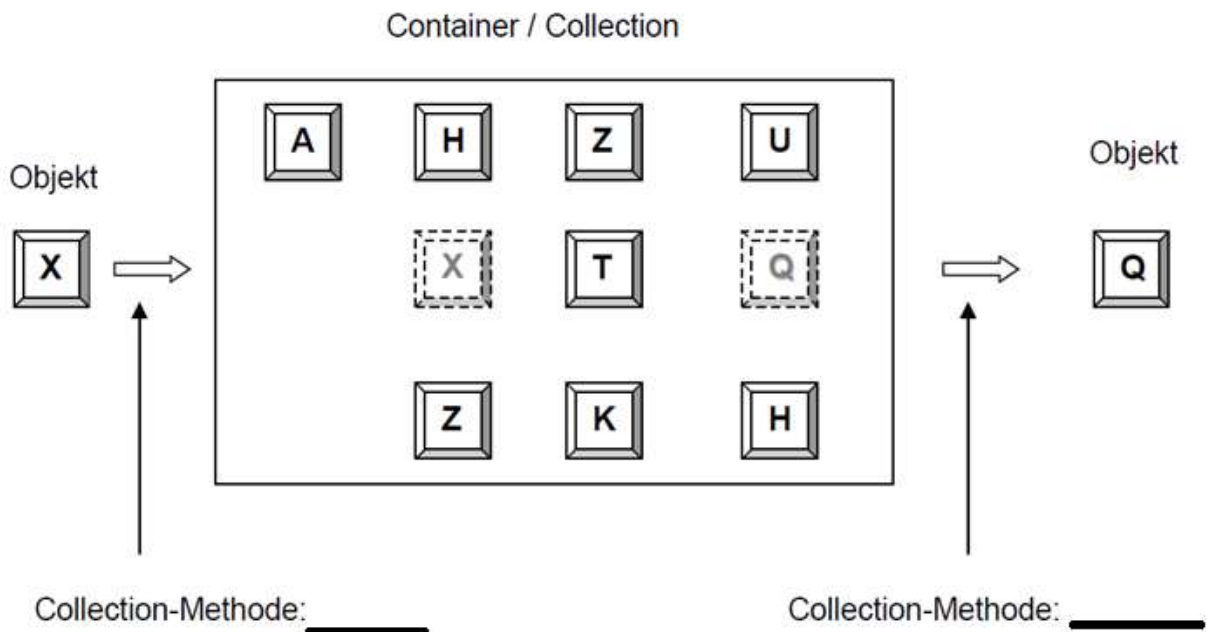


## Java: **Collection Classes**



**Aufgabe:** Ergänzen Sie den Lückentext um folgende die

- *Objekte aufzunehmen und Methoden zur Verfügung stellen* - *selbst wieder ein Objekt* - *Sammlungen oder Behälter* –  
um herauszufinden, wofür die sogenannten Collection Classes gut sind.

### 1. Definition:

Collections sind Container \_\_\_\_\_ bzw. für Objekte. In diese Container lassen sich beliebig viele Objekte hineinlegen, bei Bedarf darauf zugreifen und wieder herausholen. Ein Container bzw. eine Collection ist \_\_\_\_\_.

Die Organisationsform der Objekte im Container ist je nach verwendeter Collection unterschiedlich und entspricht der einer bekannten Datenstruktur.

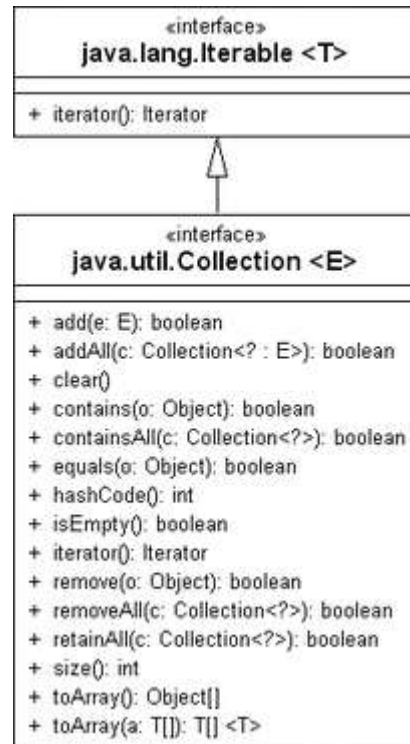
Collections sind somit **Klassen** für **Datenstrukturen**, die dazu dienen

\_\_\_\_\_, wie z.B. das Hinzufügen eines neuen Objektes, das Löschen eines vorhandenen Objektes in der Collection, das Feststellen der Anzahl der in der Collection enthaltenen Objekte, etc.

### Anmerkungen:

- In diese Container werden genaugenommen nicht die Objekte selbst, sondern Referenzen auf diese Objekte gespeichert.
- Die Container sind begrenzt von dem Speicher, den die virtuelle Maschine dem Programm zur Verfügung stellt

## 2. Interface und Klasse der Collection (java.util.Collection)



Grobeinteilung der Operationen im Collection Interface:

- **Basisoperationen (B)** zum Erfragen der Elementanzahl und zum Hinzufügen, Löschen, Selektieren und Finden von Elementen
- **Mengenoperationen (M)**, um etwa andere Sammlungen einzufügen.
- **Feldoperationen (F)** bei Collection, um die Sammlung in ein Array zu konvertieren, und bei Map Operationen, um alternative Ansichten von Schlüsseln oder Werten zu bekommen.

**Aufgabe:** Ordnen Sie die Operationsart (B, M,F) den Methoden im obigen Klassendiagramm zu.

## 3. Das Interface »Iterable«

**Iterable** hat nur die Methode `iterator()`. Ein Iterator verfügt über die Fähigkeit, eine Collection elementweise zu durchlaufen. Der Iterator ermöglicht auch insbesondere die Verwendung einer `foreach`-Schleife zum Durchlaufen der Collection.

## 4. Überblick über konkrete Container Klassen (Auszug aus dem Buch Java ist eine Insel)

Hier können die Container Klassen grob in

- Listen und Mengen
- Assoziativspeicher (Speicher mit Zuordnungen)
- Warteschlangen

eingeteilt werden.

Listen (List)	ArrayList	Implementiert Listen-Funktionalität durch die Abbildung auf ein Feld; implementiert die Schnittstelle <code>List</code> .
	LinkedList	<code>LinkedList</code> ist eine doppelt verkettete Liste, also eine Liste von Einträgen mit einer Referenz auf den jeweiligen Nachfolger und Vorgänger. Das ist nützlich beim Einfügen und Löschen von Elementen an beliebigen Stellen innerhalb der Liste.
Mengen (Set)	HashSet	Eine Implementierung der Schnittstelle <code>Set</code> durch ein schnelles Hash-Verfahren.
	TreeSet	Implementierung von <code>Set</code> durch einen Baum, der alle Elemente sortiert hält.
	LinkedHashSet	Eine schnelle Mengen-Implementierung, die sich parallel auch die Reihenfolge der eingefügten Elemente merkt.
Assoziativspeicher (Map)	HashMap	Implementiert einen assoziativen Speicher durch ein Hash-Verfahren.
	TreeMap	Exemplare dieser Klasse halten ihre Elemente in einem Binärbaum sortiert; implementiert <code>NavigableMap</code> .
	LinkedHashMap	Ein schneller Assoziativspeicher, der sich parallel auch die Reihenfolge der eingefügten Elemente merkt.
	WeakHashMap	Verwaltet Elemente mit schwachen Referenzen, sodass die Laufzeitumgebung bei Speicherknappheit Elemente entfernen kann.
Schlange (Queue)	LinkedList	Die verkettete Liste implementiert <code>Queue</code> und auch <code>Deque</code> .
	ArrayBlockingQueue	Eine blockierende Warteschlange.
	PriorityQueue	Prioritätswarteschlange.

## 5. Beispiel: Die generische List-Collection


```

public static void main(String[] args) {
    //Erzeugen einer Stringlist (ArrayList) mit dem Namen liste
    

    //Hinzufügen von Stringelementen der Liste
    liste.add("Hans");
    liste.add("Emil");
    liste.add("Frauke");
    liste.add("Evelyn");
    liste.add("Richard");
    System.out.println("Ausgabe der Liste");
    for 
    {
        System.out.println(element);
    }
    //Ausgabe des Index von Frauke in der Liste
    System.out.println("Index von Frauke: "  );

    //Einfügen von Hanna vor Frauke
    
}

```

PR1	WA-Java Collection Classes	
-----	----------------------------	---

**Aufgabe:** Ergänzen Sie das obigen Programm so, dass die nachfolgende Ausgabe erscheint:

```
Ausgabe der Liste
Hans
Emil
Frauke
Evelyn
Richard
Index von Frauke: 2
Ausgabe der Liste mit eingefügtem Element Hanna
Hans
Emil
Hanna
Frau
Evelyn
Richard
```