Dokumentationen zu

„von Stiften und Mäusen“

in Java

Inhalt

Die Klasse *Bildschirm*

Ein Bildschirm ist das Modell des angeschlossenen Computerbildschirms. Es entspricht einem Fenster auf dem Computerbildschirm. Auf ihm kann mit Stiften gezeichnet werden. Zu diesem Zweck ist die Zeichenebene auf dem Bildschirm mit einem Koordinatensystem versehen, dessen Ursprung sich in der oberen linken Ecke der Zeichenebene befindet und dessen Achsen horizontal nach rechts und vertikal nach unten gerichtet sind. Die Einheit ist ein *Pixel (picture element)*.
Hinweis: Wenn der Bildschirm von einem anderen Fenster überdeckt wird, so wird der überdeckte Bildschirminhalt gelöscht und leider nicht wieder automatisch aktualisiert.

Bildschirm
- Breite - Höhe - Ort - Hintergrundfarbe
+ ! init() + ? breite() + ! gibFrei() + ? hoehe() + ? holeGanzeZahl() + ? holeText() + ? holeZahl() + ! loescheAlles() + ! nachVorn() + ! setzeFarbe()

Jede Klasse besitzt (mindestens) einen Konstruktor, mit dessen Hilfe ein Objekt dieser Klasse erzeugt wird. Ein Konstruktor ist ein besonderer öffentlicher (public) Dienst, der den gleichen Namen besitzt wie die Klasse. Daran wird dieser Dienst auch als Konstruktor erkannt. Beachte: im Quelltext fehlt beim Konstruktor das Wort *void*.
Der Konstruktor ist praktisch der Initialisierungsteil eines Klassenobjektes. Deshalb wird er im Klassendiagramm auch oft als *init()* bezeichnet.

Konstruktoren:

Bildschirm()
Der Bildschirm ist mit seiner Zeichenebene mit maximaler Größe initialisiert.

Bildschirm(int pBreite, int pHoehe)
Der Bildschirm ist mit seiner Zeichenebene initialisiert. Der obere, linke Eckpunkt besitzt die Computer-Monitor-Koordinaten (0; 0). Die beiden Parameter bestimmen Breite und Höhe.

Bildschirm(int pLinks, int pOben, int pBreite, int pHoehe)
Der Bildschirm ist mit seiner Zeichenebene initialisiert. Die Parameter pLinks und pOben sind Koordinaten des Computer-Monitors; sie legen den oberen, linken Eckpunkt des Bildschirmfensters fest. Die beiden anderen Koordinaten bestimmen die Breite und Höhe des Bildschirmfensters.

Methoden:

int **breite**()
liefert die Breite der Zeichenebene

void **gibFrei**()
Nachdem eine Taste oder der Mausknopf gedrückt wurde, steht der Bild schirm nicht mehr zur Verfügung, d.h. die Zeichenebene verschwindet. (Es erscheint die Meldung *„Programm beendet“*)

int **hoehe**()
liefert die Hoehe der Zeichenebene.

int **holeGanzeZahl**()
liefert eine ganze Zahl, die mit einem Eingabedialog gelesen wird. Falls die Eingabe keine ganze Zahl ist, wird 0 zurückgegeben.

int **holeGanzeZahl**(String pMeldung)
liefert eine ganze Zahl, die mit einem Eingabedialog gelesen wird. Falls die

Eingabe keine ganze Zahl ist, wird 0 zurückgegeben. pMeldung wird als Eingabeaufforderung angezeigt.

String **holeText**()
liefert einen Text, der mit einem Eingabedialog gelesen wird.

String **holeText**(String pMeldung)
liefert einen Text, der mit einem Eingabedialog gelesen wird. pMeldung wird als Eingabeaufforderung angezeigt.

double **holeZahl**()
liefert eine (Komma-) Zahl, die mit einem Eingabedialog gelesen wird. Falls die Eingabe keine Zahl ist, wird 0.0 zurückgegeben.

double **holeZahl**(String pMeldung)
liefert eine (Komma-) Zahl, die mit einem Eingabedialog gelesen wird. Falls die Eingabe keine Zahl ist, wird 0.0 zurückgegeben. pMeldung wird als Eingabeaufforderung angezeigt.

void **loescheAlles**()
Die Zeichenebene ist danach leer.

void **nachVorn**()
macht das Bildschirmfenster zum vordersten Fenster

void **setzeFarbe**(int pFarbe)
aendert die Hintergrundfarbe der Zeichenebene. Alte Zeichnungen auf dem Bildschirm werden gelöscht.

```
0 = schwarz, 1 = blau, 2 = türkis, 3 = grau,
4 = hellgrau, 5 = hellgrün, 6 = hellgrau,
7 = violett, 8 = gelb, 9 = rosa, 10 = rot, 11 = weiß,
12 = hellgelb
```

Die Klasse *Stift*

Stift
Position # Richtung # Zustand # Modus
+ ! init() + ! bewegeBis() + ! bewegeUm() + ! dreheBis() + ! dreheUm() + ! dreheZu() + ! gibFrei() + ! hoch() + ? hPosition() + ? istUnten() + ! normal() + ! radiere() + ! runter() + ! schreibeZahl() + ! schreibeText() + ? vPosition() + ! wechsle() + ? winkel() + ! zeichneKreis() + ! zeichneRechteck()

Der Stift ist ein Werkzeug, das sich auf dem Bildschirm bewegen kann. Er befindet sich stets auf einer genau definierten Position des Bildschirms, die durch Zeichenkoordinaten (horizontal nach rechts, vertikal nach unten) angegeben wird, und zeigt in eine Richtung, die durch Winkel beschrieben wird (0° entspricht der Richtung nach rechts, Drehsinn ist mathematisch positiv).

Der Stift kennt zwei Zustände: Ist der Stift abgesenkt (runter) und bewegt er sich über den Bildschirm, so hinterlässt er eine Spur, die von einem Zeichenmodus abhängig ist. Ist der Stift angehoben (hoch), hinterlässt er keine Spur.

Beim Zeichnen kennt der Stift drei Modi:

- Normal: der Stift zeichnet eine Linie in der Stiftfarbe;
- Wechseln: der Stift zeichnet eine Linie, wobei die Untergrundfarbe in die Stiftfarbe und die Stiftfarbe in die Untergrundfarbe geändert wird;
- Radieren: der Stift zeichnet eine Linie in der Farbe des Untergrunds.

Konstruktor:

Stift()
Der Stift wird initialisiert. Die Zeichenebene steht zur Verfügung und der Stift befindet sich angehoben oben links an Position (0,0) mit Richtung 0° im normalen Zeichenmodus.

Methoden:

void **bewegeBis**(double pH, double pV)
Der Stift wird unabhängig von seiner vorherigen Position auf die durch die Parameter angegebene Position bewegt. Die Winkelstellung des Stiftes hat sich nicht geändert.

void **bewegeUm**(double pDistanz)
Der Stift wird von seiner aktuellen Position in die aktuelle Richtung bewegt. Der Parameter gibt die Länge der zurückgelegten Strecke an.

void **dreheBis**(double pWinkel)
Der Stift wird unabhängig von seiner vorherigen Richtung auf die durch pWinkel angegebene Winkelgröße gedreht.

void **dreheUm**(double pWinkel)
Der Stift wird ausgehend von seiner jetzigen Richtung um die durch pWinkel angegebene Winkelgröße im mathematisch positiven Sinne weitergedreht.

void **dreheZu**(double pWohinH, double pWohinV)
Der Stift wird unabhängig von seiner vorherigen Richtung in die Richtung des Punktes gedreht, dessen Koordinaten übergeben werden.

void **gibFrei**()
Der Stift wird freigegeben.

void **hoch**()
Der Stift wird angehoben.

double **hPosition**()
liefert die horizontale Koordinate der aktuellen Stiftposition.

boolean **istUnten**()
liefert *true*, wenn der Stift abgesenkt ist, ansonsten *false*.

void **normal**()
Der Stift arbeitet danach im Normalmodus.

void **radiere**()
Der Stift arbeitet danach im Radiermodus.

void **runter**()
Der Stift wird abgesenkt.

Bemerkung: bei allen folgenden Schreibmethoden wird oberhalb der aktuellen Stiftposition geschrieben. Falls der Stift also z.B. die Position (0, 0) hat, wird man nichts sehen!

void **schreibeText**(char pZeichen)
Der Stift schreibt das angegebenen Zeichen auf die Zeichenebene unter Verwendung seines aktuellen Zeichenmodus unabhängig vom Zustand. Die aktuelle Stiftposition war die linke, untere Ecke des Zeichens. Die neue Stiftposition ist die rechte, untere Ecke des Zeichens.

void **schreibeText**(String pText)
Der Stift schreibt den angegebenen Text auf die Zeichenebene unter Verwendung seines aktuellen Zeichenmodus unabhängig vom Zustand. Die aktuelle Stiftposition war die linke, untere Ecke des Textes. Die neue Stiftposition ist die rechte, untere Ecke des Textes.

void **schreibeZahl**(double pZahl)
Der Stift schreibt die angegebene Zahl auf die Zeichenebene unter Verwendung seines aktuellen Zeichenmodus unabhängig vom Zustand. Die aktuelle Stiftposition war die linke, untere Ecke der Zahl. Die neue Stiftposition ist die rechte, untere Ecke der Zahl.

double **vPosition**()
liefert die vertikale Koordinate der aktuellen Stiftposition.

void **wechsle**()
Der Stift arbeitet danach im Wechselmodus.

double **winkel**()
liefert die aktuelle Bewegungsrichtung (in Grad) des Stifts.

void **zeichneKreis**(double pRadius)
Der Stift zeichnet unabhängig von seinem Zustand im aktuellen Zeichenmodus einen Kreis mit der aktuellen Position als Mittelpunkt und dem angegebenen Radius. Die Position und die Richtung des Stiftes sind unverändert.

void **zeichneRechteck**(double pBreite, double pHoehe)
Der Stift zeichnet unabhängig von seinem Zustand im aktuellen Zeichenmodus ein achsenparalleles Rechteck mit der aktuellen Position als linker, oberer Ecke und der angegebenen Breite und Höhe. Die Position und die Richtung des Stiftes sind unverändert.

Die Klasse *Maus*

Maus
- Position - Zustand
+ ! init() + ? doppelKlick() + ! gibFrei() + ? hPosition() + ? istGedrueckt() + ? vPosition()

Eine Maus realisiert die Mauseingabe des verwendeten Computers. Diese ist gekennzeichnet durch die aktuelle Position der Maus auf dem Bildschirm des Computers und die Betätigung der Maustaste zu einem bestimmten Zeitpunkt.

Konstruktor:

Maus()
Die Maus wird initialisiert.

Methoden:

boolean **doppelKlick**()
Prüft, ob ein Doppelklick stattgefunden hat.

void **gibFrei**()
Die Maus steht nicht mehr zur Verfügung

int **hPosition**()
liefert die gegenwärtige horizontale Koordinate der Position der Maus auf dem Bildschirm, unabhängig davon, ob die Maus gedrueckt wurde.

boolean **istGedrueckt**()
Prüft, ob eine Maustaste im Moment gedrückt ist.

int **vPosition**()
liefert die gegenwärtige vertikale Koordinate der Position der Maus auf dem Bildschirm, unabhängig davon, ob die Maus gedrückt wurde.

Die Klasse *Tastatur*

Tastatur
- Zeichen
+ ! init() + ! gibFrei() + ! weiter() + ? wurdeGedrueckt() + ? zeichen()

Eine Tastatur realisiert die Tastatureingabe des verwendeten Computers. Sie speichert die eingegebenen Tastaturzeichen in der Reihenfolge ihrer Eingabe in einem eigenen Pufferspeicher.
Es gibt eine Klasse namens ***Zeichen***, welche folgende Konstanten als Tastatureingabe zur Verfügung stellt: ESCAPE, ENDE, POS1, PFEILLINKS, PFEILRECHTS, PFEILOBEN, PFEILUNTEN, BILDUNTEN, BILDAUF, TAB, EINGABE, BACKSPACE, DELETE, F1, F2, ... , F12
Der Aufruf einer solchen Konstanten erfolgt z.B. durch
Zeichen.PFEILLINKS;

Konstruktor:

Tastatur()
Die Tastatur wird initialisiert und der Tastaturpuffer enthält keine Zeichen.

Methoden:

void **gibFrei**()
Die Tastatur steht nicht mehr zur Verfügung.

void **weiter**()
Das vorderste Zeichen im Tastaturpuffer wird entfernt. Falls der Tastaturpuffer vorher nicht mit "wurdeGedrueckt()" getestet wurde, erfolgt eine Fehlermeldung.

boolean **wurdeGedrueckt**()
Falls eine Taste gedrückt wurde, der Tastaturpuffer also (mindestens) ein Zeichen enthält, ist "wurdeGedrueckt" wahr, sonst falsch.
Hinweis: der Zustand *(wurdeGedrueckt() = TRUE)* ändert sich erst wieder, wenn der Auftrag *weiter()* (entsprechend oft) erteilt wird.

char **zeichen**()
Diese Anfrage liefert eine Kopie des zuerst eingegebenen Zeichens im Tastaturpuffer. Das Zeichen im Tastaturpuffer selbst wird dabei nicht gelöscht. Falls der Tastaturpuffer vorher nicht mit "wurdeGedrueckt()" gestestet wurde, erfolgt eine Fehlermeldung.

Die Klasse *Buntstift*

Oberklasse: *Stift*

Der Buntstift übernimmt die Attribute der Klasse Stift. Allerdings besitzt er darüber hinausgehende Attribute, die mithilfe von geeigneten Methoden einzeln gesetzt werden können.

Es gibt eine Klasse namens ***Farbe***, welche einige Standardfarben zur Verfügung stellt. Beispiel: *meinBuntstift.setzeFarbe(Farbe.ROT);*

Es gibt eine Klasse namens ***Muster***, welche einige Standardfüllmuster zur Verfügung stellt. Beispiel:
meinBuntstift.setzeFuellmuster(Muster.DURCHSCHEINEND);

Es gibt eine Klasse namens ***Schrift***, welche einige Standardschriftstile zur Verfügung stellt. Beispiel: *meinBuntstift.setzeSchriftstil(Schrift.FETT);*

Farbe:	SCHWARZ, WEISS, ROT, GRUEN, BLAU, GELB, etc.
Schriftart:	STANDARDSCHRIFTART, ARIAL, HELVETICA, TIMESROMAN,
Schriftstil:	STANDARDSTIL, FETT, KURSIV, KURSIVFETT,
Schriftgroesse:	10 bzw. andere positive ganze Zahl, STANDARDGROESSE
Linienbreite:	1 bzw. andere positive ganze Zahl.
Füllmuster:	DURCHSICHTIG, DURCHSCHEINEND, GEFUELLT

Die Standardeinstellungen sind hier jeweils zuerst genannt (z.T. allerdings von der aktuellen Hardwareplattform abhängig).

Konstruktor:

Buntstift()
Der Buntstift wird als Stift initialisiert und mit den Standardeinstellungen versehen.

Methoden:

int **linienbreite**()
int **linienBreite**()
Der Buntstift liefert seine Linienbreite.

void **setzeFarbe**(Color pFarbe)
void **setzeFarbe**(int pFarbe)
Die angegebene Farbe wird die aktuelle Farbe des Buntstifts.

void **setzeFuellmuster**(int pMuster)
void **setzeFuellMuster**(int pMuster)
Das angegebene Muster ist das aktuelle Muster des Buntstifts fuer Rechtecke und Kreise.

void **setzeLinienbreite**(int pBreite)
void **setzeLinienBreite**(int pBreite)
Die angegebene Breite wird die aktuelle Linienbreite des Buntstifts.

void **setzeSchriftart**(String pArt)
void **setzeSchriftArt**(String pArt)
Die angegebene Schriftart wird die aktuelle Schriftart des Buntstifts.

void **setzeSchriftgroesse**(int pGroesse)
void **setzeSchriftGroesse**(int pGroesse)
Die angegebene Schriftgroesse wird die aktuelle Schriftgröße des Buntstifts.

void **setzeSchriftstil**(int pStil)
void **setzeSchriftStil**(int pStil)
Der angegebene Schriftstil wird der aktuelle Schriftstil des Buntstifts.

int **textbreite**(String pText)
int **textBreite**(String pText)
ermittelt die Breite des Textes unter Berücksichtigung der Eigenschaften des Buntstifts.

int **zahlbreite**(double pZahl)
int **zahlBreite**(double pZahl)
int **zahlBreite**(int pZahl)
ermittelt die Breite der Zahl unter Berücksichtigung der Eigenschaften des Buntstifts.

int **zeichenbreite**(char pZeichen)
int **zeichenBreite**(char pZeichen)
ermittelt die Breite des Zeichens unter Berücksichtigung der Eigenschaften des Buntstifts.

Die Klasse *Rechner*

Das Package ***sum.werkzeuge*** enthält eine Klasse namens ***Rechner***. Hier werden nur wenige ihrer Dienste vorgestellt.

Konstruktor:

Rechner()
Der Rechner wird initialisiert.

Methoden:

int **ganzeZufallszahl**()
int **ganzeZufallsZahl**()

int **ganzeZufallszahl**(int pVon, int pBis)
int **ganzeZufallsZahl**(int pVon, int pBis)
Es wird eine ganze Zufallszahl zwischen pVon und pBis (einschließlich der Grenzen) zurückgegeben.

int **gerundet**(double pZahl)

double **quadrat**(double pZahl)
int **quadrat**(int pZahl)

double **cos**(double pWinkel)
double **sin**(double pWinkel)
double **tan**(double pWinkel)

double **wurzel**(double pZahl)

double **zufallszahl**()
double **zufallsZahl**()
Es wird eine Zufallszahl zwischen 0 und 1 zurückgegeben.