📓 Zusammenfassung: Sammlungen & Iteration (ArrayList, Schleifen, Iteratoren)

◇ 1. Sammlungen `Collections`

Eine Sammlung kann beliebig viele Objekte speichern.

Beispiele: `ArrayList, HashSet, LinkedList, …`

Sehr nützlich in nahezu jedem Java-Programm.

Beispiel:

```
ArrayList<String> namen = new ArrayList<>();
    namen.add("Anna");
    namen.add("Ben");
```

◇ 2. Schleifen – wiederholte Ausführung von Code

Schleifen sind ein fundamentales Mittel, um über Elemente zu gehen oder Code mehrfach auszuführen.

- Arten von Schleifen:
    - `for-each`-Schleife → einfachste Art, über Sammlungen zu iterieren
    - `while`-Schleife → für unbestimmte Iteration
    - `for`-Schleife mit Index → für bestimmte Iteration, wenn man Index braucht

◇ 3. for-each-Schleife

Ideal, wenn man jedes Element einer Sammlung durchgehen möchte.

Beispiel:

```
for (String name : namen) {
    System.out.println(name);
}
```

4. Iterator

Ein Objekt, das Schritt für Schritt durch eine Sammlung läuft.

Beispiel:

```
Iterator<String> it = namen.iterator();
    while (it.hasNext()) {
        String name = it.next();
        System.out.println(name);
}
```

5. Schleifenarten und Iterationstypen

- Bestimmte Iteration

  → Anzahl der Durchläufe ist bekannt.

```
for (int i = 0; i < 5; i++) {
    System.out.println(i);
}
```

- Unbestimmte Iteration

  → Anzahl der Wiederholungen hängt von einer Bedingung ab.

```
while (!liste.isEmpty()) {
    liste.remove(0);
}
```

6. while-Schleife

Wird ausgeführt, solange die Bedingung true ist.

```
int i = 0;
    while (i < 3) {
        System.out.println(i);
        i++;
}
```

7. Indices

Sammlungen wie ArrayList unterstützen den Zugriff über Indices `0-basiert`.

```
String erster = namen.get(0);
```

8. Bibliotheken & Pakete

- Die Java-Standardbibliothek enthält viele nützliche `Klassen`.
- `Klassen` muss man oft mit `import` einbinden.

```
import java.util.ArrayList;
```

9. Anonyme Objekte

Objekte, die ohne Variablenname direkt benutzt werden.

Beispiel:

```
new ArrayList<String>().add("Test");   // wenig gebräuchlich, aber möglich
```

10. null – „kein Objekt“

- `null` bedeutet, dass eine `Objektvariable` auf kein `Objekt` zeigt.
- Zugriff auf `Methoden` von `null` führt zu `NullPointerException`.

```
String name = null;    // kein Objekt
```

Komplettes Beispiel: Sammeln & Iterieren

```
import java.util.ArrayList;

class Beispiel {
public static void main(String[] args) {
ArrayList<Integer> zahlen = new ArrayList<>();

        // Elemente hinzufügen
        zahlen.add(10);
        zahlen.add(20);
        zahlen.add(30);

        // for-each
        for (int z : zahlen) {
            System.out.println("Element: " + z);
        }

        // Iterator
        Iterator<Integer> it = zahlen.iterator();
        while (it.hasNext()) {
            int wert = it.next();
            System.out.println("Iterator: " + wert);
        }
    }
}
```

Dieses Beispiel zeigt:

- Sammlungsobjekt `ArrayList`
- Iteration mit `for-each` & `Iterator`
- bestimmte/unbestimmte `Iterationen`

Merksätze

Sammlungen ermöglichen dynamische Mengen von `Objekten`.

Schleifen = unverzichtbar für jede Art von **Datenverarbeitung**.

Iteratoren bieten eine universelle Art der **Durchlaufsteuerung**.

`null` bedeutet **„kein Objekt“** – Vorsicht beim Zugriff!

`For-each`: einfach und ideal zum Lesen von Sammlungen.