

# Übersicht über die Befehle in JavaTurtle

## Standard

```
// Bsp.java
import ch.aplu.turtle.*;
import java.awt.Color;

public class Skript
{
    Turtle me= new Turtle();

    public Skript()
    {
        me.left(90);
    }
    public static void main(String[] args)
    {
        new Skript();
    }
}
```

Name des Programms ändern: public class, public und new ändern. («Skript»)

## Movement

Turtle zwei = new Turtle(eins);	
j.speed() ;	200 Standard, -1 unendlich
j.hideTurtle() ;      j.showTurtle();	
j.getX();              j.getY();	Abfrage der Turtleposition
j.setPos(x,y);	
j.home() ;	gehe zu (0,0)
j.wrap() ;      Teleport von linker Kante zur rechter Kante wenn dort angekommen (Modus)	
j.fd(0).rt(0).lt(0).back();	Aufreihung von Befehlen
j.sleep();	warten (sek)
j.beep();	ton erzeugen (Frequenz);

## Color

Color x = new Color(0,0,0);	neue Farbe definieren(0-255)
J.setPenColor();	Stiftfarbe ändern
j.setLineWidth();	Standard = 1
j.setFillColor();	
j.clear();	Bild ausfüllen
j.fill();              j.fillOff();	füllt Fläche mit Turtle drinn
j.fill(x,y);	Füllt Fläche mit Koordinate
j.fillToPoint(x,y);	füllt "Turtle" bis Punkt
j.fillToHorizontal(y);	füllt "Turtle" bis Höhe y
j.fillToVertical(x);	füllt "Turtle" bis Höhe y

*Color.black/.blue/.cyan/(.dkgray)/.green/.gray/(.ltgray)/.magenta/.pink/.red/ .white/.yellow*

## Selektion

if (l + 4 == 0)

Tut was bei einer Bedingung («else» nicht unbedingt nötig)

else

Macht etwas, wenn nicht «if» zutrifft

## Schleifen/Variablen

int	ganze Zahlen von $-2^{31}$ bis $2^{31} - 1$
long	ganze Zahlen von $-2^{63}$ bis $2^{63} - 1$
float	Dezimalzahlen (7 Ziffern)
double	Dezimalzahlen mit der doppelten Genauigkeit (16 Ziffern)
boolean	kann nur die Wahrheitswerte true und false annehmen
char	Zeichen (z.B. eine Tastaturtaste)
Turtle	Frage nach einer bestimmten Turtle

«Float» Deklaration // «p = 0.7» Initialisierung // Deklaration und Initialisierung der variable  
Variable darf nicht mit Ziffer beginnen  
\* ; + - / sind nicht erlaubt

Int i = 10;    duoble j = (double) i;

umwandlung int in double (i bleibt 10)  
Umwandeln int in char mit Ascii Tabelle.

```
int i = 0;
while (i < 4)
    i++;      l = l + 10;
```

“while” braucht vorher schon Variable  
“true” gleich unendlich  
beides geht DO-Schleife testet nachher.

```
for (int l = 0; l < 4; i++)
```

typisch für x widerholungen

## Switch

```
Switch(p)
{
    case 0:
        j.left(90);
        break;
    default:
        case 2:
        case 3:
}
```

*Option p wird beim Switch ausgewählt(Hier:0,2,3,Anderes)*

*wird für alle Fälle wiederholt, p = 0*

*Befehl*

*End of «case», ohne wird das nächste auch gemacht*

*wenn kein case eintritt*

*Both do the same*

## Modifizier

private	sichtbar nur in eigener Klasse
public	überall sichtbar
protected	sichtbar im gleichen Package und in Subklassen

# System

## Statusbar

```
j.addStatusBar();      ()=Höhe
t.setStatusText ("Hallo" + p);    Text in « »
t.label("test");           schreibt test auf das Bild
```

## Operatoren

% = Modulo-Operation → Der Rest einer Division  
 == heisst «ist gleich» oder «entspricht»  
 != ungleich  
 || (ctrl+alt+7) oder eines muss richtig sein  
 && beide müssen wahr sein  
 ! falsch  
 ^ unterschiedlich

```
Math.random();           Zufallszahl zwischen 0 und 0.999
((int) (Math.random() * 255))    zufälliges «int»
```

## Diverses

Immer Klammern machen, wenn mehr als 1 Befehl

```
System.out.println(« »);    In=Zeilenumbruch
test= scanner.next();
«public» heisst von aussen aufrufbar
```

Jede Methode wird mit (); am Ende aufgerufen

```
void Test (int bsp)           Definition einer neuen Methode, variablen möglich(bsp)
                               ausserhalb main, innerhalb class
```

## Extension

```
public class RedTurtle extends Turtle    anstelle von public class Bsp
return this;                             Rückgabe von extension an aktuelles Programm.
Extension Befehle benötigen super.(setPenColor)
```

## Rekursionen

Methode muss selber in sich aufgerufen werden. Benötigt Abbruchbedingung

```
private void staircase (int n)           Rekursion
{
if (n == 0)
    return;
```

A \$Cl	Zeichen	A \$Cl	Zeichen	A \$Cl	Zeichen	A \$Cl	Zeichen	A \$Cl	Zeichen	A \$Cl	Zeichen
000	(Null)	046	.	92	\	138	è	184	©	230	μ
001	⊙	047	/	93	]	139	ï	185	⌋	231	þ
002	●	048	0	94	^	140	î	186	⌌	232	ð
003	♥	049	1	95	_	141	ì	187	⌍	233	U
004	♣	050	2	96	`	142	Ā	188	⌎	234	Ů
005	♠	051	3	97	a	143	š	189	⌏	235	U
006	♠	052	4	98	b	144	Ē	190	⌐	236	ý
007	•	053	5	99	c	145	æ	191	⌑	237	Ÿ
008	■	054	6	100	d	146	Æ	192	⌒	238	˘
009	○	055	7	101	e	147	ô	193	⌓	239	˙
010	■	056	8	102	f	148	ö	194	⌔	240	-
011	♂	057	9	103	g	149	ò	195	⌕	241	±
012	♀	058	:	104	h	150	û	196	⌖	242	_
013	♪	059	:	105	i	151	ù	197	⌗	243	¼
014	♫	060	<	106	j	152	ÿ	198	⌘	244	¶
015	☒	061	=	107	k	153	Ō	199	⌙	245	§
016	▶	062	>	108	l	154	Ū	200	⌚	246	÷
017	◀	063	?	109	m	155	ø	201	⌛	247	ˆ
018	↑	064	@	110	n	156	Ɛ	202	⌜	248	˚
019	!!	065	A	111	o	157	∅	203	⌝	249	˘
020	¶	066	B	112	p	158	×	204	⌞	250	˙
021	§	067	C	113	q	159	f	205	⌟	251	˚
022	—	068	D	114	r	160	á	206	⌠	252	˚
023	↑	069	E	115	s	161	í	207	⌡	253	˚
024	↑	070	F	116	t	162	ó	208	⌢	254	■
025	↓	071	G	117	u	163	ú	209	⌣	255	(leer)
026	→	072	H	118	v	164	ñ	210	⌤		
027	←	073	I	119	w	165	Ñ	211	⌥		
028	L	074	J	120	x	166	ª	212	⌦		
029	↔	075	K	121	y	167	º	213	⌧		
030	▲	076	L	122	z	168	¿	214	⌨		
031	▼	077	M	123	{	169	®	215	〈		
032	(Leerstelle)	078	N	124		170	™	216	〉		
033	!	079	O	125	}	171	½	217	⌫		
034	"	080	P	126	~	172	¾	218	⌬		
035	#	081	Q	127	△	173	ı	219	⌭		
036	\$	82	R	128	◊	174	«	220	⌮		
037	%	83	S	129	ü	175	»	221	⌯		
038	&	84	T	130	é	176	⌌	222	⌰		
039		85	U	131	â	177	⌍	223	⌱		
040	(	86	V	132	ä	178	⌎	224	⌲		
041	)	87	W	133	å	179	⌏	225	⌳		
042	^	88	X	134	ä	180	⌐	226	⌴		
043	+	89	Y	135	ç	181	⌑	227	⌵		
044	,	90	Z	136	ê	182	⌒	228	⌶		
045	"."	91	[	137	ë	183	⌓				