Dokumentation der *Klassenbibliothek „Shapes and Sprites“*

Version 5.8 (15.12.2021)

I. Package sas

Das Package wird vollständig mit `import sas.*` geladen.

1. Die Klasse View

Ein Objekt der Klasse **View** ist ein Fenster auf dem Bildschirm, das die Zeichenfläche enthält, auf der die zweidimensionalen Shapes-Objekte abgebildet werden. Jedes Shapes-and-Sprites-Programm kann genau eine Instanz dieser Klasse (Singleton) erzeugen. Dieses Objekt ist allen Shapes-Objekten bekannt. Bei dem Versuch eine weitere Instanz der View-Klasse zu erzeugen, wird eine Runtime-Exception geworfen. Die Größe der sichtbaren Zeichenfläche beträgt nach dem erzeugen eines View-Objektes mit dem Konstruktor ohne Parameter 600 Pixel in der Breite und 400 Pixel in der Höhe. Der Name des Programms, der in der Titelzeile des Fensters angezeigt wird lautet „SaS-Programm“. Sowohl die Größe der Zeichenfläche, als auch der Name des Programms können durch Konstruktoren mit entsprechenden Parametern gesetzt oder durch Methodenaufrufe verändert werden. Das Objekt der Klasse `View` überwacht außerdem die Tastaturereignisse, die durch Methoden abgefragt werden können.

Konstruktoren

`View()`

Erzeugt ein Java-Programmfenster mit einer Zeichenfläche der Größe 600 x 400 Pixel. Der Name des Programms, der in der Titelzeile des Fensters angezeigt wird, lautet „SaS-Programm“.

`View(String name)`

Erzeugt ein Java-Programmfenster mit einer Zeichenfläche der Größe 600 x 400 Pixel. Der als Parameter übergebene Name des Programms wird in der Titelzeile des Programmfensters angezeigt.

`View(int width, int height)`

Erzeugt ein Java-Programmfenster mit einer Zeichenfläche der Größe `width` x `height` Pixel. Der Name des Programms, der in der Titelzeile des Fensters angezeigt wird, lautet „SaS-Programm“.

`View(int width, int height, String name)`

Erzeugt ein Java-Programmfenster mit einer Zeichenfläche der Größe `width` x `height` Pixel. Der als Parameter übergebene Name des Programms wird in der Titelzeile des Fensters angezeigt.

Methoden

`int getWidth()`
Liefert die Breite der sichtbaren Zeichenfläche.

`int getHeight()`
Liefert die Höhe der sichtbaren Zeichenfläche.

`void setSize(int width, int height)`
Setzt die Breite und die Höhe der sichtbaren Zeichenfläche neu.

`void setName(String name)`
Setzt den Namen des Programms, der in der Titelzeile des Bildschirmfensters angezeigt wird, auf den übergebenen Wert.

`void setBackgroundColor(Color color)`
Setzt die Hintergrundfarbe der Zeichenfläche auf den übergebenen Wert.

`void remove(Shapes grafik)`
Löscht das angegebene Shapes-Objekt von der Zeichenfläche. Es wird nicht aus dem Speicher gelöscht.

`void wait(int millsec)`
Bei Aufruf dieser Methode wird das Programm *millSec* Millisekunden angehalten.

`boolean keyBackspacePressed()`
Liefert true, wenn die Backspace-Taste gedrückt wurde.

`void keyBufferDelete()`
Alle Zeichen im Tastaturpuffer werden gelöscht.

`boolean keyDownPressed()`
Liefert true, wenn die Abpfeil-Taste gedrückt wurde.

`boolean keyEnterPressed()`
Liefert true, wenn die Enter-Taste gedrückt wurde.

`char keyGetChar()`
Liefert das erste Zeichen aus dem Tastaturpuffer. Wartet, bis eine Taste gedrückt wird.

`boolean keyPressed()`
Liefert true, wenn eine beliebige Taste gedrückt wurde.

`boolean keyPressed(char c)`

Liefert true, wenn die Taste mit dem angegebenen Zeichen gedrückt wurde. Bei Großbuchstaben muss gleichzeitig die Umschalttaste gedrückt sein.

`boolean keyLeftPressed()`

Liefert true, wenn die Linkspfeil-Taste gedrückt wurde.

`boolean keyRightPressed()`

Liefert true, wenn die rechtspfeil-Taste gedrückt wurde.

`boolean keyUpPressed()`

Liefert true, wenn die Aufpfeil-Taste gedrückt wurde.

2. Klassen der Shapes-Objekte

Mit den in diesem Kapitel dokumentierten Klassen lassen sich zweidimensionale Grafik-Objekte - im Folgenden **Shapes-Objekte** genannt - erzeugen und auf der Zeichenfläche des View-Objektes anzeigen. Da alle Klassen Unterklassen der abstrakten Klasse `Shapes` sind, lassen sich alle Transformationen (verschieben, drehen, skalieren, spiegeln) auf all diese Objekte anwenden. Da die Klasse `Shapes` die Zugriffsklasse `public` hat, `können` Shapes-Objekte beliebigen Typs in einem Array vom Typ `Shapes[]` gespeichert werden (Polymorphie).

2.1 Die Klasse Circle

Ein Objekt der Klasse `Circle` ist ein Kreis auf der Zeichenfläche. Attribute sind die momentane Position, die Breite und die Höhe des Kreises, bezogen auf das minimale Rechteck, das den Kreis umgibt, sowie das Merkmal der Sichtbarkeit. Darüber hinaus besitzt ein Objekt der Klasse `Circle` eine Füllfarbe oder eine Textur, mit der es ausgefüllt ist. Das Attribut Richtung hat zunächst keinen Einfluss auf die Darstellung des Kreises. Es besteht aber die Möglichkeit, den Kreis in diese Richtung zu bewegen.

Konstruktoren

`Circle(double xp, double yp, double radius)`

Erzeugt einen schwarz ausgefüllten Kreis an der angegebenen Position mit dem angegebenen Radius. Die Koordinaten beziehen sich auf die linke obere Ecke des minimalen Rechtecks, das den Kreis umgibt. Die Richtung, bezogen auf die Nordrichtung im Uhrzeigersinn, erhält den Wert 90^0.

`Circle(double xp, double yp, double radius, Color color)`

Erzeugt einen Kreis an der angegebenen Position mit dem angegebenen Radius, ausgefüllt mit der angegebenen Farbe. Die Koordinaten beziehen sich auf die linke obere Ecke des minimalen Rechtecks, das den Kreis umgibt. Die Richtung, bezogen auf die Nordrichtung im Uhrzeigersinn, erhält den Wert 90^0.

`Circle(double xp, double yp, double radius, String textur)`

Erzeugt einen Kreis mit dem angegebenen Radius, ausgefüllt mit der Textur, deren Dateiname als Parameter übergeben wird. Kann die Datei nicht geladen werden, wird ein blau ausgefüllter Kreis angezeigt . Die Koordinaten der Position beziehen sich auf die linke obere Ecke des Rechtecks, das den Kreis umgibt. Die Richtung, bezogen auf die Nordrichtung im Uhrzeigersinn, erhält den Wert 900.

Methoden

`Circle clone()`

Liefert eine Kopie des die Methode aufrufenden Circle-Objekts.

2.2 Die Klasse Ellipse

Ein Objekt der Klasse `Ellipse` ist eine Ellipse auf der Zeichenfläche. Attribute sind die momentane Position, die Breite und die Höhe der Ellipse, bezogen auf das minimale Rechteck, das die Ellipse umgibt, sowie das Merkmal der Sichtbarkeit. Darüber hinaus besitzt ein Objekt der Klasse Ellipse eine Füllfarbe oder eine Textur, mit der es ausgefüllt
ist. Das Attribut Richtung hat zunächst keinen Einfluss auf die Darstellung der Ellipse. Es besteht aber die Möglichkeit, die Ellipse in diese Richtung zu bewegen.

Konstruktoren

`Ellipse(double xp, double yp, double width, double height)`

Erzeugt eine schwarz ausgefüllte Ellipse mit der angegebenen Breite und Höhe an der angegebenen Position. Die Koordinaten beziehen sich auf die linke obere Ecke des Rechtecks, das die Ellipse umgibt. Die Richtung erhält den Wert 90^0.

`Ellipse(double xp, double yp, double width, double height, Color color)`

Erzeugt eine Ellipse mit der angegebenen Breite und Höhe, ausgefüllt mit der angegebenen Farbe, an der angegebenen Position. Die Koordinaten beziehen sich auf die linke obere Ecke des Rechtecks, das die Ellipse umgibt. Die Richtung erhält den Wert 90^0.

`Ellipse(double xp, double yp, double width, double height, String textur)`

Erzeugt eine Ellipse mit der angegebenen Höhe und Breite, ausgefüllt mit der Textur, deren Dateiname als Parameter übergeben wird. Kann die Datei nicht geladen werden, wird die Ellipse blau gefüllt. Die Koordinaten der Position der Ellipse beziehen sich auf die linke obere Ecke des Rechtecks, das sie umgibt.

Methoden

`Ellipse clone()`

Liefert eine Kopie des die Methode aufrufenden Ellipse-Objekts.

2.3 Die Klasse Picture

Ein Objekt der Klasse `Picture` ist ein digitales Bild im Format „jpg“ oder „png“, das aus einer Datei geladen und auf der Zeichenfläche abgebildet wird. Attribute sind die momentane Position (linke obere Ecke), die Breite und die Höhe der Grafik sowie das Merkmal der Sichtbarkeit. Das Attribut Richtung hat zunächst keinen Einfluss auf die Darstellung der Grafik. Es besteht aber die Möglichkeit, die Grafik in diese Richtung zu drehen oder zu bewegen. Die Richtung erhält den Wert 90^0.

Konstruktoren

`Picture(double xp, double yp, String name)`

Lädt eine Bilddatei (jpg oder png) mit dem angegebenen Namen und zeigt das Bild in Originalgröße im Ausgabefenster an der Position `xp`, `yp` (linke obere Ecke) an. Die Richtung erhält den Wert 90^0.

`Picture(double xp, double yp, double width, double height, String name)`
Lädt eine Bilddatei (jpg oder png) mit dem angegebenen Namen und zeigt das Bild, skaliert auf die angegebene Breite und Höhe, im Ausgabefenster an der Position `xp`, `yp` (linke obere Ecke) an. Die Richtung erhält den Wert 90^0.

Methoden

`Picture clone()`
Liefert eine Kopie des die Methode aufrufenden Picture-Objekts.

2.4 Die Klasse Polygon

Ein Objekt der Klasse Polygon ist ein Vieleck mit beliebiger Eckenzahl. Attribute sind die momentane Position (linke obere Ecke des umgebenden Rechtecks), die Breite und die Höhe des Polygons sowie das Merkmal der Sichtbarkeit. Das Attribut Richtung hat zunächst keinen Einfluss auf die Darstellung des Polygons. Es besteht aber die Möglichkeit, das Polygon in diese Richtung zu bewegen.

Konstruktoren

`Polygon(double xp, double yp)`
Erzeugt ein schwarz ausgefülltes Polygon mit dem ersten Eckpunkt an der angegebenen Position. Das Polygon ist erst sichtbar, wenn mindestens eine weitere Ecke mit dem Auftrag `add` hinzugefügt wurde. Ein Polygon mit zwei „Ecken“ ist eine Strecke zwischen den beiden Punkten. Die Richtung, bezogen auf die Nordrichtung im Uhrzeigersinn, erhält den Wert 90^0.

`Polygon(double xp, double yp, Color color)`
Erzeugt ein mit der angegebenen Farbe ausgefülltes Polygon mit dem ersten Eckpunkt an der angegebenen Position. Das Polygon ist erst sichtbar, wenn mindestens eine weitere Ecke mit dem Auftrag `add` hinzugefügt wurde. Ein Polygon mit zwei „Ecken“ ist eine Strecke zwischen den beiden Punkten. Die Richtung, bezogen auf die Nordrichtung im Uhrzeigersinn, erhält den Wert 90^0.

`Polygon(double xp, double yp, String textur)`
Erzeugt ein Polygon mit dem ersten Eckpunkt an der angegebenen Position. Es wird ausgefüllt mit der Textur, deren Dateiname als Parameter übergeben wird. Kann die Datei nicht geladen werden, wird das Polygon blau gefüllt. Das Polygon ist erst sichtbar, wenn mindestens eine weitere Ecke mit dem Auftrag `add` hinzugefügt wurden. Ein Polygon mit zwei „Ecken“ ist eine Strecke zwischen den beiden Punkten. Die Richtung, bezogen auf die Nordrichtung im Uhrzeigersinn, erhält den Wert 90^0.

Methoden

`void add(xp, yp)`

Fügt dem Polygon einen weiteren Eckpunkt mit den Koordinaten relativ zum ersten Punkt hinzu und zeichnet das Polygon.

`Polygon clone()`

Liefert eine Kopie des die Methode aufrufenden Polygon-Objekts.

2.5 Die Klasse Rectangle

Ein Objekt der Klasse `Rectangle` ist ein Rechteck auf der Zeichenfläche. Attribute sind die momentane Position, die Breite und die Höhe des Rechtecks sowie das Merkmal der Sichtbarkeit. Darüber hinaus besitzt ein Objekt der Klasse `Rectangle` eine Füllfarbe oder eine Textur, mit der es ausgefüllt ist. Das Attribut Richtung hat zunächst keinen Einfluss auf die Darstellung des Rechtecks. Es besteht die Möglichkeit, das Rechteck in diese Richtung zu bewegen.

Konstruktoren

`Rectangle(double xp, double yp, double width, double height)`

Erzeugt ein schwarzes Rechteck mit der angegebenen Breite und Höhe an der angegebenen Position. Die Koordinaten beziehen sich auf die linke obere Ecke des Rechtecks. Die Richtung erhält den Wert 90^0.

`Rectangle(double xp, double yp, double width, double height, Color color)`

Erzeugt ein Rechteck mit der angegebenen Breite und Höhe, ausgefüllt mit der angegebenen Farbe, an der angegebenen Position. Die Koordinaten beziehen sich auf die linke obere Ecke des Rechtecks. Die Richtung erhält den Wert 90^0.

`Rectangle(double xp, double yp, double width, double height, String textur)`

Erzeugt ein Rechteck mit der angegebenen Höhe und Breite, ausgefüllt mit der Textur, deren Dateiname als Parameter übergeben wird. Kann die Datei nicht geladen werden, wird das Rechteck blau gefüllt. Die Koordinaten beziehen sich auf die linke obere Ecke des Rechtecks. Die Richtung erhält den Wert 90^0.

Methoden

`Rectangle clone()`

Liefert eine Kopie des die Methode aufrufenden Rectangle-Objekts.

2.6 Die Klasse Sprite

Objekte der Klasse `Sprite` verbinden Grafik-Objekte zu einem neuen Grafik-Objekt, auf das alle Transformationen angewendet werden können. Attribute sind die momentane Position, die Breite und die Höhe des Sprites (des umgebenden minimalen Rechtecks) sowie das Merkmal der Sichtbarkeit. Das Attribut Richtung hat zunächst keinen Einfluss auf die Darstellung des Sprite-Objekts. Es besteht die Möglichkeit, das Sprite-Objekt in diese Richtung zu bewegen.

Konstruktoren

`Sprite(Shapes shape)`

Erzeugt ein neues Sprite-Objekt, das aus dem angegebenen Shapes-Objekt besteht. Mit dem Auftrag `add` können dem Sprite-Objekt weitere Grafik-Objekt hinzugefügt werden. Die Richtung erhält den Wert 90^0.

`Sprite()`

Erzeugt ein neues Sprite-Objekt, das noch kein Shapes-Objekt enthält. Mit dem Auftrag `add` können dem Sprite-Objekt beliebige Grafik-Objekt hinzugefügt werden. Die Richtung erhält den Wert 90^0.

Methoden

`void add(Shapes shape)`

Fügt dem `Sprite-Objekt` ein weiteres Shapes-Objekt hinzu.

`Sprite clone()`

Liefert eine Kopie des die Methode aufrufenden Picture-Objekts.

2.7 Die Klasse Text

Ein Objekt der Klasse `Text` ist ein Shapes-Objekt auf der Zeichenfläche, mit dem ein beliebiger Text dargestellt wird. Es besitzt als Attribute den Text als String-Objekt, die Textposition, die Schriftart, die Schriftgröße, die Schriftfarbe sowie das Merkmal der Sichtbarkeit. Das Attribut Richtung hat zunächst keinen Einfluss auf die Darstellung des Text-Objekts. Es besteht die Möglichkeit, das Text-Objekt in diese Richtung zu bewegen.

Konstruktoren

`Text(double xp, double yp, String text)`

Erzeugt ein neues Text-Objekt und stellt es an der angegebenen Position und in schwarzer Farbe auf der Zeichenfläche dar. Die Ausgabe auf der Zeichenfläche erfolgt in einer Serifen-Schrift in der Schriftgröße 24 Punkte mit nicht fettem Stil. Die Richtung erhält den Wert 90^0.

`Text(double xp, double yp, String text, Color fontColor)`

Erzeugt ein neues Text-Objekt und stellt es an der angegebenen Position und in der angegebenen Farbe auf der Zeichenfläche dar. Die Ausgabe auf der Zeichenfläche erfolgt in einer Serifen-Schrift in der Schriftgröße 24 Punkte mit nicht fettem Stil. Die Richtung erhält den Wert 90^0.

Methoden

`void setText(String text)`

Weist dem Text -Objekt einen neuen Text zu und stellt es auf der Zeichenfläche dar.

`String getText()`
Liefert die Zeichenkette des Text-Objektes als String-Objekt.

`void setFontMonospaced(boolean bold, int size)`
Das Text -Objekt wird in einer nicht proportionalen Schrift in der angegebenen Punktgröße ggf. fett auf der Zeichenfläche angezeigt.

`void setFontSansSerif(boolean bold, int size)`
Das Text-Objekt wird in einer serifenfreien Proportionalschrift in der angegebenen Punktgröße ggf. fett auf der Zeichenfläche angezeigt.

`void setFontSerif(boolean bold, int size)`
Das Text-Objekt wird in einer Proportionalschrift mit Serifen in der angegebenen Punktgröße ggf. fett auf der Zeichenfläche angezeigt.

`Color getFontColor()`
`Liefert die Schriftfarbe des Text-Objektes.`

`void setFontColor(Color fontColor)`
Ändert die Schriftfarbe.

2.8 Methoden für alle Shapes-Objekte

`double getShapeHeight()`
Liefert die momentane Höhe des minimalen Rechtecks, das das Shapes-Objekt umgibt.

`double getShapeWidth()`
Liefert die momentane Breite des minimalen Rechtecks, das das Shapes-Objekt umgibt.

`double getShapeX()`
Liefert die momentane x-Position des minimalen Rechtecks, das das Shapes-Objekt umgibt.

`double getShapeY()`
Liefert die momentane y-Position des minimalen Rechtecks, das das Shapes-Objekt umgibt.

`double getCenterX()`
Liefert die x-Koordinate des Mittelpunktes des Shapes-Objektes.

`double getCenterY()`
Liefert die y-Koordinate des Mittelpunktes des Shapes-Objektes.

`Color getColor()`
Liefert die Farbe des Shapes-Objektes.

`double getDirection()`
Liefert die Richtung des Shapes-Objektes. Die Richtung liegt immer zwischen 0^0 (einschließlich) und 360^0 (ausschließlich).

`boolean getHidden()`
Liefert `true`, wenn das Shapes-Objekt sichtbar ist und `false` wenn es nicht sichtbar ist.

`float getTransparency()`
Die Abfrage liefert den Wert der Transparenz des Shapes-Objektes als Dezimalzahl zwischen 0 und 1. Bei 0 ist das Objekt vollständig transparent, bei 1 nicht transparent.

`void setTransparency(float alpha)`
Die Methode setzt den Grad der Transparenz des Shapes-Objektes. Der Parameter muss als Wert eine Dezimalzahl im Bereich von 0 bis 1 haben und gib den anteiligen Grad der Transparenz des Objektes an.

`void setColor(Color color)`
Setzt die Füllfarbe des Objektes.

`void setDirection(double direction)`
Weist dem Shapes-Objekt eine neue Richtung zu. Die Methode verändert die Darstellung des Objektes nicht.

`void setHidden(boolean hidden)`
Macht das Objekt beim Parameterwert `false` sichtbar oder beim Parameterwert `true` unsichtbar.

`void flipHorizontal()`
Das Shapes-Objekt wird an der Parallelen zur y-Achse durch seinen Mittelpunkt gespiegelt.

`void flipVertical()`
Das Shapes-Objekt wird an der Parallelen zur x-Achse durch seinen Mittelpunkt gespiegelt.

`void move(double d)`
Das Shapes-Objekt wir um d Einheiten in seine Richtung bewegt.

`void move(double dx, double dy)`
Das Shapes-Objekt wird von seiner aktuellen Position um dx-Einheiten in x- und um dy-Einheiten in y-Richtung verschoben.

`void moveTo(double x, double y)`
Das Shapes-Objekt wird an die angegebene Position bewegt.

`void scale(double sx, double sy)`
Das Shapes-Objekt wird um den Faktor `sx` in x- und um den Faktor `sy` in y-Richtung skaliert. Der Mittelpunkt des Shapes-Objektes bleibt an seiner Position.

`void scaleTo(double width, double height)`
Das Shapes-Objekt wird auf die Breite `width` und die Höhe `height` skaliert, bezogen auf das minimae umgebende Rechteck. Der Mittelpunkt des Shapes-Objektes bleibt an seiner Position.

`void turn(double angle)`
Das Objekt wird um den angegebenen Winkel gedreht. Drehpunkt ist der Mittelpunkt des Shapes-Objektes. Die Richtung des Shapes-Objektes wird entsprechend angepasst.

`void turn(double x, double y, double angle)`
Das Shapes-Objekte wird um den angegebenen Winkel gedreht. Drehpunkt ist der Punkt mit den angegebenen Koordinaten. Die Richtung des Shapes-Objektes bleibt unverändert.

`void turnTo(double angle)`
Das Shapes-Objekte wird in die durch den Winkel angegebene Richtung gedreht. Drehpunkt ist der Mittelpunkt des Shapes-Objektes. Die Richtung des Objektes wird entsprechend angepasst.

`void turnTo(double x, double y)`
Das Shapes-Objekte wird in die Richtung des angegebenen Punktes gedreht. Drehpunkt ist der Mittelpunkt des Shapes-Objektes. Die Richtung des Shapes-Objektes wird entsprechend angepasst.

`boolean intersects(Shapes shape)`
Liefert `true`, wenn sich das Shapes-Objekt mit dem als Parameter angegebenen Shapes-Objekt überschneidet. Die Abfrage berücksichtigt nur sichtbare Objekte.

`boolean contains(Shapes shape)`
Liefert `true`, wenn das Shapes-Objekt das als Parameter übergebene Shapes-Objekt vollständig umschließt. Die Abfrage berücksichtigt nur sichtbare Objekte.

`int getMouseX()`
Die Anfrage liefert die x-Koordinate der letzten Klickposition innerhalb des Shapes-Objektes bezogen auf die Zeichenfläche.

`int getMouseY()`
Die Anfrage liefert die y-Koordinate der letzten Klickposition innerhalb des Shapes-Objektes bezogen auf die Zeichenfläche.

`boolean mouseClicked()`
Liefert `true`, wenn mit der Maus innerhalb des Shapes-Objektes geklickt wurde.

`boolean mouseDragged()`
Liefert `true`, wenn die linke Maustaste gedrückt ist und die Maus innerhalb des Shapes-Objektes bewegt wird.

`boolean mousePressed()`
Liefert `true`, wenn die linke Maustaste innerhalb des Shapes-Objektes gedrückt ist.

`void reset()`
Setzt die Attribute des Shapes-Objekt auf die Anfangswerte zurück.

3. Die Klasse Tools

Die Klasse `Tools` enthält ausschließlich statische Klassenmethoden, die direkt nach dem Klassennamen aufgerufen werden.

Statische Methoden

`static double degreeToRadian(double angle)`
Liefert das Bogenmaß zu dem im Gradmaß übergebenen Winkel

`static double getDirection(double x1, double y1, double x2, double y2)`
Liefert im Gradmaß den Winkel im Uhrzeigersinn zwischen der Nordrichtung und dem Vektor, der durch die als Paramter übergebenen Koordinaten festgelegt ist.

`static int randomNumber(int from, int to)`
Die Anfrage liefert eine ganzzahlige Zufallszahl `von` „from“ `bis` „to“ jeweils einschließlich.

`static int getHour();`
Liefert die aktuelle Stunde.

`static int getMinute()`
Liefert die aktuelle Minute.

`static int getSecond()`
Liefert die aktuelle Sekunde.

`static long getStartTime()`
Liefert die aktuelle Uhrzeit in Millisekunden. Dient als Startzeit für eine Stoppuhr.

`static float getElapsedTime(long startTime)`
Liefert die Zeit, die seit übergebenen Startzeit vergangen ist, als Dezimalzahl (`float`) in Sekunden mit einer Dezimalstelle. Dient als Stoppuhr.

`static String getElapsedTimeString(long startTime)`
Liefert die Zeit, die seit der übergebenen Startzeit vergangen ist, als
String in Sekunden mit einer Dezimalstelle. Dient als Stoppuhr.

`static void message(String msg, String ueberschrift))`
Erzeugt ein modales Nachrichtenfenster mit den als Parameter übergebenen Texten,, das durch Drücken des OK-Buttons geschlossen wird.

`static int confirmDialog(String msg)`
Erzeugt ein modales Fenster. Die als Parameter übergebene Frage muss mit dem „ja“- oder dem „nein“-Button beantwortet werden. Rückgabe 0 bei „ja“ und 1 bei „nein“.

`static String inputDialog(String msg)`
Erzeugt ein modales Fenster mit dem als Parameter übergebenen Text und einem Eingabefeld. Wird das Fenster mit dem „ok“-Button geschlossen, wird der Text des Eingabefeldes zurückgegeben. Wird „Abbrechen“ gedrückt, wird null zurückgegeben.

II. Package sasio

Das Package wird vollständig mit dem Befehl `import sasio.*` geladen.

1. Die Klasse Textfield

Ein Objekt dieser Klasse ist ein editierbares Texteingabefeld.

Konstruktor

`Textfield(int x, int y, int width, int height, String text, View view)`

Ein Objekt der Klasse ist ein Texteingabefeld, das an der angegebenen Position in den angegebenen Maßen auf der Zeichenfläche mit dem als Parameter übergebenen Text abgebildet wird. Um auf die Tastatur zugreifen zu könnem, muss zwingend eine Referenz Objekt der Klasse View als Parameter übergeben werden. Bevor eine Text eingegeben werden oder ediert werden kann, muss das Objekt mit `setActivated` aktiviert werden

Methoden

`String getText()`

Die Methode liefert den Text des Textfeldes

`deleteText()`

Löscht den Text des Eingabefeldes.

`setActivated(boolean activated)`

Mit dem Parameter `true` wird das Textfeld aktiviert und mit `false` deaktiviert. Nach Aktivierung kann der Text des Textfeldes überschrieben oder ediert werden. Der Cursor kann mit den Pfeiltasten im Text bewegt werden. Es sollte immer nur ein Textfeld aktiviert sein. Bei nicht aktivierten Objekten werden Mausklicks oder Eingaben mit der Tastatur ignoriert.

`boolean getActivated()`

Liefert `true`, wenn das Objekt aktiviert ist, andernfalls `false`.

`boolean enterPressed()`

Liefert `true`, wenn die Enter-Taste gedrückt wurde, während das Objekt aktiviert ist.

`void setHidden(boolean pHidden)`

Macht das Objekt unsichtbar, wenn der Parameter den Wert `true` hat oder sichtbar beim Wert `false`. Bei versteckten Objekten werden Mausklicks ignoriert.

`boolean getHidden()`

Liefert `true`, wenn das Objekt unsichtbar ist, `false`, wenn es sichtbar ist.

`boolean clicked()`
Die Methode liefert `true` zurück, wenn mit der Maus in das Textfeld geklickt wurde, andernfalls `false`.

2. Die Klasse Button

Ein Objekt dieser Klasse ist ein klickbarer Button auf der Zeichenfläche.

Konstruktor

`Button (int xPos, int yPos, int width, int height, String label, Color col)`
Ein Objekt dieser Klasse ist ein klickbarer Button auf der Zeichenfläche. Die Parameter legen die Position, die Größe die Aufschrift sowie die Farbe des Buttons fest.

Methoden

`boolean clicked()`
Liefert `true`, wenn der Button geklickt wurde, andernfalls `false`.

`setActivated(boolean activated)`
Mit dem Parameter `true` wird der Button aktiviert und mit `false` deaktiviert. Beim deaktivierten Button ist die Aufschrift grau, beim aktivierten schwarz. Der deaktivierte Button ignoriert Mausklicks.

`boolean getActivated()`
Liefert `true`, wenn das Objekt aktiviert ist, andernfalls `false`.

`void setHidden(boolean pHidden)`
Macht das Objekt unsichtbar, wenn der Parameter den Wert `true` hat oder sichtbar beim Wert `false`. Bei versteckten Objekten werden Mausklicks ignoriert.

`boolean getHidden()`
Liefert `true`, wenn das Objekt unsichtbar ist, `false`, wenn es sichtbar ist.

3. Die Klasse Label

Ein Objekt dieser Klasse ist ein Text in farbigem Rechteck auf der Zeichenfläche. Die Klasse ist eine Unterklasse der Klasse Sprite. Objekte dieser Klasse können daher mit view.remove(<Objektvariable>) von der Zeichenfläche gelöscht werden.

Konstruktor

`public Label(int x, int x, int width, int height, String text, Color color, boolean right, boolean mono)`

Erzeugt ein Rechteck mit den angegebenen Maßen in der angegebenen Farbe, das den als Parameter übergebenen Text enthält. Die Größe des Textes hängt von der Größe des Rechtecks ab. Passt der Text in der Länge nicht in das Rechteck, wird die Schrift solange verkleinert bis er passt.

`public Label(int x, int x, int width, int height, String text, Color color, boolean right, boolean mono)`

Erzeugt ein Rechteck mit den angegebenen Maßen in der angegebenen Farbe, das den als Parameter übergebenen Text enthält. Hat der Parameter `right` den Wert `true`, wird der Text rechtsbündig in dem Rechteck angezeigt, bei `false` linksbündig. Hat der Parameter mono den Wert `true`, wird der Text in einer Mono-Schrift, d. h. nicht proportional, beim Wert `false` in serifenloser Proprotionalschrift. Die Größe des Textes hängt von der Größe des Rechtecks ab. Passt der Text in der Länge nicht in das Rechteck, wird die Schrift solange verkleinert bis er passt.

Methoden

`String getLabelText()`

Liefert den Text des Objektes als String

`setLabelText(String newText)`

Ändert den Text des Objektes auf den als Parameter übergebenen Wert.

`void setHidden(boolean pHidden)`

Macht das Objekt unsichtbar, wenn der Parameter den Wert `true` hat oder sichtbar beim Wert `false`. Bei versteckten Objekten werden Mausklicks ignoriert.

`boolean getHidden()`

Liefert `true`, wenn das Objekt unsichtbar ist, `false`, wenn es sichtbar ist.

4. Die Klasse StringFileTools

Die Klasse enthält statische Klassenmethoden zum Verwalten von String-Dateien.

Methoden

`static boolean fileExists(String filename){`
Die Methode liefert nur dann true, wenn eine Datei mit dem angegebenen Namen (einschließlich Pfad) existiert.

`static String[] loadFileInStringArray(String filename)`
Liest die Datei mit dem als Parameter übergebenen Namen und liefert ihre Zeilen als String-Array zurück. Die erste Zeile der Datei hat den Index 0 usw. Wird die Datei nicht gefunden, tritt ein Laufzeitfehler auf.

`static String loadFileInString(String filename)`
Liest die Datei mit dem als Parameter übergebenen Namen und liefert sie in einem String zurück. Wird die Datei nicht gefunden, tritt ein Laufzeitfehler auf.

`static int countFileRows(String filename)`
Gibt die Anzahl der Zeilen der String-Datei mit dem als Parameter übergebenen Namen zurück.

`static void writeInFile(String filename, String dateiInhalt)`
Erstellt oder überschreibt eine Datei mit dem übergebenen Namen und schreibt den übergebenen String in die Datei. Am Ende des Textes wird automatisch ein Zeilenumbruch eingefügt.

`static void appendToDatei(String filename, String dateiInhalt)`
Erweitert eine bereits vorhandene Datei um den übergebenen Inhalt. Am Ende des anzufügenden Textes wird automatisch ein Zeilenumbruch eingefügt. Wird die Datei nicht gefunden, gibt es einen Laufzeitfehler.