Java – Kurzanleitung

Kapitel 1 Kontrollstrukturen:

1. If-Befehl (Bedingte Anweisung)

Die bedingte Anweisung besteht aus einer Bedingung, deren Wahrheitswert wahr oder falsch sein kann. Ist der Wahrheitswert wahr, wird Anweisungsblock 1 ausgeführt, ist der Wahrheitswert falsch, wird der Anweisungsblock 2 ausgeführt. Die else-Anweisung kann auch weggelassen werden.

2 Fälle:	**3 Fälle:**	**Beispiel für 2 Fälle:**

if (Bedingung) {
  Anweisungsblock1
}
else {
  Anweisungsblock 2
}
	if (Bedingung1) {
	  Anweisungsblock1
	}
	else if (Bedingung2) {
	  Anweisungsblock 2
	}
	else {
	  Anweisungsblock 3
	}
		if (huegelVorhanden („rechts“)){
		  fahre();
		}
		else {
		  drehe („rechts“);
		}

2. switch-Befehl (Verzweigte Anweisung)

Die verzweigte Anweisung führt auf der Grundlage eines Wertes einen bestimmten Anweisungsblock aus. Durch den break-befehl wird jeder Anweisungsblock beendet. Mit dem default-Befehl kann ein Anweisungsblock in allen Fällen ausgeführt werden, die nicht vorher aufgeführt sind.

switch (Ausdruck){
  case Wert1: Anweisungsblock1;
  break;
  case Wert2: Anweisungsblock2;
  break;
  case Wert3: Anweisungsblock3;
  break;
default: AnweisungsblockN
}
	switch (aktuellesFahrzeug){
	  case 0:
	  zweirad = roller;
	  break;
	  case 1:
	  zweirad = harley;
	  break;
	  case 2:
	  zweirad = honda;
	  break;
	  case 3:
	  zweirad = yamaha;
	  break;
	}

3. while-Schleife (Wiederholungsschleife)

Die Wiederholungsschleife wiederholt einen Anweisungsblock solange, bis die vor dem Anweisungsblock angegebene Bedingung nicht mehr erfüllt ist.

while (Bedingung){
  Anweisungsblock
}
	while (huegelVorhanden („rechts“ ()){
	  fahre();
	}

Alternativ kann die Bedingung auch nach der Ausführung des Anweisungsblockes geprüft werden:

do{
  Anweisungsblock
}
while (Bedingung)
	do{
	  fahre();
	}
	while (huegelVorhanden („rechts“))

4. for-Schleife (Zählschleife)

Die Zählschleife wiederholt einen Anweisungsblock so lange, bis eine angegebene Bedingung erstmals falsch wird. In der Zählschleife wird eine Zählvariable deklariert und mit einem Anfangswert definiert. Der Wert der Schleifenvariable wird nach jedem Schleifendurchlauf aktualisiert. Zählschleifen können benutzt werden, um einen Anweisungsblock eine vorher festgelegte Anzahl mal zu wiederholen.

for (Initialisierung; Bedingung; Aktualisierung){
  Anweisung
}
	for (int i = 0; i < 11; i = i + 1){
	  fahre();
	}

Kapitel 2: Datentypen in Java:

1. Primitive Datentypen:

Typ	Speicherplatz	Wertebereich	Beispiel
boolean	1 Bit	True, false	True, false
char	16 Bit	A,..Z,a..z,1..9, . , : , / , …	‘a’, ‘Z’, ’1’
byte	8 Bit	-128 bis 127	1345, -12, 0, 897
short	16 Bit	-32768 bis 32767	7823, -999. 0, 12
int	32 Bit	-2147483648 bis 2147483647	234567, -89999
long	64 Bit	-9223372036854775808 bis 9223372036854775807	98, - 5678456
float		1.4E-45 bis 3.41 E 38	1.23 , -4,987654321
double		4.9E-324 bis 1,79 E 308	3.5678, -4444.5555

Primitive Datentypen müssen vor ihrer Benutzung **deklariert** werden. Dabei wir einem Variablennamen ein Datentyp zugewiesen. Die erste Zuweisung eines Wertes nennt man **Initialisierung**.

Deklaration	Datentyp bezeichner1;	int anzahl;
Initialisierung	bezeichner1 = wert;	Anzahl = 6

Deklaration und Initialisierung können auch in einem Schritt durchgeführt werden: int anzahl = 6;

2. Komplexe Datentypen:

Komplexe Datentypen werden innerhalb einer eigenen Klasse definiert. Die Deklaration erfolgt in der Regel vor dem Konstruktor der Klasse, in der der komplexe Datentyp Deklariert wird.
Die Initialisierung kann innerhalb jeder Methode dieser Klasse erfolgen. Komplexe Datentypen werden **referenziert**, d. h. es wird eine Referenz auf einen bestimmten Speicherbereich (Speicheradresse) festgelegt.

Deklaration	Klassenname bezeichner1;	Fahrzeug roller;
Initialisierung	bezeichner1 = new Klassenname();	roller = new Fahrzeug(20,0.03,100,10);

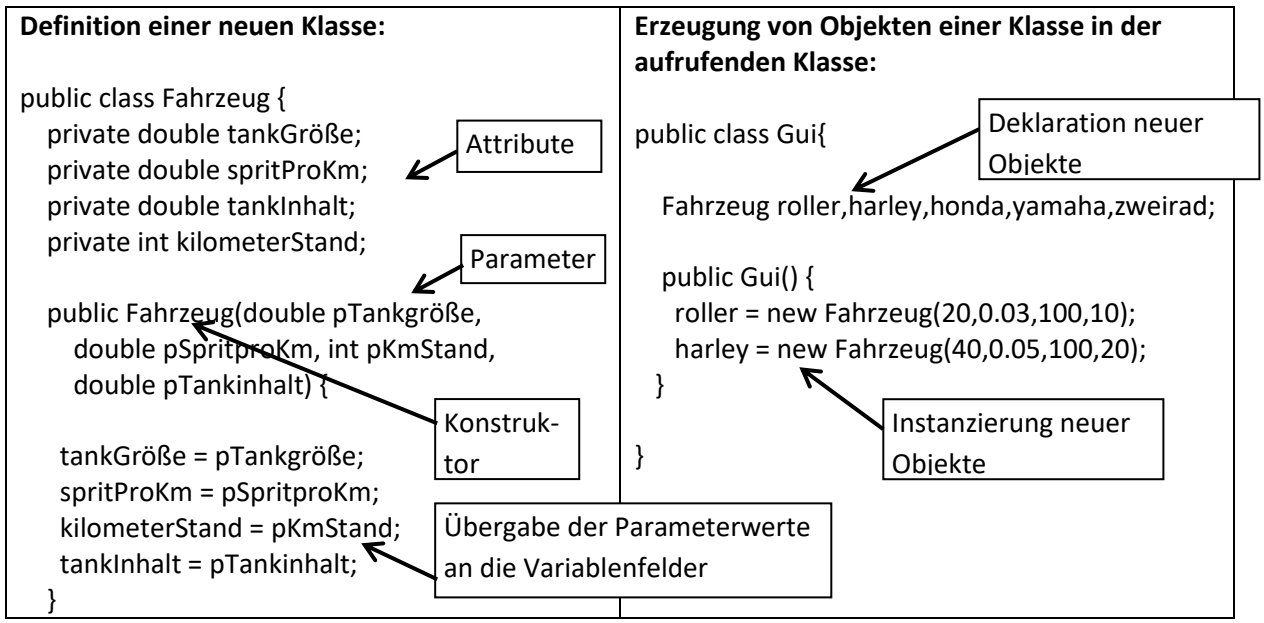
Deklaration und Initialisierung können auch in einem Schritt durchgeführt werden:
Fahrzeug roller = new Fahrzeug(20,0.03,100,10);

Vergleichsoperatoren für primitive Datentypen:

Operator	Beschreibung	Mögliche Datentypen	Beispiel
==	ist gleich	alle	a == true
!=	ist ungleich	alle	b != 5
> bzw. <	ist kleiner bzw. größer	alle außer boolean	d > 6
>= bzw. <=	größer oder gleich	alle außer boolean	e <= „Schmitz“

Vergleichsoperatoren für Komplexe Datentypen:
Auch komplexe Datentypen lassen sich miteinander vergleichen. Es dürfen nur die Operatoren == und != benutzt werden. Gleich sind Objekte nur, wenn Sie auf das gleiche Objekt verweisen. Objekte mit identischen Attributswerten aber unterschiedlichen Identitäten (d. h. verschiedene Instanzen einer Klasse wurden definiert) sind nicht gleich, wenn sie mit == verglichen werden.

Kapitel 3: Klassen in Java



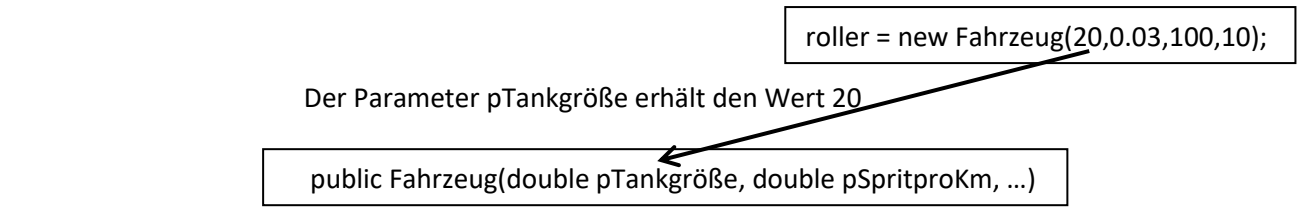
Innerhalb einer **Klasse** können verschiedene **Objekte** verwaltet werden, die alle ähnliche Eigenschaften haben. Die Klasse bietet insbesondere verschiedene **Variablenfelder** an, d. h. Variablen, die für jedes Objekt der Klasse getrennt gespeichert werden (z. B. hat jedes Fahrzeug einen eigenen Kilometerstand, Tankgröße usw.).

Definition einer neuen Klasse:
Die Definition einer neuen Klasse beginnt mit dem Access Modifier **public** und dem Schlüsselwort **class** gefolgt von dem neuen **Klassennamen** (hier Fahrzeug). Danach werden alle **Variablenfelder** deklariert, die für jedes Objekt der Klasse angelegt werden sollen.
Es folgt der Konstruktor der Klasse. Er beginnt mit dem Access Modifier **public** gefolgt von dem Klassennamen. Anschließend wird in Klammern für jede der deklarierten Variablenfelder ein **Parametername** zusammen mit seinem Datentypen angegeben. Der Parametername entspricht in der Regel dem Namen des Variablenfeldes nach dem Buchstaben p (für Parameter). Bei der Erzeugung eines Klassenobjektes wird jeder Parametervariablen der gewünschte Variablenwert übergeben. Im Anweisungsblock wird anschließend in der Regel jedem Variablenfeld der Wert der Parametervariablen übergeben, z. B. tankGröße = pTankgröße.

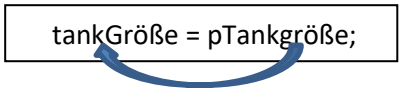
Erzeugung eines neuen Objektes einer Klasse:
Bevor ein Objekt erzeugt wird, muss zunächst der Name des Objektes **deklariert** werden:
Dabei schreibt man den Klassennamen gefolgt von dem neuen Objektnamen, mehrere Objekte der gleichen Klasse können in einem Befehl erzeugt werden:
Fahrzeug roller,harley,honda,yamaha,zweirad;

Anschließend kann mit dem new-Befehl ein Objekt der Klasse erzeugt werden, diesen Vorgang nennt man **Instanzierung**, d. h. es wird eine Instanz der Klasse erzeugt:
roller = new Fahrzeug(20,0.03,100,10);
Nach dem neuen Objektnamen folgen die Schlüsselwörter „=“ und „new“, danach wird der Klassenname angegeben. In Klammern wird für jede Parametervariable ein Wert zugewiesen.

Die Parametervariablen werden somit beim Erzeugen mit den angegeben Werten gefüllt. In der Regel wird im Konstruktor anschließend dem Variablenfeld der Wert aus dem Parametervariablen weitergeleitet.



Der Wert des Parameters pTankgröße wird in das Variablenfeld Tankgröße übertragen:



Get und Setmethoden (Anfragen und Aufträge):

Mit einer **Get-Methode (Anfrage)** kann der Wert eines Variablenfeldes einer Klasse an die aufrufende Klasse ausgelesen werden:

Getmethode in der Klasse Fahrzeug, die das Variablkenfeld kilometerStand verwaltet:	Aufruf der Getmethode in der Klasse Gui, die auf die Klasse Fahrzeug zugreift:

public int getKilometerstand() {
  return kilometerStand;
}
	Roller1.getKilometerStand();

Nach dem Schlüsselwort public folgt der Variablentyp (hier int), indem der Variablenwert zurückgegeben werden soll. Der Variablenwert wird mit dem Befehl return zurückgegeben.

Mit einer **Set-Methode (Auftrag)** kann der Wert eines Variablenfeldes aus der aufrufenden Klasse heraus verändert werden:

Setmethode in der Klasse Fahrzeug, die das Variablenfeld kilometerStand verwaltet:	Aufruf der Setmethode in der Klasse Gui, die auf die Klasse Fahrzeug zugreift:

public void setKilometerstand(int pStrecke) {
  kilometerStand = pStrecke;
}
	Roller1.setKilometerStand(100);

Da die Setmethode keinen Variablenwert zurückgegeben werden, wird nach dem Schlüsselwort public, statt eines Variablentypes das Schlüsselwort **void (leer)** verewendet.

Weitere Methoden (Aufträge):

Methode fahre in der Klasse Fahrzeug, die die Variablenfelder kilometerStand und tankInhalt verwaltet:	Aufruf der Methode fahre in der Klasse Gui, die auf die Klasse Fahrzeug zugreift:

public void fahre(int pStrecke) {
  double spritVerbrauch = spritProKm * pStrecke;
  kilometerStand = kilometerStand + pStrecke;
  tankInhalt = tankInhalt - spritVerbrauch;
}
	roller1.fahre (100);

Weitere Methoden können deklariert werden, die die Werte der Variablenfelder der Klasse verändern können. Die Methoden können Variablenwerte zurückgeben (Der Datentyp wird asngegeben) oder nicht (Schlüsselwort void).

Access Modifier für Variablenfelder:

Der Programmierer möchte, dass nur er auf seine Variablenfelder innerhalb der Klasse zugreifen kann. Der Zugriff von Außen ist dann nur über die Getmethoden möglich. Deshalb wird in der Regel der Modifier **private** verwendet. Zum Vererben (s. Kapitel 4) wird der Modifier **protected** verwendet.

Kapitel 4: Methoden der Klassen String, Math und Umwandlung von Datentypen

Methoden der Klasse String:

String begruessung = „Hallo!“; Die einzelnen Zeichen können über Ihren Index angesprochen werden:

Index	[0]	[1]	[2]	[3]	[4]	[5]
Zeichen	H	a	l	l	o	!

Methode	Beschreibung	Beispiel
int length()	Gibt die Anzahl der Zeichen des Strings zurück	begruessung.length() liefert 6
int indexOf (String str)	Gibt die Position zurück, an der die übergebende Zeichenkette str zum ersten mal auftritt	begruessung.indexOf(„1“) liefert 2
String substring(int beginIndex)	Gibt den Teil des Strings zurück, der bei dem angegebenen Index beginnt	begruessung.substring(2) liefert „llo!“
String substring(int beginIndex, int endIndex)	Gibt den Teil des Strings zurück, der bei dem angegebenen beginIndex beginnt und bei endIndex endet	begruessung.substring(1,3) liefert „all“
char charAt(int index)	Gibt den Buchstaben an der angegebenen Position zurück	begruessung.charAt(4) liefert „o“
boolean equals (Object o)	Liefert true, wenn o ein StringObjekt mit dem gleichen Inahlt ist	begruessung.equals(„Hallo!“) liefert true
int compareTo (String s)	Vergleicht zwei Strings zeichenweise und gibt 0 zurück, wenn sie gleich sind, eine negative Zahl, wenn dieser String s kleiner ist und eine positive Zahl, wenn er größer ist als der übergebende String.	begruessung.compareTo(„Hallo!“) liefert 0 begruessung.compareTo(„Tschüss“) liefert -12 begruessung.compareTo(„Adios“!) liefert 7

Methoden der Klasse Math:

int round (float a)	Gibt die mathematisch gerundete Zahl zurück	Math.round (2.022) liefert 2 Math.round (2.5) liefert 3
double sqrt (diuble a)	Gibt die Quadratwurzel der Zahl a zurück (sqrt = squareroot)	Math.sqrt (81) liefert 9.
double pow (double a, double b	Liefert den Wert der Potenz a^b	Math.pow(2,3) liefert 8
double random()	Erzeugt eine Zufallszahl größer als 0.0 und kleiner als 1.0. Durch Multiplikation und Addition kann der Bereich entsprechend angepasst werden.	Math.random()*5 + 2ß liefert ein x mit 10.0 <= x <15.0

Umwandlung zwischen String, Integer und Float:

String valueOf()	Wandelt einen Integerwert um in einen String	int num = 456 + 5; String.valueOf (num) liefert „461“
String f2.format (float a) wobei f2 ein definiertes Dezimalformat ist	DezimalFormat steuert das Format der Ausgabe von Kommazahlen: „#0.00“ entspricht einer Kommazahl mit 2 Nachkommastellen	float zahl = 2.1 + 0.4; DecimalFormat f2 = new DecimalFormat("#0.00"); f2.format(zahl) liefert „2.50“
Integer.parseInt (String text)	Wandelt einen String in einen Integerwert um	String text = „5“ Integer.parseInt (text) liefert 5

Kapitel 5: Methoden der Swing-Gui-Objekte:

Methoden der Klasse JLabel oder jTextField:

void setText(String text)	Diese Methode setzt den anzuzeigenden Text.
String getText()	Diese Methode liefert den Text des *JLabels* zurück.

Methoden der Klasse JComboBox:

int getSelectedIndex()	Hier wird der Index des in der *JComboBox* ausgewählten Elementes zurückgegeben.
void setSelectedItem(Object anObject)	Hier kann man die Auswahl direkt auf ein bestimmtes Objekt setzen.

Methoden der Klasse JCheckBox:

boolean setSelected()	Bestimmt, ob der Haken der Checkbox gesetzt ist
void isSelected(boolean wahr)	Gibt an, ob der Haken der Checkbox aktuell gesetzt ist.

Methoden der Klasse JTextArea:

void append(String str)	Diese Methode fügt an das Ende des bereits vorhandenen Textes den über den Parameter angegebenen Text hinzu.
void setText(String str)	Diese Methode stellt den über den Parameter angegeben Text dar, mit einem Leerstring "" kann die JTextArea gelöscht werden.

Methoden der Klasse JTable:

void setValueAt (String text1, int zeile, int spalte)	Setzt als Text einer Zelle den angegebenen text1. Die Zählung der Spalte und Zeile beginnt jeweils mit 0.

Kapitel 6: Vererbung von Klassen:

Einige Objekte einer Klasse können spezielle Eigenschaften haben, die nicht alle Objekte der Klasse benötigen. So können einige Motorräder auf der Autobahn fahren, Roller können das nicht. Für alle Zweiräder, die auf der Autobahn fahren können, wird daher die Unterklasse AutobahnFahrzeug definiert. Sie beinhaltet zwei zusätzliche Variablenfelder, die das Fahren auf der Autobahn regeln (autobahn und spritproKmAutobahnZusätzlich). Die Klasse AutobahnFahrzeug erbt alle Variablenfelder und Methoden, die die Klasse Fahrzeug schon besitzt, diese müssen also nicht neu geschrieben werden:

Neue Unterklasse AutobahnFahrzeug:

nach extends wird die Oberklasse benannt

```
public class AutobahnFahrzeug extends Fahrzeug{
   protected boolean autobahn;
   protected double spritproKmAutobahnZusätzlich;

public AutobahnFahrzeug (double pTankgroesse,
      double pSpritproKm,
      int pKmStand,
      double pTankinhalt,
      boolean pAutobahn,
      double pSpritproKmAutobahnZusätzlich) {

  super (pTankgroesse, pSpritproKm,
           pKmStand, pTankinhalt);

  this.autobahn = pAutobahn;
  this.spritproKmAutobahnZusätzlich =
     pSpritproKmAutobahnZusätzlich;
}
```

Oberklasse Fahrzeug , von der die neue Unterklasse Autobahnfahrzeug erbt:

```
public class Fahrzeug {
   protected double tankGröße;
   protected double spritProKm;
   protected double tankInhalt;
   protected int kilometerStand;

   public Fahrzeug(double pTankgröße,
      double pSpritproKm, int pKmStand,
      double pTankinhalt) {

    tankGröße = pTankgröße;
    spritProKm = pSpritproKm;
    kilometerStand = pKmStand;
    tankInhalt = pTankinhalt;
   }
```

Variablen-felder

Konstruk-tor

Übergabe der Parameterw an die Variablenfelder

Aufruf der Unterklasse in der Klasse Gui:

pTankgröße, pSpritproKm usw.

```
harley = new
    AutobahnFahrzeug(40,0.05,100,20,false,0.01);
```

In der Definition der neuen Unterklasse AutobahnFahrzeug wird gefolgt von dem Schlüsselwort extends die bereits bestehende Oberklasse angebeben, hier also die Klasse Fahrzeuge. Nach der Deklaration der Klasse werden die Variablenfelder deklariert, die die Unterklasse zusätzlich haben soll.
Es folgt der Konstruktor der Unterklasse: alle Parameter, mit denen die Unterklasse aufgerufen werden soll, werden mit ihrem Datentyp angegeben. Mit dem Befehl super werden anschließend die Variablenwerte benannt, die an die Parameter der Oberklasse weitergegeben werden sollen, in der Regel verwendet man die Parameternamen selbst. Schließlich werden für die neuen Feldvariablen die Variablenwerte gesetzt, indem die entsprechenden Parameter übergeben werden.

Vererbung: Alle Methoden, die in der Oberklasse implementiert sind, können direkt von der Tochterklasse (Unterklasse) verwendet werden, die Tochterklasse (Unterklasse) erbt dann die Methode der Elternklasse (Oberklasse).
Soll in der Tochterklasse (Unterklasse) eine spezielle Version einer Methode benutzt werden, kann die Methode der Elternklasse (Oberklasse) durch eine Methodendefinition mit gleichem Namen überschrieben werden.

Methode fahre der Tochterklasse Autobahnfahrzeug:

```
public void fahre(int pStrecke) {
  double spritVerbrauch = spritProKm * pStrecke;
   if (autobahn == true) {
     spritVerbrauch = spritVerbrauch +
          spritproKmAutobahnZusätzlich * pStrecke;
   }
   kilometerStand = kilometerStand + pStrecke;
   tankInhalt = tankInhalt - spritVerbrauch;
}
```

Methode fahre der Elternklasse Fahrzeug:

```
public void fahre(int pStrecke) {
  double spritVerbrauch = spritProKm * pStrecke;
  kilometerStand = kilometerStand + pStrecke;
  tankInhalt = tankInhalt - spritVerbrauch;
}
```

Methodenaufruf in der übergeordneten Klasse Gui:

```
Autobahnfahrzeug harley = new Autobahnfahrzeug (…);
Fahrzeug zweirad = harley;
zweirad.fahre (strecke);
```

Im dargestellten Methodenaufruf mit dem Autobahnfahrzeug wird die Methode der Tochterklasse Autobahnfahrzeug aufgerufen, da mit der Variablen zweirad auf ein Autobahnfahrzeug zugegriffen wird.
Eine Methode ist polymorph, wenn sie in verschiedenen Klassen mit der gleichen Signatur (Namen und Parameterliste) definiert wird. Es wird jeweils die Methode der Klasse verwendet, von der das Objekt ein Exemplar ist (**Polymorphie**).

Type casting (Spezialisierung): Will man mit einem Objekt der Elternklasse eine Methode aufrufen, die es nur in der Tochterklasse gibt, kann man das Objekt der Elternklasse in ein Objekt der Tochterklasse umwandeln. Anschließend kann man für dieses Objekt alle Methoden der Tochterklasse verwenden.:

```
Fahrzeug zweirad;
zweirad = harley;  // harley ist ein Objekt der Klasse AutobahnFahrzeug
zweiRadAutobahn = (AutobahnFahrzeug) zweirad;
```

aus dem Objekt der Klasse Fahrzeug wird ein Objekt der Unterklasse AutobahnFahrzeug

Subtyping (Verallgemeinerung): Wird eine Methode nur in einer Oberklasse definiert, kann sie auch von Objekten der Unterklasse aufgerufen werden:

```
zweiradAutobahn.getKilometerStand();   //die Methode getKilometerStand() wurde nur in der Klasse Fahrzeug definiert
```

Kapitel 7: Arrays

In einem Array können viele Daten eines bestimmten Datentyps gespeichert werden. Die größe des Arrays, d. h. die zahl der Daten, die gespeichert werden kann, muss bei der Instanzierung festgelegt werden und kann später nicht mehr verändert werden.

Beipsiel Lottozahlen:

Deklaration: int [] lottozahlen; // in diesem Array können Integer-Zahlen gespeichert werden
Instanzierung: lottozahlen = new int [7]; // das Array wird instanziert, es stehen 7 Plätze von 0 bis 6 zur Verfügung

Lottozahlen [3] = 5; // der Speicherplatz 3 des Arrays wird mit der Zahl 5 belegt