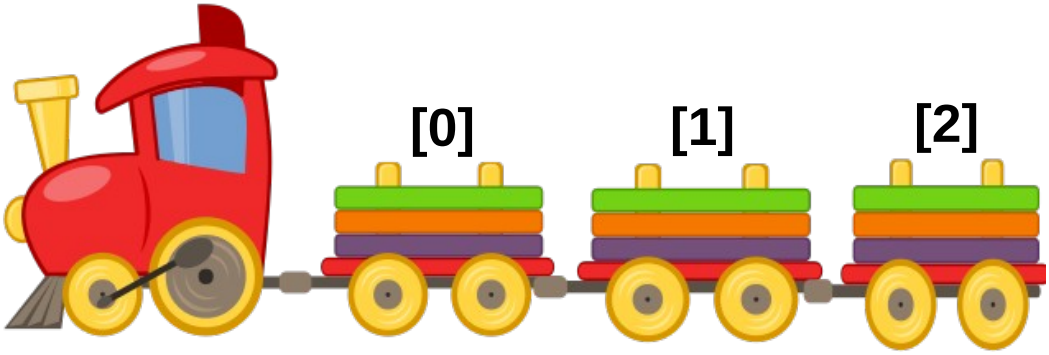
Arrays

Ein ***Array*** stellst du dir am besten vor wie einen langen Zug mit vielen Wagons. Die Wagons sind speziell für eine bestimmte Ware angefertigt.



- Arrays sind ***Aufbewahrungsbehälter*** für gleichartige Objekte oder Attribute.
- Gleichartig bedeutet hierbei, dass ***alle Mitglieder*** dieser Sammlung ***denselben Datentyp*** haben müssen.
- Die „Wagons“ eines Arrays sind nummeriert, beginnend mit 0.

Arrays in JAVA

Deklarieren von Arrays

Man kann aus jedem Datentyp ein Array machen, indem man eckige Klammern dahinter schreibt.

```
int[ ] punktestaende;
String[ ] spielernamen;
RECHTECK[ ] mauern;
```

Erzeugen von Arrays

Ein Array ist ein eigenständiges Objekt, so wie der Zug auch ein eigenständiges Objekt ist und nicht mit der Ware gleichgesetzt werden sollte, die er transportiert.

Man erzeugt Arrays wie jedes andere Objekt mit dem new-Operator. Der Konstruktor eines Arrays hat dann ebenfalls eckige Klammern, in denen man die Anzahl der „Wagons“ angeben muss.

```
punktestaende = new int[10];
spielernamen = new String[10];
mauern = new RECHTECK[15];
```

Die „Wagons“ eines Arrays sind nach dem Erzeugen allerdings noch leer!

Füllen der „Wagons“ eines Arrays

Man kann nun jeden „Wagon“ eines Arrays ansprechen, indem man den Namen des Arrays nennt gefolgt von der Nummer des „Wagons“ in eckigen Klammern.

```
punktestaende[0] = 5;
spielernamen[0] = “Sepp“;
mauern[0] = new RECHTECK();
```

Beachte, dass Referenz-Typen erst als neues Objekt erzeugt werden müssen.

Arbeiten mit Mitgliedern eines Arrays

Man kann die „Wagons“ eines Arrays genauso ansprechen, wenn man mit ihrem Inhalt arbeiten möchte.

```
punktestaende[1] = punktestaende[1] + 10;
return spielernamen[1] + “ hat gewonnen“;
mauern[1].setzeFarbe(“grau“);
```

Übung 01:

Schreibe eine Klasse HIGHSCORE, mit den beiden Array-Attributen *punktestaende* und *spielernamen*.

Initialisiere die beiden Arrays im Konstruktor mit der laenge 10. Setze im Konstruktor bereits die beiden ersten Spielernamen auf „Sepp“ und „Susi“ und die beiden ersten Punktestände auf „12000“ und „7500“.

Schreibe eine Methode
String nenneRang(int platznummer),
die zurück gibt, welcher Spieler mit wie vielen Punkten auf dem entsprechendem Rang im Array gespeichert ist.

Schreibe eine Methode
void setzeRang(int platznummer, String name, int punkte)
die einen neuen Spieler und seine Punkte an einem bestimmten Rang einträgt. Überprüfe die Funktion dieser Methode durch Nutzen der anderen.

Übung 02:

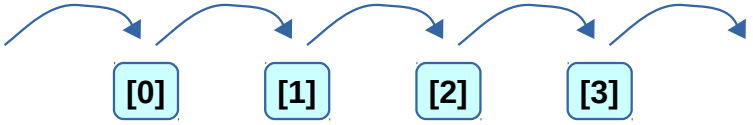
Schreibe eine Klasse MAUERN, die ein RECHTECK-Array der Länge 5 referenziert.

Schreibe einen Konstruktor, der das Array initialisiert und in die ersten beiden „Wagons“ je ein neues Rechteck legt. Setze den Mittelpunkt des ersten Rechtecks auf (100|300) und den des anderen auf (700|300).

Schreibe eine Methode
setzeFarbe(int nummer, String farbe)
welche das Rechteck mit der Nummer *nummer* auf die angegebene Farbe färbt.

Iteration von Arrays

Ein Array zu ***iterieren*** bedeutet, für alle Mitglieder dieser Sammlung eine (ähnliche) Aktion auszuführen.



Das kommt recht häufig vor, z.B. wenn du für alle Mauern deines Mauer-Arrays herausfinden möchtest, ob die Spielfigur gerade eine davon schneitet.

Die Zählschleife

Bei der ***Zähl-Schleife*** handelt es sich um eine Struktur der Algorithmik, die mithilfe einer ***Zähl-Variablen*** einen ***Befehl mehrmals hintereinander ausführen*** kann. Da dabei in der Regel die Zähl-Variable bei jedem Durchlauf um Eins größer oder auch kleiner wird, kann man diese Zähl-Variable nutzen, um von vorne nach hinten oder auch umgekehrt diesen Befehl auf alle Mitglieder eines Arrays anwenden kann.

Zähl-Schleifen in JAVA

Die Zähl-Schleife in JAVA hat eine auf den ersten Blick verwirrende Syntax:

```
for ( int z = startwert ; z <= maximalwert ; z = z + 1 )
{
    // ToDo – hier deine Befehle
}
```

Das Schlüsselwort ist `for` gefolgt von einem ***runden Klammern-Paar*** und einem ***geschweiften Klammern-Paar***.

In den runden Klammern stehen 3 Informationen *(mit Strichpunkt getrennt)*:

Zähl-Variable und Startwert

`int z = 0`	Meine Zähl-Variable heißt z und startet mit dem Wert 0.	*(vorne im Array)*
`int z = 10`	Meine Zählvariable heißt z und startet mit dem Wert 10.	*(hinten im Array)*

Bedingung für das Ausführen der Befehle

Die Befehle in den geschweiften Klammern werden nur ausgeführt, wenn diese Bedingung WAHR ist.

`z <= 5`	„Wagon-Nummern“ von 0 bis 5	*(nur sinnvoll beim Erhöhen der Zähl-Variablen)*
`z >= 0`	„Wagon-Nummern“ von 10 bis 0	*(nur sinnvoll beim Verringern der Zähl-Variablen)*

Wert der Zähl-Variablen ändern

Wenn die Befehle in den geschweiften Klammern einmal ausgeführt wurden, dann muss der Wert der Zähl-Variablen verändert werden.

`z = z + 1`	Wert um 1 erhöhen	*(Array von links nach rechts durchlaufen)*
`z = z – 2`	Wert um 2 verringern	*(von rechts nach links; nur jedes zweite Element)*

Beispiel 1:

```
int[ ] punktestaende;
punktestaende = new int[10];
// Punktestände in das Array speichern

for ( int z=0 ; z<=9 ; z=z+1 )
{
    System.out.println( "Platz " + z );
    System.out.println( punktestaende[z] + " Punkte" );
}
```

Beispiel 2:

```
String[ ] spielernamen;
spielernamen = new String[10];
// Namen in das Array speichern

for ( int z=0 ; z<=9 ; z=z+1 )
{
    System.out.println( "Platz " + z );
    System.out.println( spielernamen[z] );
}
```

Beispiel 3:

```
KREIS[ ] kugeln;
kugeln = new KREIS[7];
// Namen in das Array speichern

for ( int z=0 ; z<=6 ; z=z+1 )
{
    kugeln[z] = new KREIS(40);
    kugeln[z].setzeMittelpunkt( (z+1)*100 , 300 );
    kugeln[z].setzeFarbe("pink");
}
```

Übung 03:

Erweitere Übung 01 um eine Methode, die eine Bestenliste ausgibt. Gib hierzu innerhalb einer Zähl-Schleife erst den Rang, dann den Spielernamen und danach den Punktestand aus.

Übung 04:

Erweitere Übung 02 so, dass du ein Array der Länge 7 hast. Erzeuge im Konstruktor mit Hilfe einer Zähl-Schleife 7 Rechtecke der Breite 20, Höhe 600, Mittelpunkt (50+z*100|300) und Farbe „grau“.