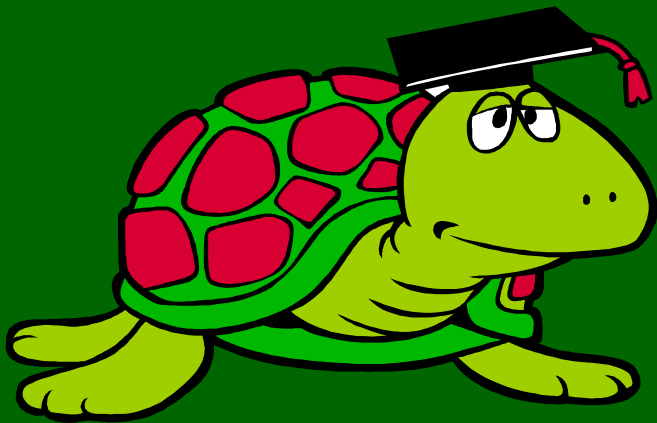
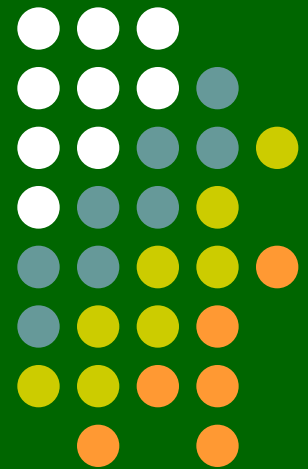


Elemente der Softwareentwicklung

Nutzung von Turtle-Objekten

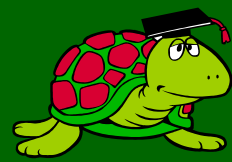


Tino Hempel






Abiturhinweise 2010/11



Vorabhinweise 2010, Punkt 5: Curriculare Hinweise

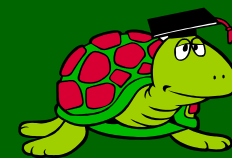
„Die folgenden inhaltlichen Hinweise sind Konkretisierungen zu den Themenfeldern des Kerncurriculums in Hinblick auf die schriftliche Abiturprüfung 2010/11. Diese Ausrichtung auf Prüfungsschwerpunkte kann von Jahr zu Jahr variieren und stellt **keine** inhaltliche Einschränkung des Kerncurriculums dar.“

3. Software-Entwicklung

A	UML-Klassendiagramm interpretieren und erweitern Problemlösen mit Turtleobjekten  Attribute zuordnen und Methoden verändern und implementieren Algorithmenstrukturen anwenden Softwarelebenszyklus
B	Listen implementieren und Listenmethoden anwenden Interpretation von Struktogrammen Logische Programmierung: Listen in PROLOG

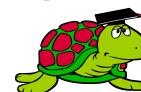


Abiturhinweise 2010



Vorabhinweise 2010/2011, Punkt 2.3 – Hilfsmittel

* **zusätzliche Bibliotheken**
Ein- und Ausgabe,
Datentypenkonvertierung,
Textdatei lesen und schreiben,
Turtle

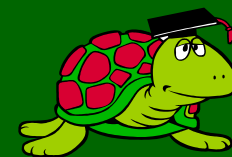


Zusammenfassung

- Es geht „nur“ um das Problemlösen auf Grundkursniveau!
- Als Hilfsmittel (Visualisierung) werden Turtle-Objekte genutzt!

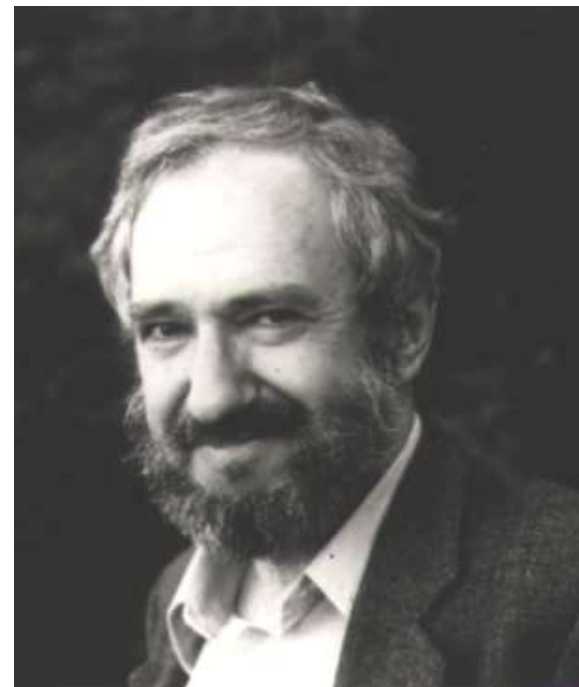


Geschichte der Turtle



Seymour Papert

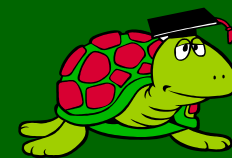
- KI-Bereich des MIT in den 1970er
- Projekte für Kinder mit den Zielen:
 - Selbständiges Erschließen geometrischer Zusammenhänge
 - Computer als Werkzeug zur Erzeugung geometrischer Figuren
 - Veranschaulichung der Mathematik, insbesondere der Geometrie
- Ergebnis:
Programmiersprache LOGO mit Turtle-Geometrie und präobjektorientierter Sichtweise



Quelle: <http://web.media.mit.edu/~papert/>



Geschichte der Turtle



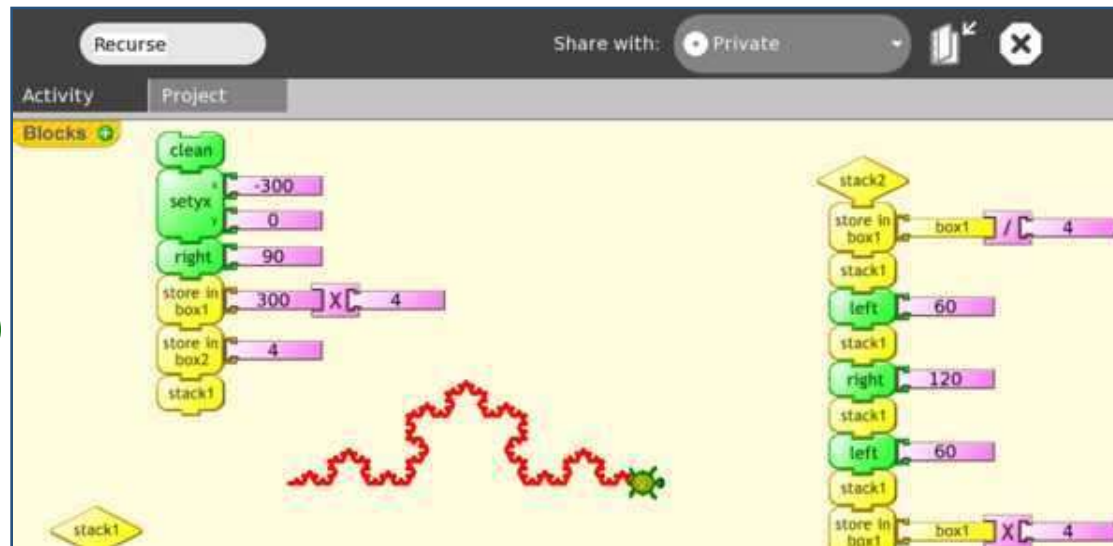
Kritik

„Zugleich ist ihr Kernstück, die Turtle-Geometrie, unter fachinhaltlichen, anwendungspraktischen und lebensweltbezogenen Aspekten zu wenig relevant.“

Quelle: P. Bender: Kritik der Logo-Philosophie. In: Journal für Mathematik-Didaktik 8. Heft 1/2. 1987.

Weiterentwicklungen der Turtle-Ideen am MIT

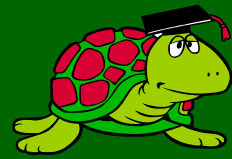
- LEGO Mindstorms
- Scratch & Co.
- [MaMaMedia](http://www.mamamedia.org)
- OLPC (100 \$ Laptop)



Quelle: <http://wiki.laptop.org/images/e/e6/Screenshot.png>



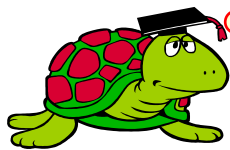
Turtle – Bestandsaufnahme



Bestandsaufnahme

- Turtle-Systeme für die meisten Programmiersprachen frei verfügbar
- Umfang, Methodenbezeichner/-struktur und Einbindung/Nutzung jedoch sehr unterschiedlich

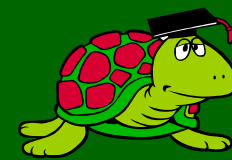
**Nur der kleinste
gemeinsame Nenner ist
im Abitur nutzbar.**



→ Analyse vorhandener **Java**-Bibliotheken



Turtle – Bestandsaufnahme

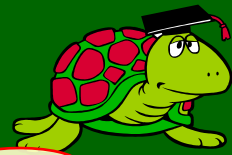


- BlueJTurtle von Alfred Hermes

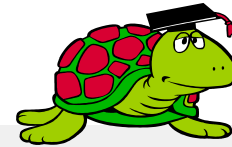
The screenshot shows the BlueJ IDE interface. On the left, the 'Turtle Version 6a (c) Alfred Hermes' window displays a colorful spiral path drawn by a turtle cursor. The cursor is labeled 'Turtleseh nicht'. On the right, the 'BlueJ: turtleliebe' window shows a class diagram with two classes: 'TurtleRennen' and 'Verfolgung'. A dashed arrow points from 'TurtleRennen' to 'Verfolgung'. Below the diagram, a red box contains the text 'turtleRe1: TurtleRennen'. A yellow thought bubble with a red border contains the text: 'Ggf. Fehlermeldung, falls Fenster mehrfach falsch geschlossen wird.'



Turtle – Bestandsaufnahme



- JavaTurtle von G. Röhner



Hier bin ich!

The screenshot shows an IDE window with the following components:

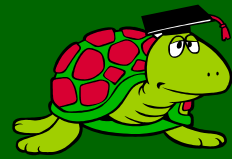
- Menu Bar:** Datei, Bearbeiten, Start, Test, UML, Komponenten, Tools, Fenster, Hilfe
- Toolbar:** Includes icons for file operations, editing, and development tools.
- Project Explorer:** Shows files JE1.java and JE1.jfm.
- Code Editor:** Contains the following Java code:


```

44     setResizable(false);
45     setVisible(true);
46 }
47
48 // Anfang Ereignisprozeduren
49 public void jBtnStart_ActionPer
50     for (int i = 1; i <= 6; i++)
51     {
52         hans.draw(50);
53         hans.turn(60);
54     }
55 }
56
57 // Ende Ereignisprozeduren
      
```
- Output/Console Window:** Shows a window titled "JE1" with a drawing of a hexagon and a "Start" button.



Turtle – Bestandsaufnahme



- JavaTurtle von Ä. Plüss (Java exemplarisch)

```
4  *
5  * @version 1.0 vom 12.01.200
6  * @author
7  */
8  import ch.aplu.turtle.*;
9
10 public class AP1 {
11
12     public static void main(Stri
13         Turtle john = new Turtle(
14         int i = 0;
15         do
16         {
17             john.forward(20);
18             john.right(90);
19             i++;
20         } while (i<5);
21     }
22 }
23
```

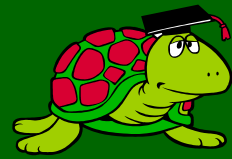
Java Turtle Playground

Diese Zeile muss immer vor der Klassendefinition stehen!





Turtle – Bestandsaufnahme



- JavaTurtle von SwissEduc-Team

The screenshot displays the JavaTurtle programming environment, split into two main windows.

Left Window: JavaTurtle programmieren
The editor shows the code for a spiral program in `Spirale.java`. The code is as follows:

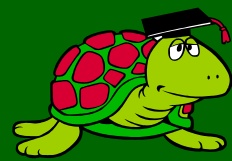
```
*/  
import javaturtle.JavaTurtleProgram;  
  
public class Spirale extends JavaTurtleProgram {  
  
    // hier können Sie eigene Methoden definieren  
  
    void spiral (int size, int angle, int stepSize, int max  
        while (size <= maxSize) {  
            turtle.setColor(size % 256, (size / 2) % 256, (size  
            turtle.forward(size);  
            turtle.turn(angle);  
            size += stepSize;  
        }  
    }  
}
```

Below the code editor is a "Programm kompilieren" button and a terminal window showing the execution command: `Executed: C:\Programme\Java\jdk1.6.0_05\bin\javac.exe -class`. The status bar at the bottom indicates "Zeile: 0, Spalte: 0".

Right Window: JavaTurtle, die Java-Schildkröte
The main window features a "Programmieren" button and a toolbar. The central "Welt" (World) area contains a green turtle wearing a graduation cap and a square spiral drawing. A yellow thought bubble above the turtle contains the text "Beta-Version". At the bottom, there is a "Geschwindigkeit" (Speed) slider ranging from "langsam" to "schnell" and an "Ausführen" (Run) button with play, pause, and stop icons.



Turtle – Bestandsaufnahme

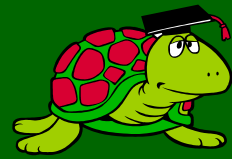


- Turtle in Greenfoot

The screenshot shows the Greenfoot turtleGraphics application window. The main canvas, titled "turtleWorld", displays a green turtle icon at the bottom center. To its left, a black spiral drawing is visible, and to its right, a red square drawing is shown. The interface includes a menu bar with "Scenario", "Edit", "Controls", and "Help". On the right side, there is a "Scenario Information" panel and a class hierarchy diagram. The class hierarchy is divided into "World classes" and "Actor classes". Under "World classes", "World" is the superclass and "TurtleWorld" is the subclass. Under "Actor classes", "Actor" is the superclass, and "Turtle" is the subclass. "Turtle" has three subclasses: "SpiralTurtle", "CircleTurtle", and "SquareTurtle". "SquareTurtle" has one subclass: "FlowerTurtle". At the bottom of the window, there are buttons for "Act", "Run", and "Reset", along with a "Speed" slider and a "Compile all" button.



Turtle – Bestandsaufnahme

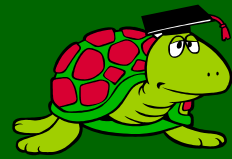


- A. Hermes:
<http://www.zitadelle.juel.nw.schule.de/if/java/java.html>
- Ä. Plüss:
<http://www.aplu.ch/home/download.jsp>
- G. Röhner:
<http://www.javaeditor.org/>
- SwissEduc:
<http://www.swisseduc.ch/informatik/turtles/>
- Greenfoot-Turtle:
<http://www.greenfoot.org/scenarios/index.html>





Turtle-Klasse einbinden



Es kann nur Einen geben!

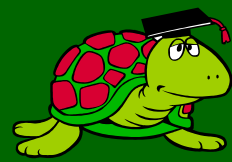
Installieren Sie stets nur eine der Turtle-Klassen in die Entwicklungsumgebung, da es sonst zu Fehlverhalten kommen kann!




Alternative (auch für Schüler): IoStick Version 2.1c mit allen Turtle-Versionen, Scratch und Greenfoot



Turtle – Bestandsaufnahme



Vergleich der „fertiger“ Turtle-Klassen

	Hermes	Röhner	Plüss	SwissEdu	Greenfoot
Einbindung	BlueJ/Java-Console	JavaEditor Komponente	BlueJ/Java/ Online	Eigene Umgebung	Szenario
Methoden Bezeichner Java-Hilfe	genügende dt./engl. dt. Word	minimale englisch dt. JavaDoc	viele englisch engl. JavaDoc	minimale englisch dt. Hilfe	minimale englisch engl. JavaDoc
Ereignissteuerung	Eigenbau	über GUI Eigenbau	teilweise integriert	keine	
Aussehen					

A. Hermes

gekürzt um deutsche Methoden

Turtle

```
Turtle()
Turtle(int, int)
clearScreen(): void
getXPos(): double
getYPos(): double
hideFrame(): void
plotPixel(): void
copyPixel(int, int, int, int): void
penUp(): void
penDown(): void
turn(double): void
turnTo(double, double): void
setSpeed(int): void
hideTurtle(): void
showTurtle(): void
toStartingPoint(double, double): void
moveArc(double, double): void
drawCircle(double): void
fillCircle(double): void
getPixelColor(): int
setFilled(boolean): void
restoreWindow(): void
register(EventStation): void
write(String): void
sleep(int): void
paint(Graphics): void
mouseClicked(MouseEvent): void
mouseDragged(MouseEvent): void
mouseEntered(MouseEvent): void
mouseExited(MouseEvent): void
mouseMoved(MouseEvent): void
mousePressed(MouseEvent): void
mouseReleased(MouseEvent): void
move(double): void
setColor(int): void
getColor(): int
getMaxX(): double
getMaxY(): double
moveTo(double, double): void
actionPerformed(ActionEvent): void
setDirection(double): void
setDirection(double, double): void
getDirection(): double
```

G. Röhner

Turtle

```
Turtle(int, int)
turn(double): void
turnto(double): void
drawto(double, double): void
moveto(double, double): void
clear(): void
paint(Graphics): void
update(Graphics): void
isDoubleBuffered(): boolean
move(double): void
setBackground(Color): void
setForeground(Color): void
draw(double): void
```

Greenfoot

Turtle

```
© Turtle()
⊕ turn(double): void
⊕ moveTo(double, double): void
⊕ move(double): void
⊕ penUp(): void
⊕ penDown(): void
⊕ setColor(String): void
⊕ addToWorld(World): void
⊕ setLocation(double, double): void
⊕ setLocation(int, int): void
```

Ä. Plüss

gekürzt und bearbeitet

Turtle

```
Turtle(Turtle)
Turtle()
addKeyListener(KeyListener)
addMouseListener(MouseListener)
addMouseMotionListener(MotionListener)
back(double): void
beep(): void
clear(): void
distance(double, double): double
fill(): void
forward(double): void
getColor(): Color
getSpeed(): double
getX(): double
getY(): double
heading(): double
heading(double): double
hideTurtle(): void
home(): void
isHidden(): boolean
isPenUp(): boolean
label(String): void
left(double): void
leftCircle(double): void
penDown(): void
penUp(): void
right(double): void
rightCircle(double): void
setColor(Color): void
setFillColor(Color): void
setHeading(double): void
setPenColor(Color): void
setPos(double, double): void
setX(double): void
setY(double): void
showTurtle(): void
speed(double): void
```

SwissEdu

turtle

```
forward(double): void
moveto(double, double): void
turn(double): void
penUp(): void
penDown(): void
show(): void
hide(): void
clearWorld(): void
setDirection(double): void
getDirection(): double
setPosition(double, double): void
setX(double): void
getX(): double
setY(double): void
getY(): double
setPenDown(boolean): void
isPenDown(): boolean
setVisible(boolean): void
isVisible() boolean
setColor(double, double, double): void
getColorRed(): double
getColorGreen(): double
getColorBlue(): double
setPenWidth(double): void
getPenWidth(): double
```

A. Hermes

gekürzt um deutsche Methoden

Turtle

```
Turtle()
Turtle(int, int)
clearScreen(): void
getXPos(): double
getYPos(): double
hideFrame(): void
plotPixel(): void
copyPixel(int, int, int, int): void
penUp(): void
penDown(): void
turn(double): void
turnTo(double, double): void
setSpeed(int): void
hideTurtle(): void
showTurtle(): void
toStartingPoint(double, double): void
moveArc(double, double): void
drawCircle(double): void
fillCircle(double): void
getPixelColor(): int
setFilled(boolean): void
restoreWindow(): void
register(EventStation): void
write(String): void
sleep(int): void
paint(Graphics): void
mouseClicked(MouseEvent): void
mouseDragged(MouseEvent): void
mouseEntered(MouseEvent): void
mouseExited(MouseEvent): void
mouseMoved(MouseEvent): void
mousePressed(MouseEvent): void
mouseReleased(MouseEvent): void
move(double): void
setColor(int): void
getColor(): int
getMaxX(): double
getMaxY(): double
moveTo(double, double): void
actionPerformed(ActionEvent): void
setDirection(double): void
setDirection(double, double): void
getDirection(): double
```

G. Röhner

Turtle

```
Turtle(int, int)
turn(double): void
turnto(double): void
drawto(double, double): void
moveto(double, double): void
clear(): void
paint(Graphics): void
update(Graphics): void
isDoubleBuffered(): boolean
move(double): void
setBackground(Color): void
setForeground(Color): void
draw(double): void
```

keine Getter

Greenfoot

Turtle

```
© Turtle()
⊕ turn(double): void
⊕ moveTo(double, double): void
⊕ move(double): void
⊕ penUp(): void
⊕ penDown(): void
⊕ setColor(String): void
⊕ addToWorld(World): void
⊕ setLocation(double, double): void
⊕ setLocation(int, int): void
```

keine Getter

Ä. Plüss

gekürzt und bearbeitet

Turtle

```
Turtle(Turtle)
Turtle()
addKeyListener(KeyListener)
addMouseListener(MouseListener)
addMouseMotionListener(MotionListener)
back(double): void
beep(): void
clear(): void
distance(double, double): double
fill(): void
forward(double): void
getColor(): Color
getSpeed(): double
getX(): double
getY(): double
heading(): double
heading(double): double
hideTurtle(): void
home(): void
isHidden(): boolean
isPenUp(): boolean
label(String): void
left(double): void
leftCircle(double): void
penDown(): void
penUp(): void
right(double): void
rightCircle(double): void
setColor(Color): void
setFillColor(Color): void
setHeading(double): void
setPenColor(Color): void
setPos(double, double): void
setX(double): void
setY(double): void
showTurtle(): void
speed(double): void
```

**kein geheAuf
mit Spur**

SwissEdu

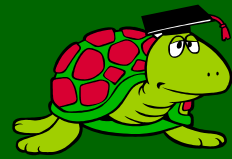
turtle

```
forward(double): void
moveto(double, double): void
turn(double): void
penUp(): void
penDown(): void
show(): void
hide(): void
clearWorld(): void
setDirection(double): void
getDirection(): double
setPosition(double, double): void
setX(double): void
getX(): double
setY(double): void
getY(): double
setPenDown(boolean): void
isPenDown(): boolean
setVisible(boolean): void
isVisible() boolean
setColor(double, double, double): void
getColorRed(): double
getColorGreen(): double
getColorBlue(): double
setPenWidth(double): void
getPenWidth(): double
```

**kein Konstruktor,
da Objekt**



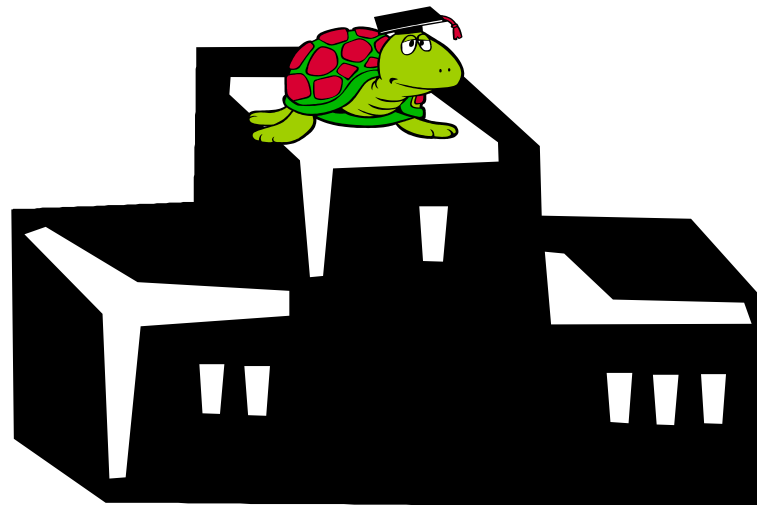
Welche Klasse wird's nun?



Persönliche Empfehlung:

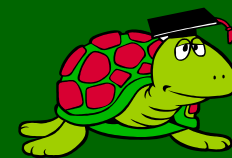


Turtle von A. Hermes



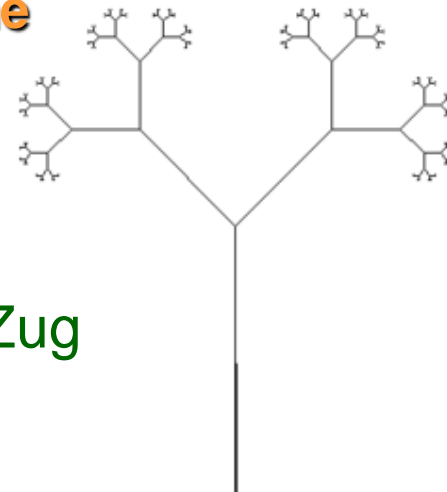


Turtle benutzen – Wir(r)!



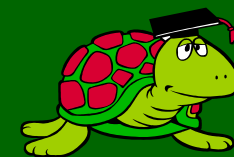
Ideen für Turtle-Einstiege im IU

- Projekt 1: Darstellung KO-System und Funktionen
Orientierung im Turtle-System – Koordinatensysteme
- Projekt 2: Zeichnen geometrischer Figuren
(Rechteck, Quadrat, Stern, Kreis, Dreieck, ...)
Nutzbarkeit von Methoden
- Projekt 3: Zeichnen des Nikolaushauses in einem Zug
(statische Breite, variable Breite, Häuserreihe)
Idee für Einstieg in die OOP – statt BlueJ-Malerei
- Projekt 4: Zeichnen von rekursiven Grafiken (binärer Baum, Kochkurve, ...)
Umgang mit algorithmischen Strukturen und Rekursion
- Projekt 5: Verfolgung zweier Turtles, Steuerung der Turtle(s)
Simulation, Threads, Ereignissteuerung





Brainstorming TiU

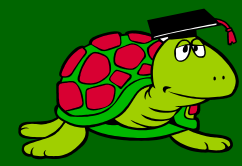


? Grundintension ?

- bewährtes Konzept möglichst beibehalten
- wenige Stunden (max. 5 von 20/40 DS) für Turtle verwenden
- Beispiele:
 - Objects first mit Turtle statt mit Maler (Gierhardt, Pohlig)
 - SOL-Kurs mit Ä. Plüss-Turtle
- Projekte mit Turtle, die alle Aspekte (Turtle, Object first, Datenkapselung, Vererbung, Polymorphie, Algorithmen, ...) umfassen
- Beispiele auf LK-Niveau:
 - Java-ZUM-Projekte
 - G. Röhner: Zeitkomplexität rekursiver Grafikalgorithmen
 - E. Modrow: „LEGO für Arme“
 - Figurenprojekt



Brainstorming TiU



SOL-Kurs: <http://clab2.phbern.ch/lego/turtleGrafik.php>

Turtle-Grafik mit Java 🇬🇧

PHBern Virtueller Campus Projekt

Home
Lernprogramm Turtle-Grafik
drucken
Java-Online

- Home
- Online-Editor
- Lernprogramm
 - erstes Programm
 - Einfaches Zeichnen
 - while-Schleife
 - for-Schleife
 - if-else
 - switch
 - Methoden**
 - Variablen
 - mehrere Turtles
 - Vererbung
 - Komposition
 - Events
 - Rekursionen
 - Turtle Applets
 - Übersicht Methoden
 - Turtle-Dokumentation
- Aufgaben
- Literatur und Links
- Autorenteam

Online-Editor starten

Methoden definieren

Komplexe Programme können mit Hilfe von Methoden in kleinere, leichter zu programmierende Teilprogramme zerlegt werden. Ausser dem werden mit Hilfe von Methoden auch wiederkehrende Programmteile beschrieben, damit sie nicht mehrmals geschrieben werden müssen. Man kann diese Programmieretechnik an ganz einfachen Beispielen üben.

Die folgenden Programme bewirken dasselbe, in beiden Beispielen wird ein Quadrat gezeichnet. Im Beispiel rechts wird dies aber mit der Methode *quadrat* definiert.

Beispiel zeigen (Tu9.java)

```

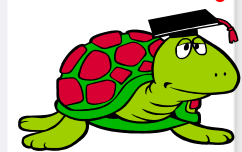
// Tu9.java

import ch.aplu.turtle.*;

class Tu9
{
    Turtle joe = new Turtle();

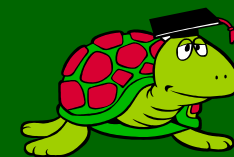
    Tu9()
    {
        joe.hideTurtle();
        for (int k = 0; k < 10; k++)
        {
            quadrat();
            joe.left(36);
        }
    }
}

```





Brainstorming TiU



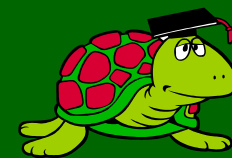
Algorithmik, OO und Rekursion:

<http://www.gierhardt.de/informatik/info11/turtle/turtle.html>

- [Einstiegsprogramm](#) zur Demonstration der Programmstruktur für ein Applet
- [Einstiegsprogramm](#) zur Demonstration der Programmstruktur für eine Applikation
- [Das Interface](#) von Turtle.class
- [Turtle.java](#) direkt zum Download
- [Arbeiten mit Editor, Compiler usw.](#)
- [Haus des Nikolaus](#): Einzelhaus und Häuserreihe sollen gezeichnet werden (Wiederholung: Arbeiten mit Variablen, Parameterübergabe und For-Schleife)
- [Quadratspiralen](#) (Wiederholung: Arbeiten mit Methoden, Variablen, Parameterübergabe und For-Schleife)
- [Aufgaben für Donnerstag, den 17.05.2001](#)
- [Noch eine Spirale](#) (Wiederholung: Arbeiten mit Methoden, Variablen, Parameterübergabe und While-Schleife)
- [Binärer Baum](#) (Rekursion mit Parametern)
- [Baum des Pythagoras](#) (Rekursion mit Parametern)
- [Kochsche Kurve mit mehreren Varianten](#) (Rekursion mit Parametern)
- [Drachenkurve](#) (Rekursion mit Parametern)
- [Sierpinski-Dreieck und Variante](#) (Rekursion mit Parametern)
- ["EKG"-Kurve von Clemens Adolphs und Tom Mannheim](#) (Rekursion mit Parametern)



Brainstorming TiU



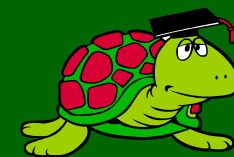
Algorithmik und Brücke zur OOP:

<http://www.pohlig.de/Unterricht/Inf2004/inf2004.htm>

- 2. Zeichnen mit einer Turtle
 - 2.1 Zeichnen mit einer fertigen Turtle
 - 2.2 Übungen und Aufgaben
 - 2.3 Turtle-Methoden
 - 2.4 Übungen und Aufgaben
- 3. Die Modellierung der Turtle
 - 3.1 Klasse und Objekt
 - 3.2 Die Turtle in UML-Notation
 - 3.3 Übungen und Aufgaben
 - 3.4 Eine Turtle-Objekt erzeugen und ein Quadrat zeichnen lassen
- 4. Programmieren mit der Turtle
 - 4.1 Mit einer Java-Turtle in Java programmieren
 - 4.2 Editieren - Compilieren
 - 4.3 Übungen und Aufgaben
 - 4.4 Walter tanzt den Random-Walk
 - 4.5 Walter und Elsa versuchen den Passo-Doble
 - 4.6 Der Turtle Balztanz
 - 4.7 Übungen
 - 4.8 Schleifen
 - 4.8.1 Die for-Schleife
 - 4.8.2 Die do-while-Schleife
 - 4.8.3 Die while-Schleife
 - 4.8.4 Übungen
 - 4.9 Weitere Übungen
 - 4.9.1 Kreise und Gitter
 - 4.9.2 Four Cool Cats
 - 4.10 Verzweigung
 - 4.10.1 if-else - Konstruktion



Brainstorming TiU



Ereignissteuerung, Threads, mehrere Turtle-Objekte: [http://wiki.zum.de/Turtle-Grafik_\(Java\)](http://wiki.zum.de/Turtle-Grafik_(Java))

Die folgenden Beispiele basieren auf der in [Java](#) geschriebenen [Turtle-Grafik-Engine](#) von A. Hermes, die sich problemlos in [BlueJ](#) einbinden lässt. Es fehlen zu den Aufgaben einige Beispielbilder, die ich aus Copyrightgründen nicht einfügen kann.

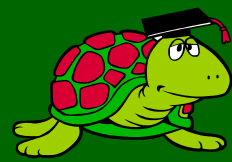
Inhaltsverzeichnis [\[Verbergen\]](#)

- 1 Elementare Methoden
- 2 Erstes Beispiel
- 3 Nach Einführung der for-Schleife
 - 3.1 Zweites Beispiel: Ein Zahlenstrahl
- 4 Eventsteuerung
- 5 Threads
- 6 Perlenkette
- 7 Rudimentärer Funktionsplotter
- 8 Verfolgende Schildkröten
- 9 Uhr
- 10 Weblink
 - 10.1 Andere Turtlelibraries
- 11 Siehe auch





Unterrichtsbeispiel



Einstieg mit Scratch

Vorgabe einer Zeichnung – Analyse: Objekte mit Eigenschaften und Fähigkeiten

→ algorithmische Erweiterung

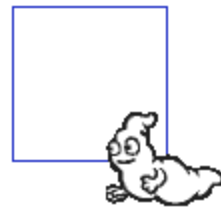
Vorgabe Rechteck



```

Wenn  angeklickt
  zeige dich
  zeige Richtung  $-90^\circ$ 
  gehe zu x: 200 y: -150
  wische Malspuren weg
  senke Stift ab
  setze Stiftfarbe auf 
  wiederhole 4 mal
    gehe  $\text{länge}$  -er Schritt
    drehe  $90^\circ$ 
  stoppe alles
  
```

- ▶ nach links
- ▶ rechts unten



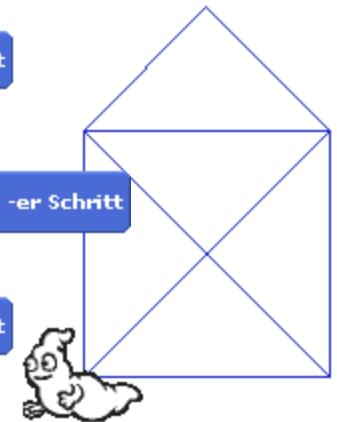
Erweiterung Haus

- ▶ nach links
- ▶ rechts unten



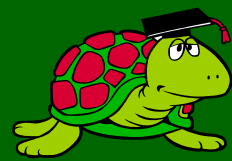
```

Wenn  angeklickt
  zeige dich
  zeige Richtung  $-90^\circ$ 
  gehe zu x: 200 y: -150
  wische Malspuren weg
  senke Stift ab
  setze Stiftfarbe auf 
  wiederhole 4 mal
    gehe  $\text{länge}$  -er Schritt
    drehe  $90^\circ$ 
  drehe  $45^\circ$ 
  gehe  $\sqrt{2} \cdot \text{länge}$  -er Schritt
  drehe  $90^\circ$ 
  wiederhole 2 mal
    gehe  $\sqrt{2} \cdot \text{länge} \cdot 0,5$  -er Schritt
    drehe  $90^\circ$ 
  gehe  $\sqrt{2} \cdot \text{länge}$  -er Schritt
  drehe  $45^\circ$ 
  stoppe alles
  
```





Unterrichtsbeispiel

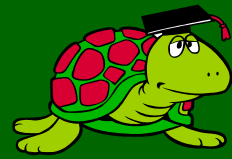


Vergleich Scratch mit Java

Scratch	BlueJ/Java
<pre> Wenn [] angeklickt zeige dich zeige Richtung -90 gehe zu x: 200 y: -150 wische Malspuren weg senke Stift ab setze Stiftfarbe auf wiederhole 4 mal gehe länge -er Schritt drehe 90 Grad drehe 45 Grad gehe Wurzel von 2 * länge -er Schritt drehe 90 Grad wiederhole 2 mal gehe Wurzel von 2 * länge * 0.5 -er Schritt drehe 90 Grad gehe Wurzel von 2 * länge -er Schritt drehe 45 Grad stoppe alles </pre>	<pre> public class Zeichner { //===== Objekt-Felder ===== private Turtle casper; //===== Konstruktor ===== public Zeichner() { casper = new Turtle(500,500); } //===== Objekt-Methoden ===== /** * casper zeichnet ein Rechteck */ public void zeichneRechteck(int pLaenge) { casper.showTurtle(); casper.setDirection(180); //180 ... nach links zeigend casper.toStartingPoint(casper.getMaxX()-20,casper.getMaxY()-20); casper.clearScreen(); casper.penDown(); casper.setColor(2); for (int i = 1; i<= 4; i++) { casper.move(pLaenge); casper.turn(90); } casper.turn(45); casper.move(Math.sqrt(2)*pLaenge); casper.turn(90); for (int i = 1; i<= 2; i++) { casper.move(Math.sqrt(2)*pLaenge*0.5); casper.turn(90); } casper.move(Math.sqrt(2)*pLaenge); casper.turn(45); } } </pre>

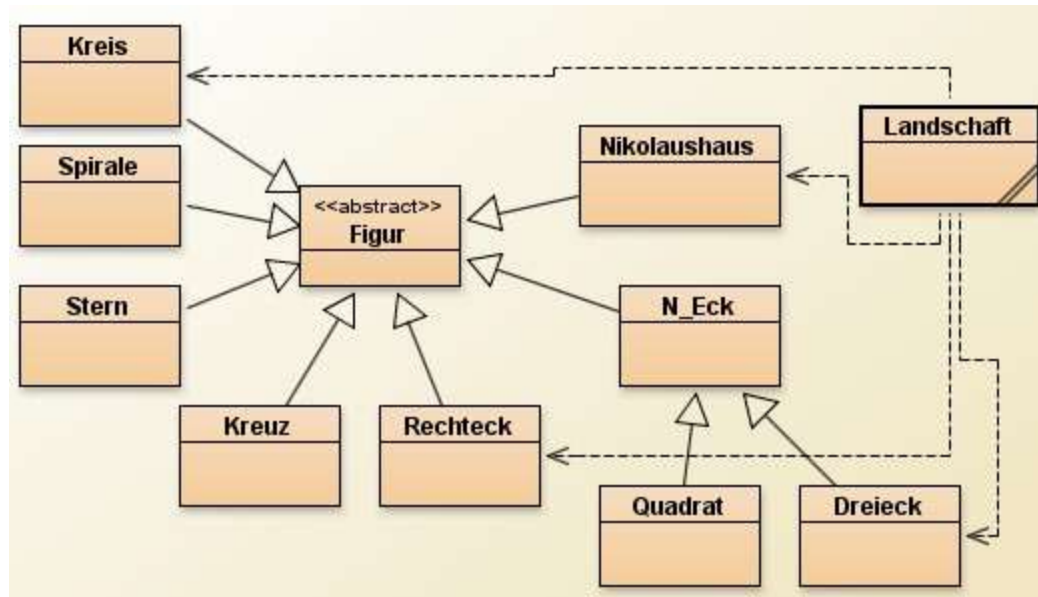


Unterrichtsbeispiel



Entwicklung eines Figurenzeichner und -rechner-Projektes

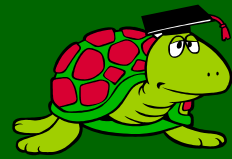
- Planung und schrittweise Entwicklung eines Landschaftzeichners mit mathematischen Elementen 😊
- Grundprinzipien der Vererbung und Abstraktion



- Problem: Figur **hat** Stift und kann sich zeichnen?

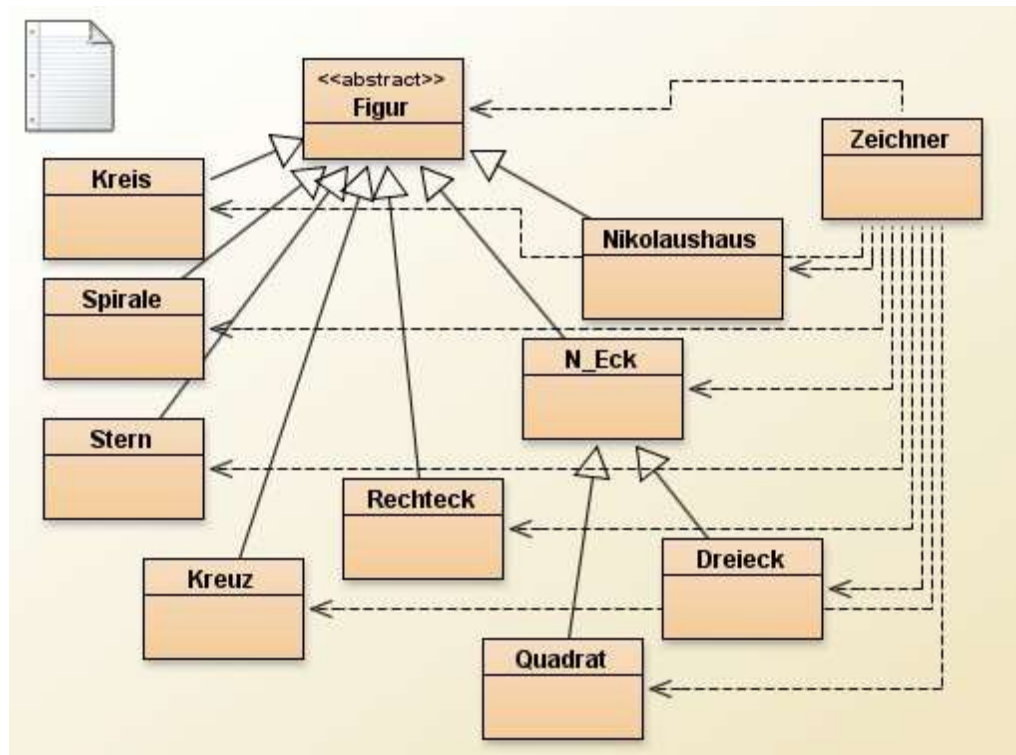


Unterrichtsbeispiel



Entwicklung eines Figurenzeichner und -rechner-Projektes

- Lösung: Figur kann mit Hilfe eines Stifts die Darstellung beschreiben und mit Hilfe eines Zeichner darstellen!





Fragestunde

