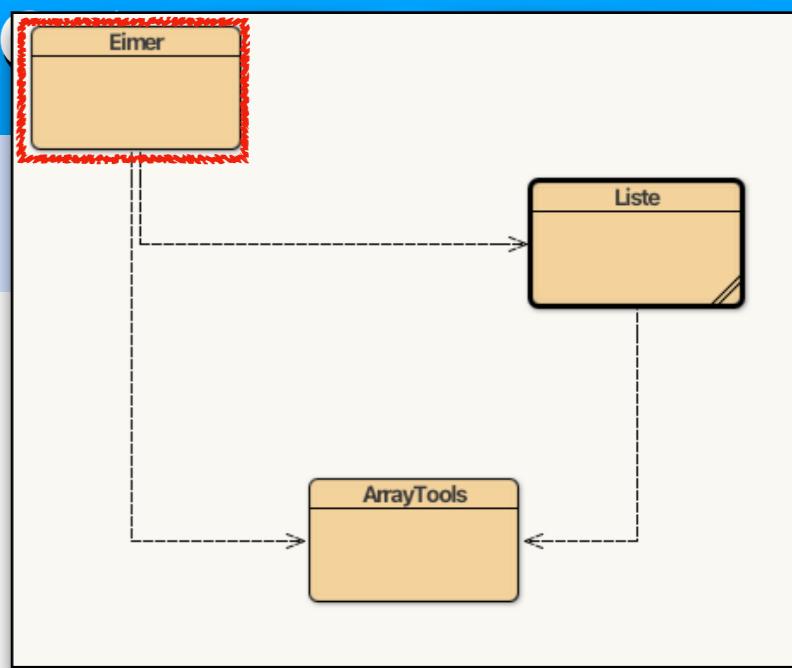
```
8 public Eimer()
9 {
10     zahlen = ArrayTools.erzeugeArray(MAX);
11     ArrayTools.fuelleArrayMitZufallszahlen(zahlen, MAX, 1, MAXWERT);
12
13     System.out.println("Anzeigen des unsortierten Arrays");
14     ArrayTools.zeigeArray(zahlen, MAX);
15
16     bucketSortMitListe(); ← Aufruf der Sortier-Methode!
17
18     System.out.println("\nAnzeigen des sortierten Arrays");
19     ArrayTools.zeigeArray(zahlen, MAX); ← Hier wird das
20 }                                         sortierte Array angezeigt.
```

Aufruf der Sortier-Methode!

Hier wird das sortierte Array angezeigt.

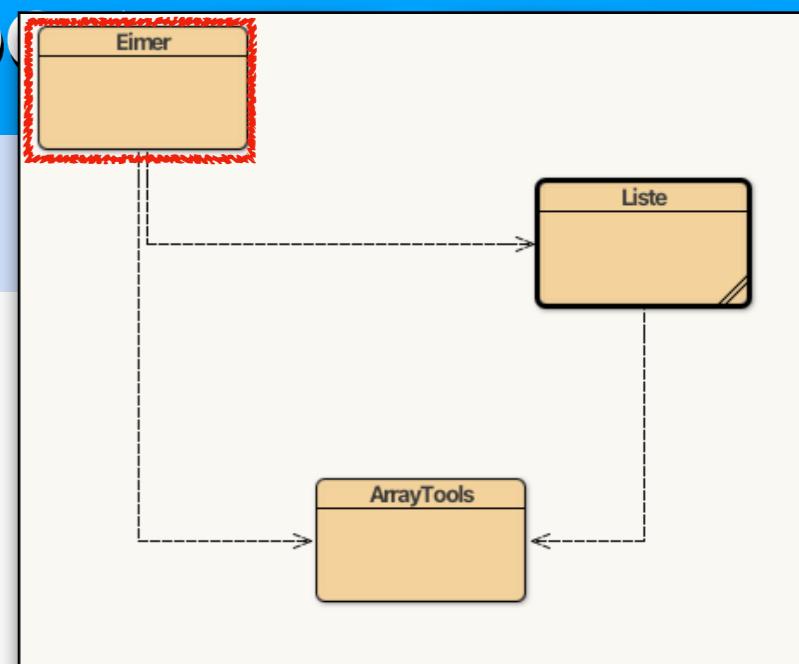
5.6 Bucket sort

```
23 public void bucketSortMitListe()
24 {
25     final int BUCKETS = (MAXWERT / 10) + 1; ←
26
27     Liste[] bucket = new Liste[BUCKETS]; ←
28     for (int i = 0; i < BUCKETS; i++)
29         bucket[i] = new Liste(); ←
30 }
```



5.6 Bucket sort

```
23 public void bucketSortMitListe()
24 {
25     final int BUCKETS = (MAXWERT / 10) + 1; ← Anzahl der Buckets
26
27     Liste[] bucket = new Liste[BUCKETS]; ← festlegen, als lokale
28     for (int i = 0; i < BUCKETS; i++) Konstante!
29         bucket[i] = new Liste(); ← Ein Array von
30 }
```



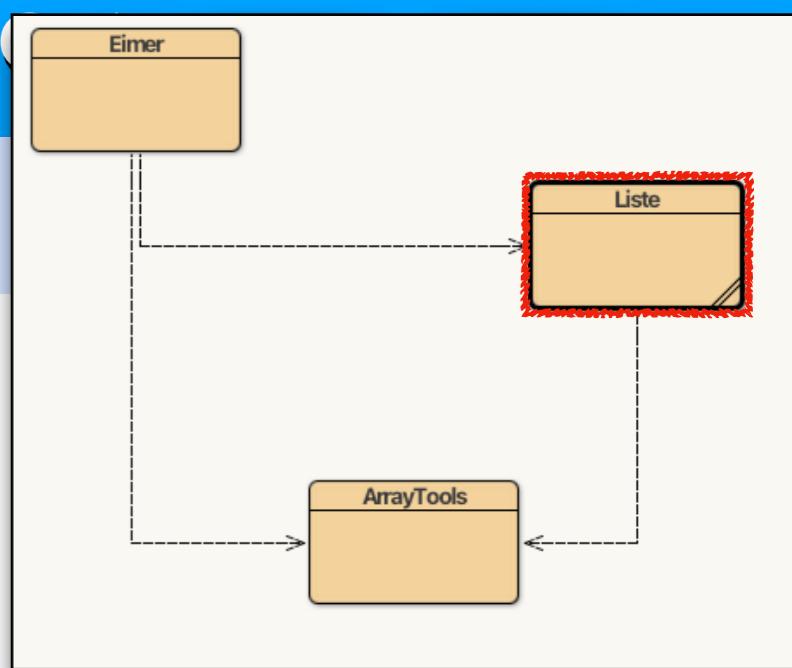
Jeder Bucket ist ein Objekt der Klasse
Liste

Anzahl der Buckets
festlegen, als lokale
Konstante!

Ein Array von
Buckets anlegen

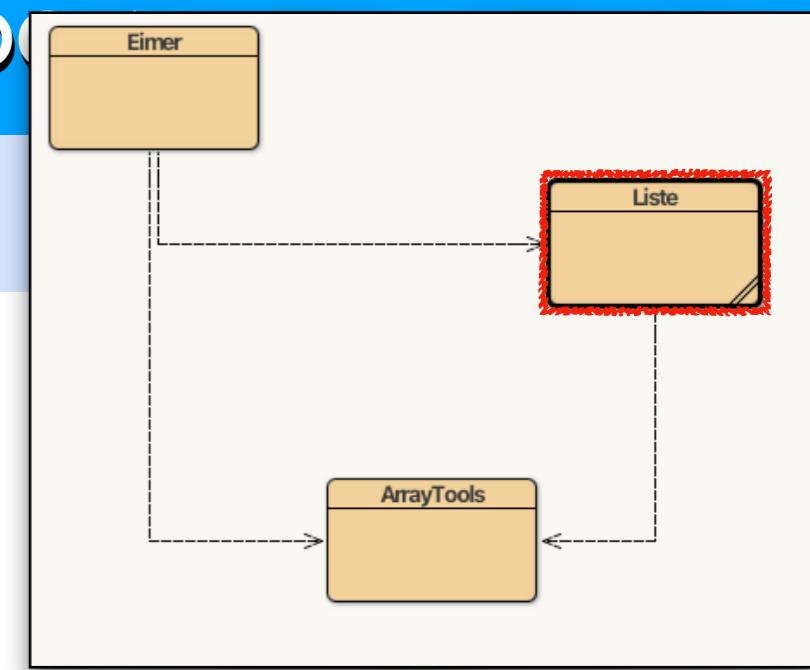
5.6 Bucket sort

```
1 public class Liste
2 {
3     int[] element;
4     int    anzahl;
5
6     public Liste()
7     {
8         element = new int[100];
9         anzahl  = 0;
10    }
11}
```



5.6 Bucket sort

```
1 public class Liste  
2 {  
3     int[] element;  
4     int    anzahl;  
5  
6     public Liste()  
7     {  
8         element = new int[100];  
9         anzahl  = 0;  
10    }  
11}
```



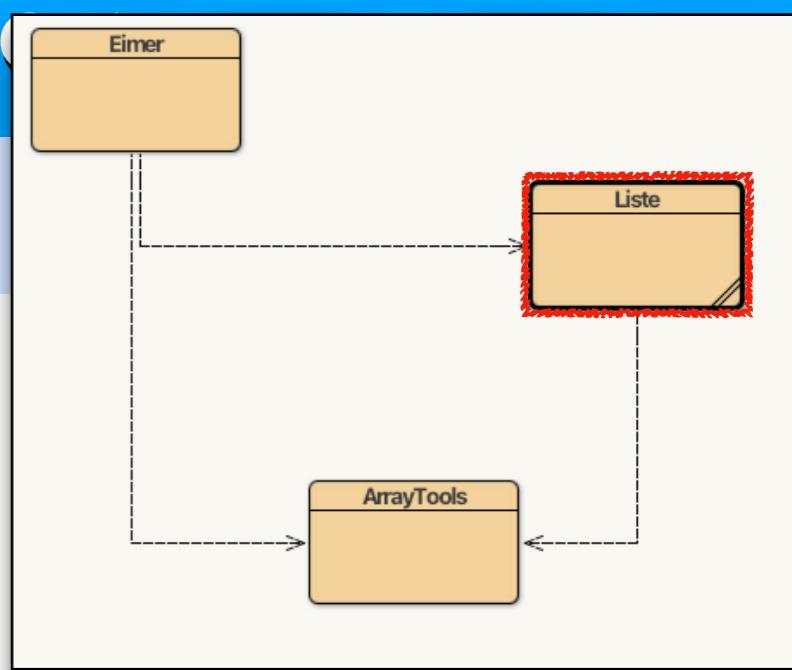
Das interne Array der Klasse Liste

Die Zahl der jeweils enthaltenen Elemente

Die Zahl der jeweils enthaltenen Elemente

5.6 Bucket sort

```
12 public void fuegeEin(int neu)
13 {
14     if (anzahl >= 100) return;
15
16     int pos = 0;
17     while (pos < anzahl && element[pos] < neu)
18         pos++;
19
20     for (int i = anzahl; i > pos; i--)
21         element[i] = element[i - 1];
22
23     element[pos] = neu;
24     anzahl++;
25 }
```



5.6 Bucket sort

```

12 public void fuegeEin(int neu)
13 {
14     if (anzahl >= 100) return;
15
16     int pos = 0;
17     while (pos < anzahl && element[pos] < neu)
18         pos++;
19
20     for (int i = anzahl; i > pos; i--)
21         element[i] = element[i - 1];
22
23     element[pos] = neu;
24     anzahl++;
25 }
```

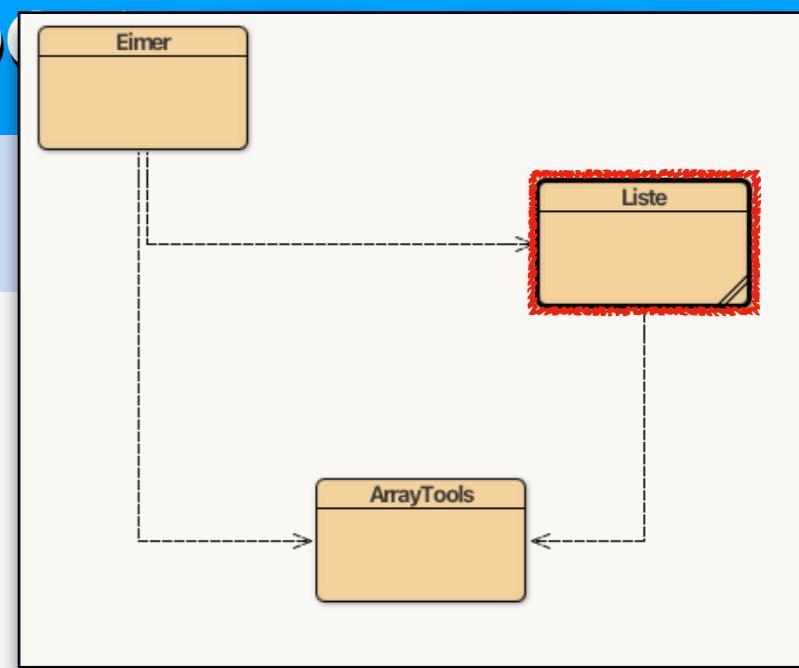
Alle Plätze belegt?

Suche nach der Einfügeposition.

Alle Elemente rechts der Einfügeposition
rücken eine Position nach rechts auf.

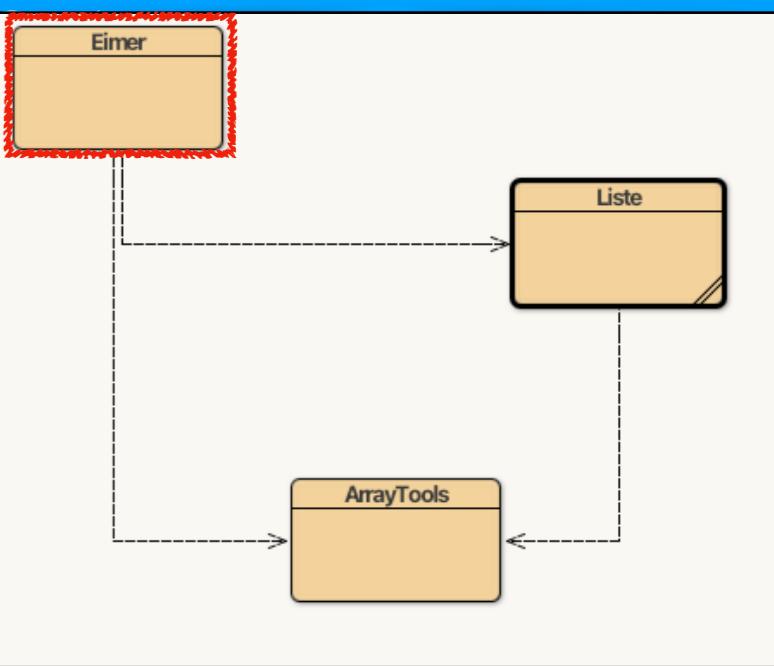
Neues Element einfügen.

Die Anzahl der vorhandenen Elemente wird
aktualisiert.



5.6 Bucket sort

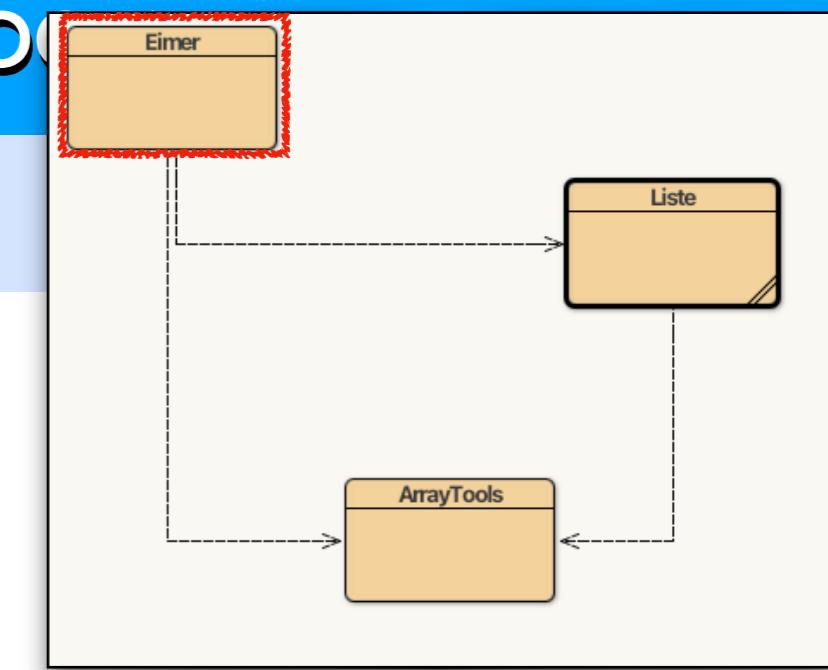
```
31 for (int i = 0; i < MAX; i++)  
32 {  
33     int z = zahlen[i];  
34     int b = z / 10;  
35     bucket[b].fuegeEin(z);  
36 }
```



5.6 Bucket sort

```

31
32     for (int i = 0; i < MAX; i++)
33     {
34         int z = zahlen[i];
35         int b = z / 10;
36         bucket[b].fuegeEin(z);
37     }
  
```



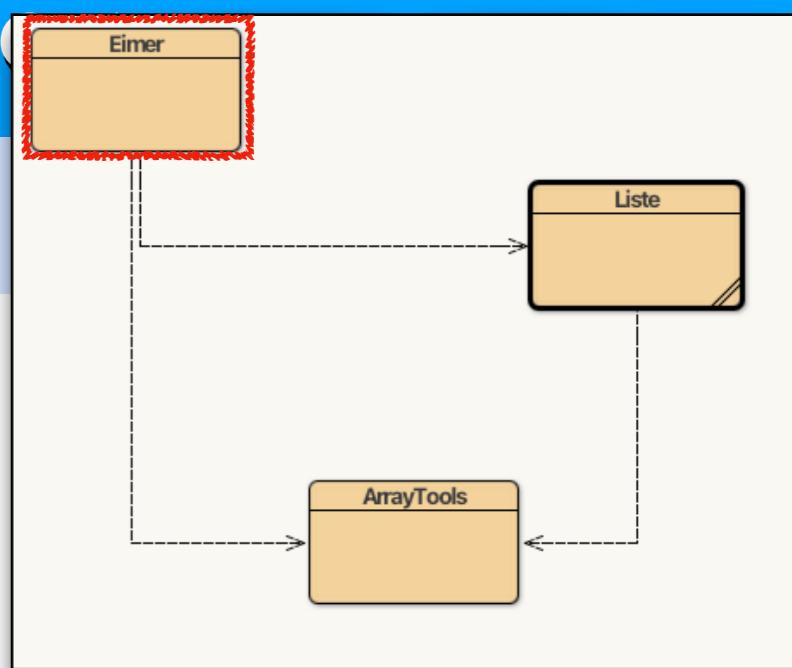
Aktuelles Element.

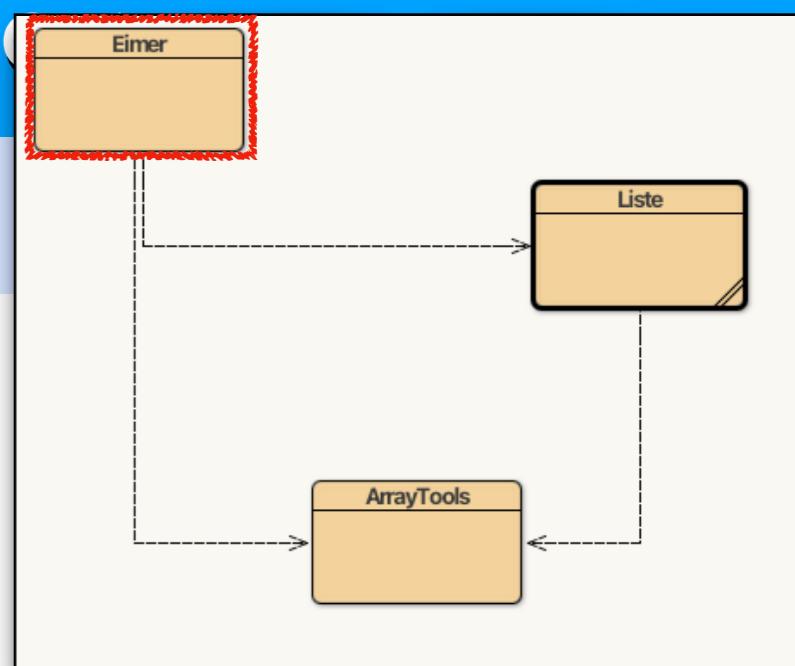
Index des zuständigen Buckets berechnen.

Arrayelement in den passenden Bucket sortiert einfügen.

5.6 Bucket sort

```
53 int index = 0;
54 for (int b = 0; b < BUCKETS; b++)
55 {
56     for (int i = 0; i < bucket[b].anzahl; i++)
57     {
58         zahlen[index] = bucket[b].element[i];
59         index++;
60     }
61 }
```





5.6 Bucket sort

```

53     int index = 0;
54     for (int b = 0; b < BUCKETS; b++)
55     {
56         for (int i = 0; i < bucket[b].anzahl; i++)
57         {
58             zahlen[index] = bucket[b].element[i];
59             index++;
60         }
61     }
  
```

Nächste Position im Hauptarray festlegen.

Extrahierte Zahl in das Hauptarray zurück schreiben.

Alle Buckets durchgehen.

Jede Zahl aus dem Bucket extrahieren.