Beyond Pascal & Co

Programming Environments for Education in CS

Emanuel Giger & Michael Würsch

Overview

Learning Environments

- Kara
- JGameGrid (aplus.ch)
- Scratch
- StarLogo TNG

Programming Languages

- Scheme
- Groovy

Misc

- Processing
- BlueJ

*Kara

Logo-like environment for introducing students to programming

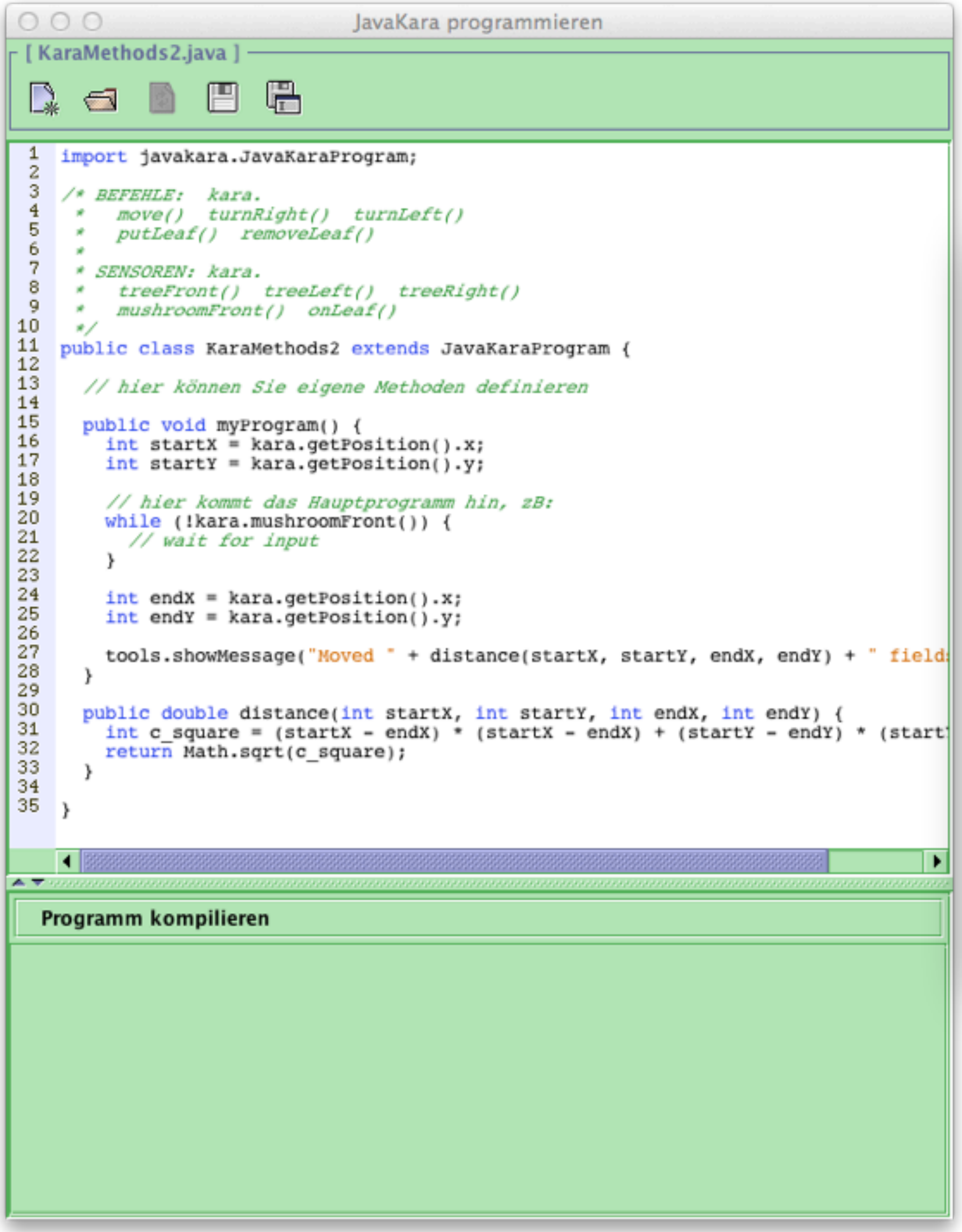
Strengths

- Fast visual results
- Very useful material for teachers ("Leitprogramme")
- Many supported languages and paradigms (State Machines, PythonKara, MultiKara, ScratchKara, etc.)

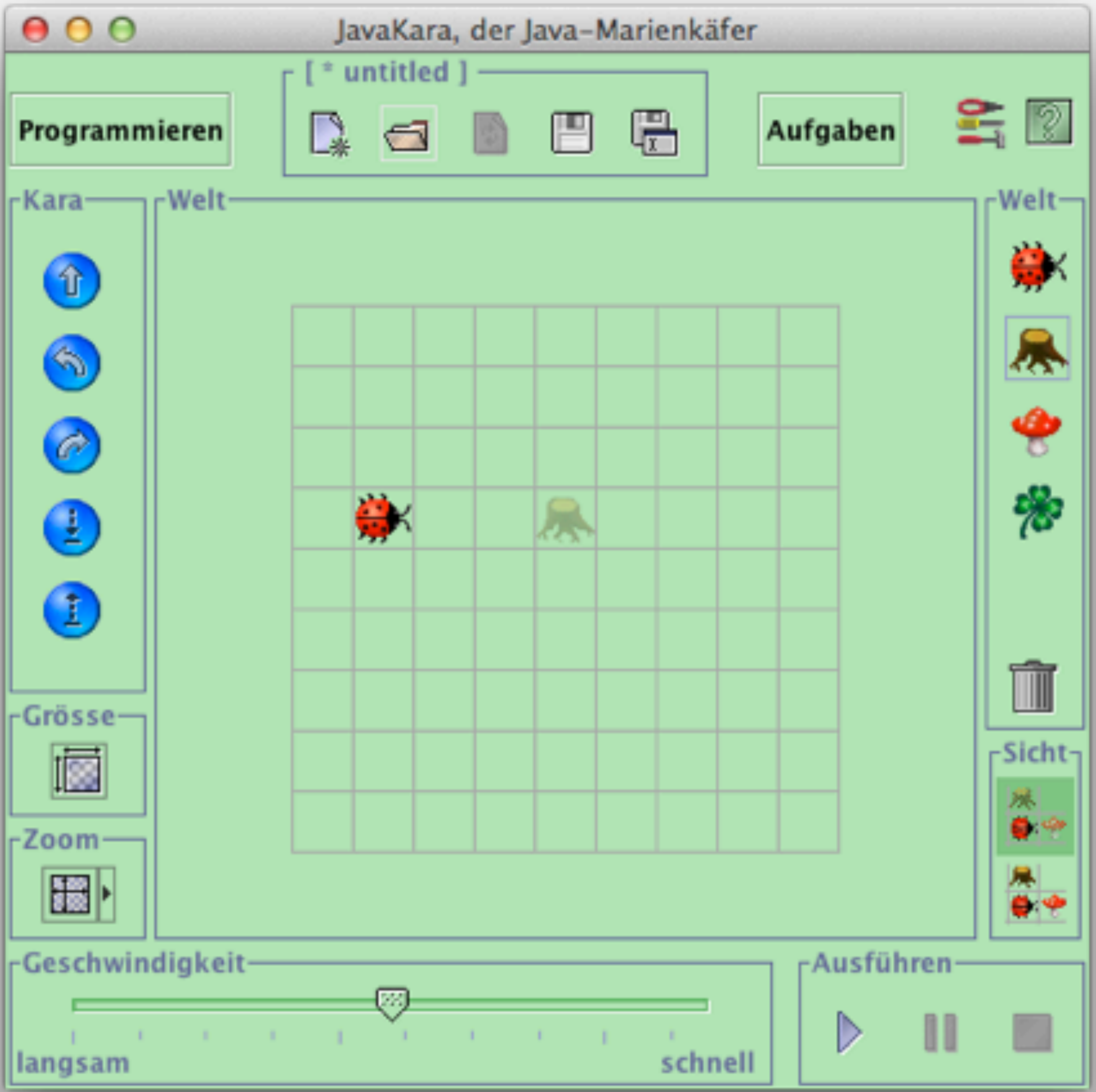
Weaknesses/Limitations

- Full Java syntax (for JavaKara)
- Limited application scenarios
- Outdated UI

http://www.swisseduc.ch/informatik/karatojava



```
import javakara.JavaKaraProgram;

/* BEFEHLE:  kara.
 *   move()  turnRight()  turnLeft()
 *   putLeaf()  removeLeaf()
 *
 * SENSOREN: kara.
 *   treeFront()  treeLeft()  treeRight()
 *   mushroomFront()  onLeaf()
 */
public class KaraMethods2 extends JavaKaraProgram {

  // hier können Sie eigene Methoden definieren

  public void myProgram() {
    int startX = kara.getPosition().x;
    int startY = kara.getPosition().y;

    // hier kommt das Hauptprogramm hin, zB:
    while (!kara.mushroomFront()) {
      // wait for input
    }

    int endX = kara.getPosition().x;
    int endY = kara.getPosition().y;

    tools.showMessage("Moved " + distance(startX, startY, endX, endY) + " field
  }

  public double distance(int startX, int startY, int endX, int endY) {
    int c_square = (startX - endX) * (startX - endX) + (startY - endY) * (start
    return Math.sqrt(c_square);
  }

}
```



```java
import javakara.JavaKaraProgram;

/* BEFEHLE:  kara.
 *   move()  turnRight()  turnLeft()
 *   putLeaf()  removeLeaf()
 *
 * SENSOREN: kara.
 *   treeFront()  treeLeft()  treeRight()
 *   mushroomFront()  onLeaf()
 */
public class KaraMethods extends JavaKaraProgram {

  public void myProgram() {
    while (true) {
      if(kara.treeFront()) {
         byPass();
      } else if(kara.mushroomFront()) {
         turnAround();
      } else {
         kara.move();
      }
    }
  }

  public void turnAround() {
     kara.turnLeft();
     kara.turnLeft();
  }

  public void byPass() {
     kara.turnLeft();
     kara.move();
     kara.turnRight();
     kara.move();
     kara.move();
     kara.turnRight();
     kara.move();
     kara.turnLeft();
  }
}
```

JGameGrid

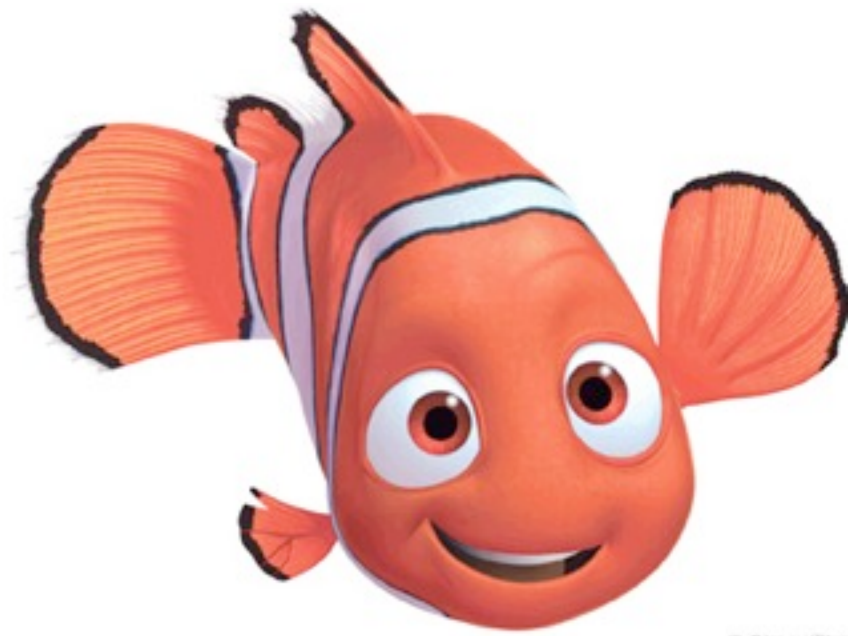
An Education Oriented Java Package for Developing Computer Games

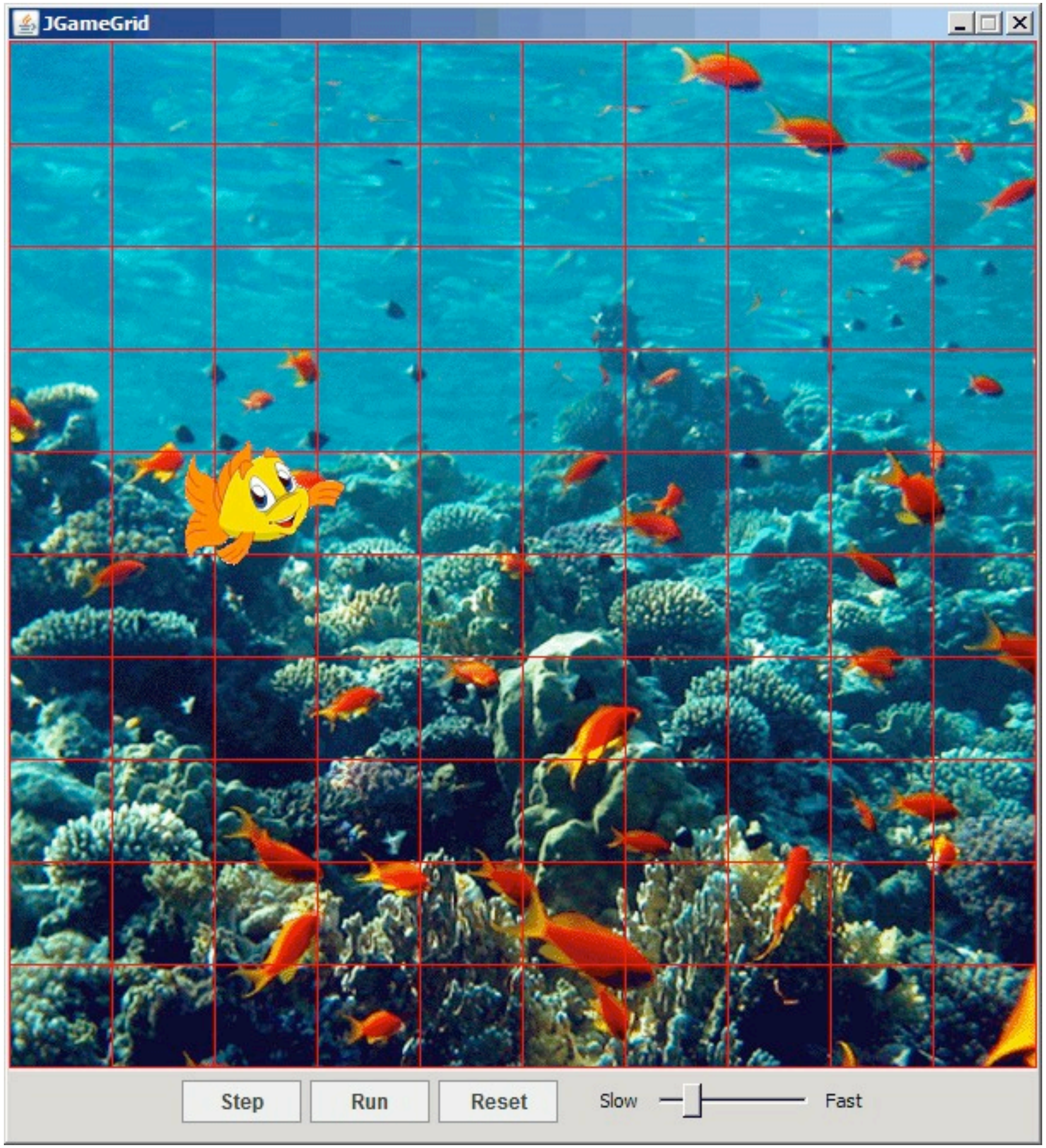
Strengths

- Good focus on OOP due to Actor-concept
- Hides complexity of Threading, Java UI programming, Collision Detection, etc.

Weaknesses/Limitations

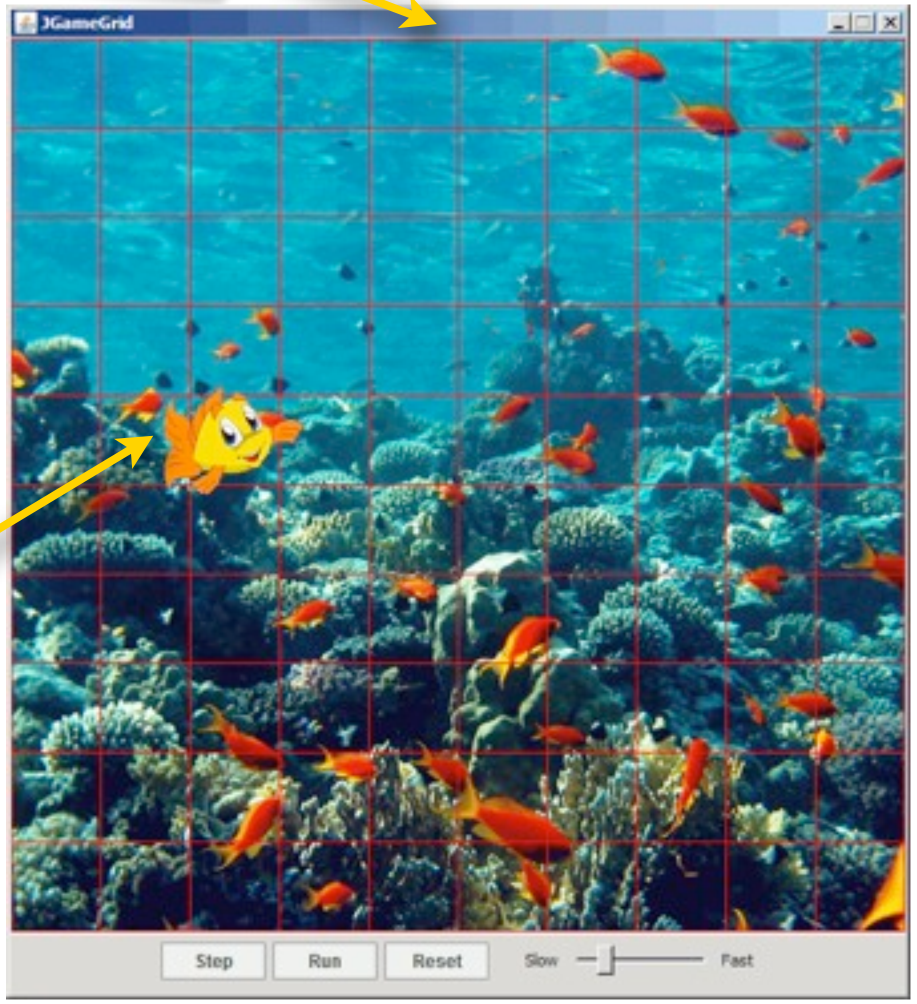
- Full Java syntax and vocabulary
- Limited to static game grids




JGameGrid
Step
Run
Reset
Slow
Fast

```java
import ch.aplu.jgamegrid.*;

public class MyGame extends GameGrid
{
  public MyGame()
  {
    super(10, 10, 60, java.awt.Color.red, "sprites/reef.gif");
    Fish nemo = new Fish();
    addActor(nemo, new Location(2, 4));
    show();
  }

  public static void main(String[] args)
  {
    new MyGame();
  }
}
```

```java
import ch.aplu.jgamegrid.*;

public class Fish extends Actor
{
  public Fish()
  {
    super("sprites/nemo.gif");
  }

  public void act()
  {
    move();
    if (!isMoveValid())
      turn(180);
  }
}
```



Scratch

A graphical programming language including an IDE. Applications are developed (assembled) in a puzzle-like manner.

Strengths

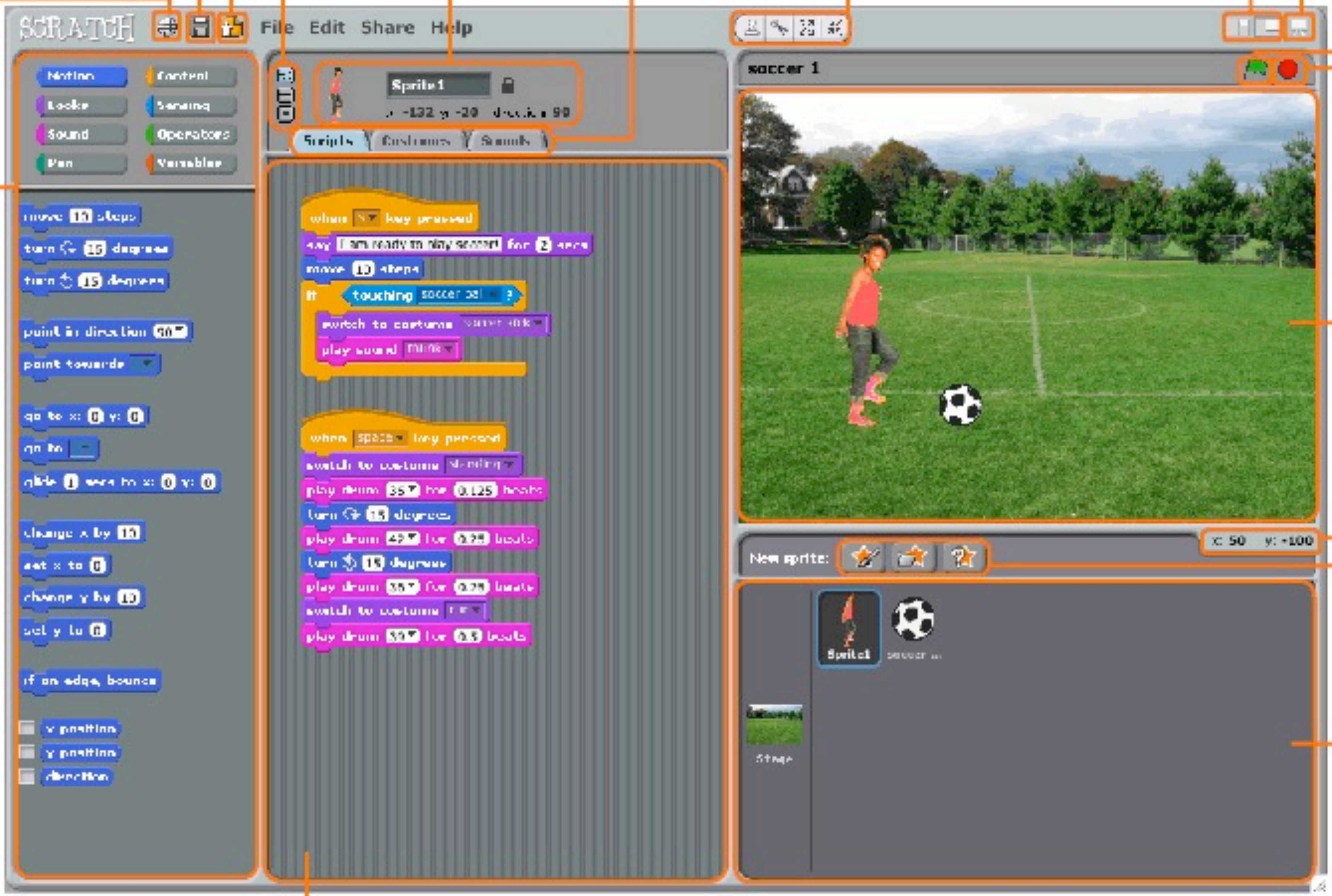
- Easy and intuitive access to programming
- Reduce complexity, e.g., no need to deal with syntax
- Extremely motivating, i.e., cool results within short time

Weaknesses/Limitations

- It is limited to a small domain of applications
- Switching from scratch to a ‘real’ language is difficult for students

http://scratch.mit.edu

SCRATCH 1.4 INTERFACE



```
when [illegible] key pressed
say [I am ready to play soccer!] for (2) secs
move (10) steps
if <touching [soccer ball]?>
    switch to costume [illegible]
    play sound [illegible]

when [space] key pressed
switch to costume [standing]
play drum (35) for (0.125) beats
turn (15) degrees
play drum (42) for (0.25) beats
turn (15) degrees
play drum (35) for (0.25) beats
switch to costume [illegible]
play drum (35) for (0.5) beats
```

SHARE

SAVE

LANGUAGE

SPRITE ROTATION STYLE

CURRENT SPRITE INFO

TABS
Edit scripts, costumes, or sounds.

TOOLBAR

VIEW MODE
Change to large or small stage view.

PRESENTATION MODE
Present your project.

GREEN FLAG
A way to start scripts.

STOP SIGN
Stops all scripts.

BLOCKS PALETTE
Blocks for programming your sprites.

STAGE
Where your Scratch creations come to life.

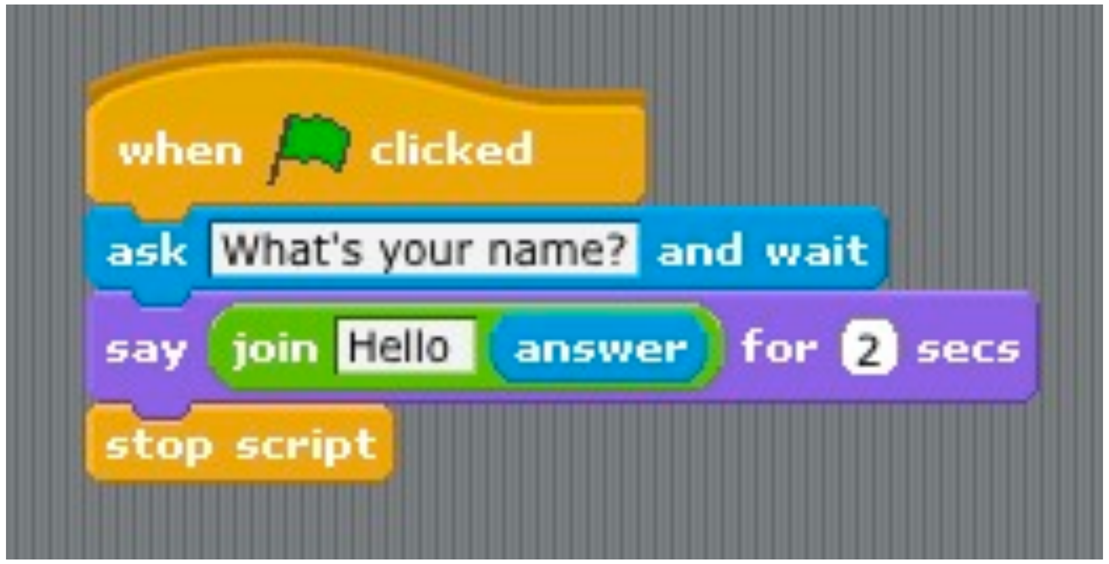
MOUSE X-Y DISPLAY
Shows location of cursor.

NEW SPRITE BUTTONS
Create a new character or object for your project.

SPRITE LIST
Thumbnails of all your sprites. Click to select and edit a sprite.

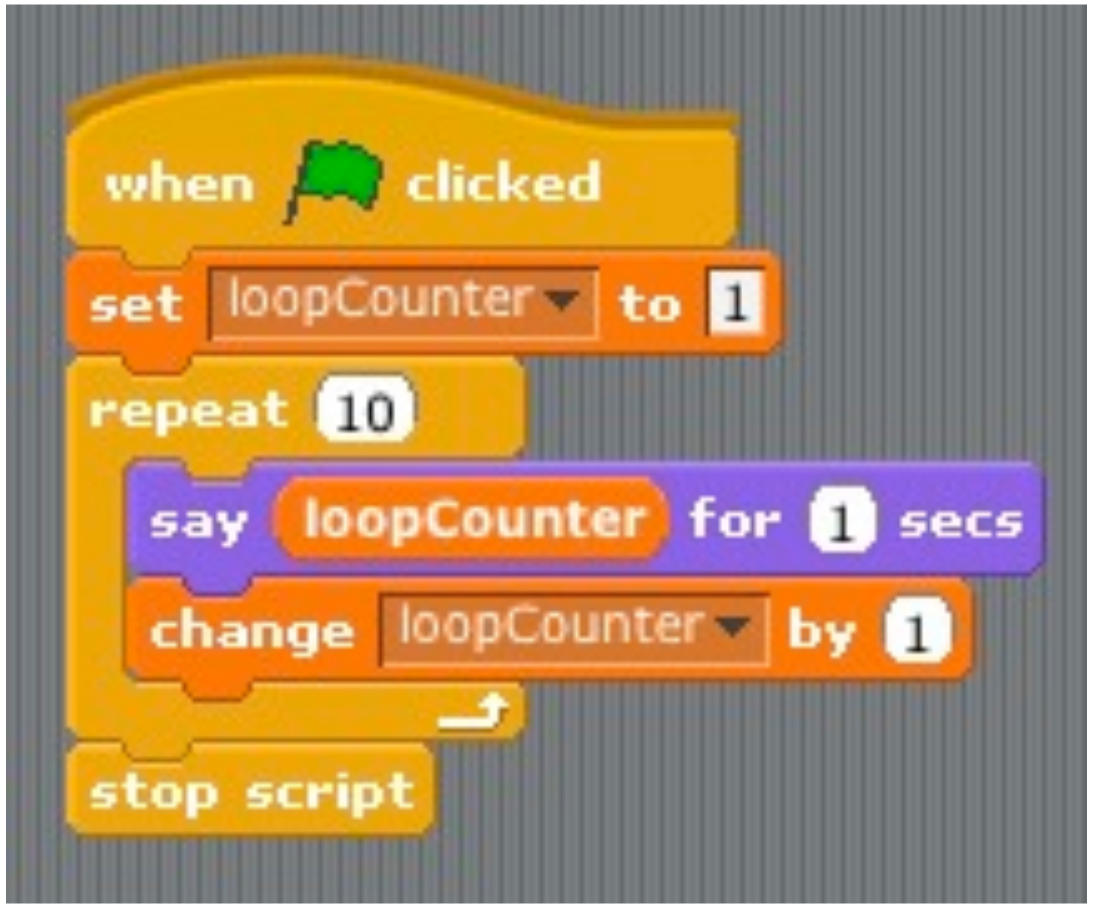
SCRIPTS AREA
Drag blocks in, snap them together into scripts.

Reading any input from the keyboard and display it



```
when [flag] clicked
ask [What's your name?] and wait
say (join [Hello] (answer)) for (2) secs
stop script
```

Display all numbers from 1 to 10 using a loop



```
when [flag] clicked
set [loopCounter] to [1]
repeat (10)
    say (loopCounter) for (1) secs
    change [loopCounter] by (1)
stop script
```

http://scratch.mit.edu

StarLogo TNG

The Next Generation of StarLogo is an environment for exploring the workings of decentralized systems. For instance, you can model traffic jams, ant colonies, and market economies.

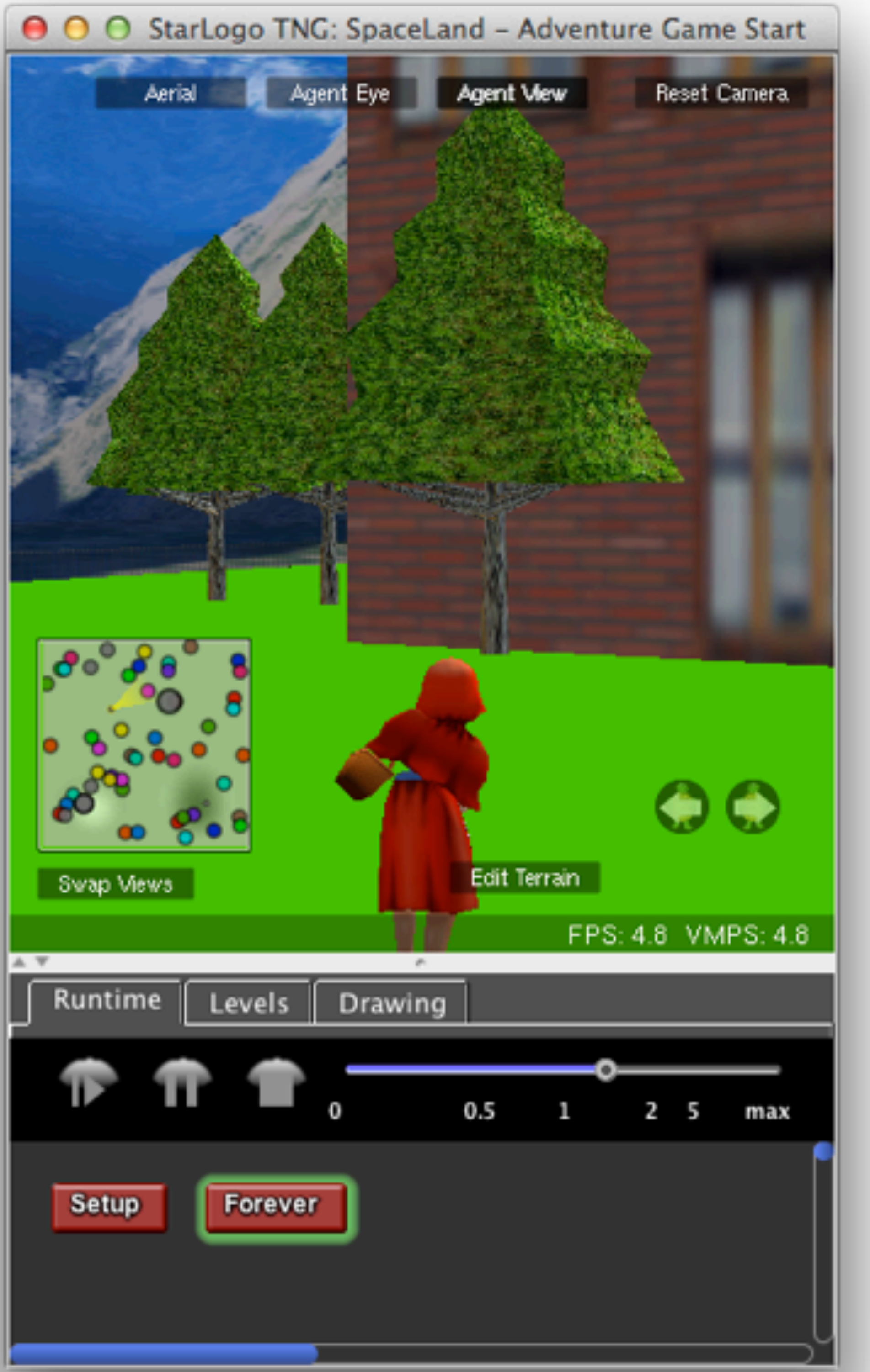
Strengths

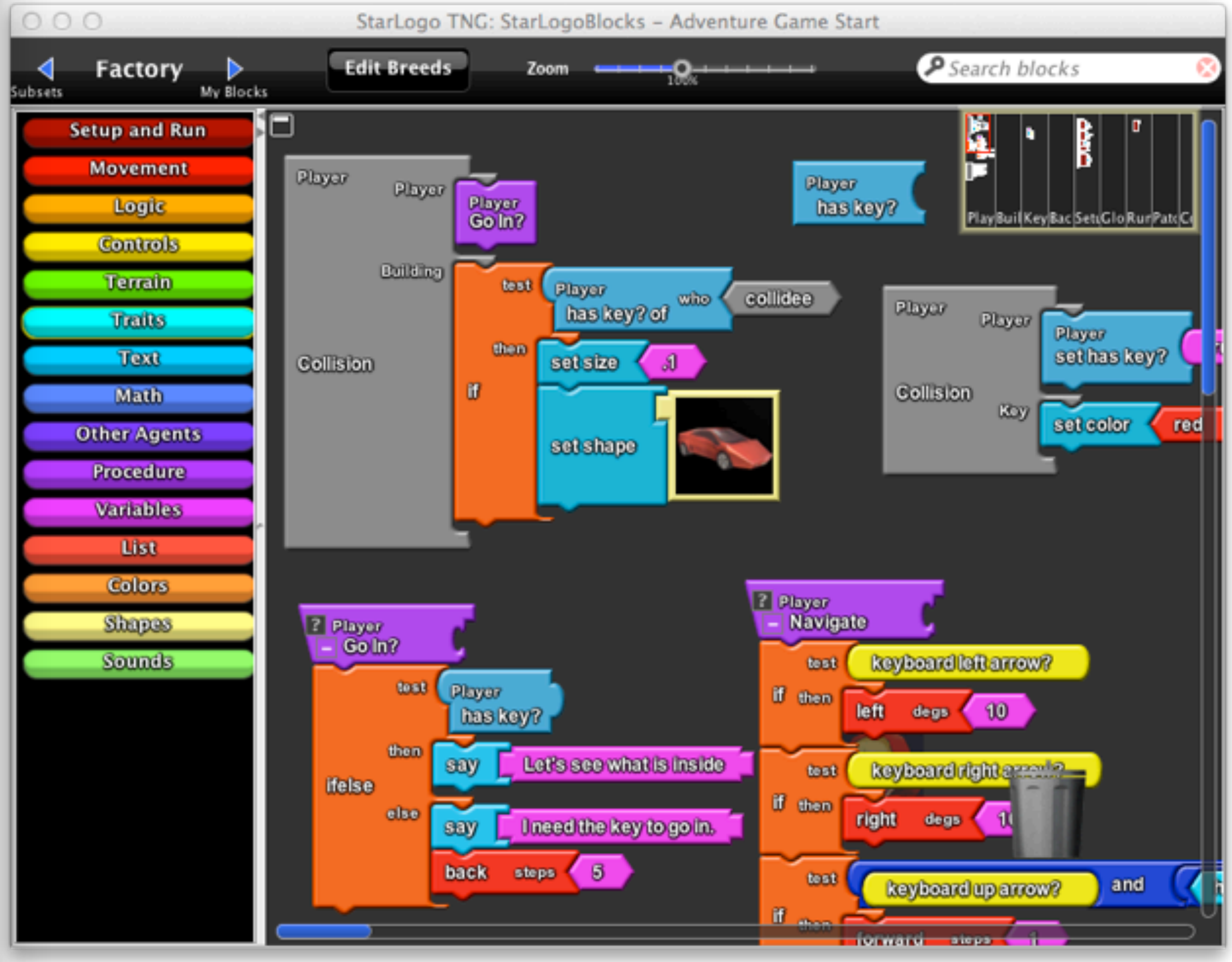
- Derived from Scratch
- Experience the behavior of real-life scenarios, e.g., autonomous agents

Weaknesses/Limitations

- It is limited to a small domain of applications
- Difficult for teaching the concept of OO to students
- Resource hog

http://education.mit.edu/projects/starlogo-tng


StarLogo TNG: SpaceLand – Adventure Game Start
Aerial
Agent Eye
Agent View
Reset Camera
Swap Views
Edit Terrain
FPS: 4.8 VMPS: 4.8
Runtime
Levels
Drawing
0
0.5
1
2
5
max
Setup
Forever


StarLogo TNG: StarLogoBlocks – Adventure Game Start
Subsets
Factory
My Blocks
Edit Breeds
Zoom
Search blocks
Setup and Run
Movement
Logic
Controls
Terrain
Traits
Text
Math
Other Agents
Procedure
Variables
List
Colors
Shapes
Sounds
Collision
Player
Building
Player Go In?
test
Player has key? of
who
collidee
if
then
set size
.1
set shape
Player has key?
Collision
Player
Key
Player set has key?
set color
red
Player Go In?
ifelse
test
Player has key?
then
say
Let's see what is inside
else
say
I need the key to go in.
back
steps
5
Player Navigate
if
test
keyboard left arrow?
then
left
degs
10
keyboard right arrow?
right
keyboard up arrow?
and
forward
steps
1

BlueJ

BlueJ is an integrated development environment (IDE) for the Java programming language. It's main goal is to teach OO concepts.

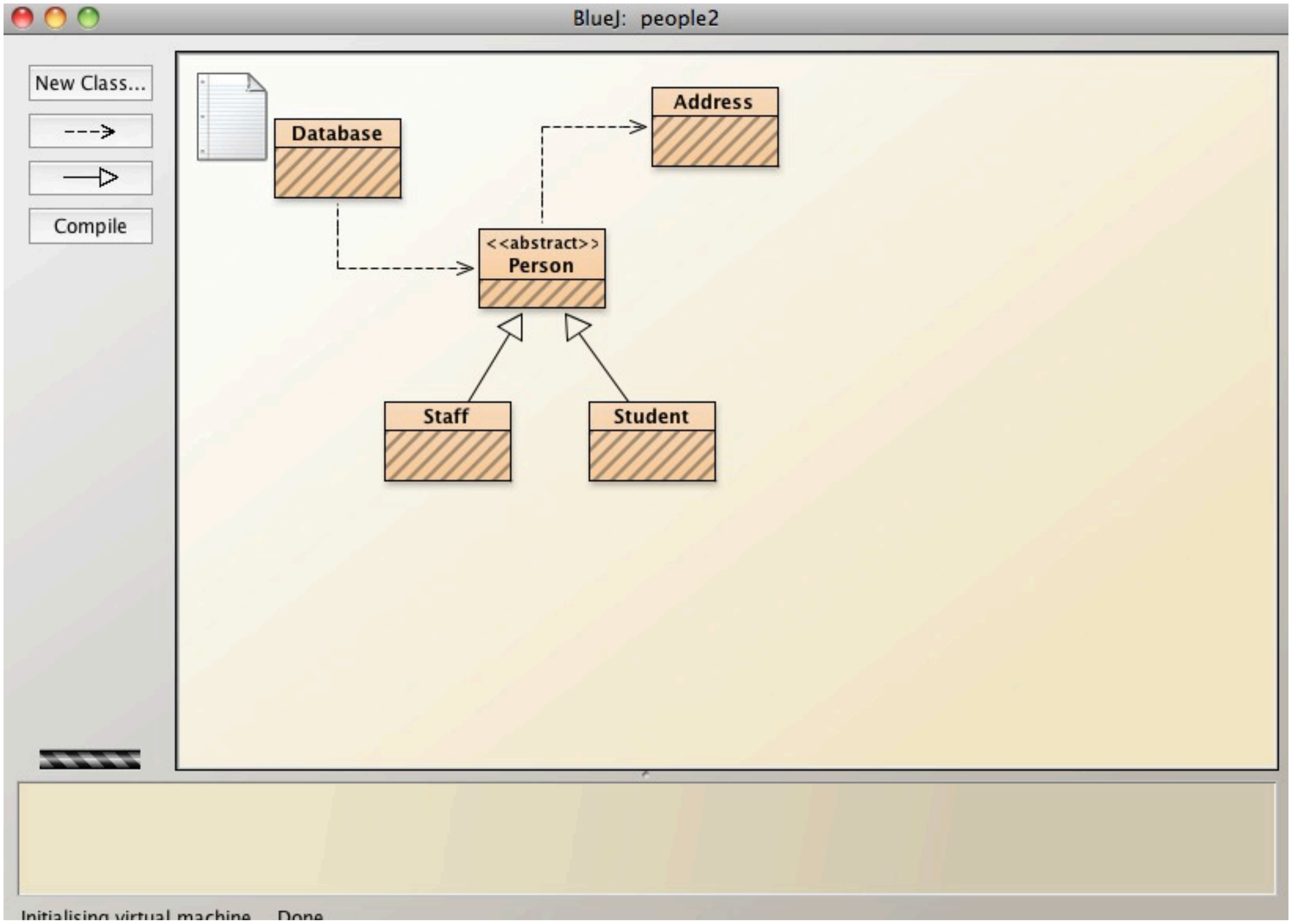
Strengths

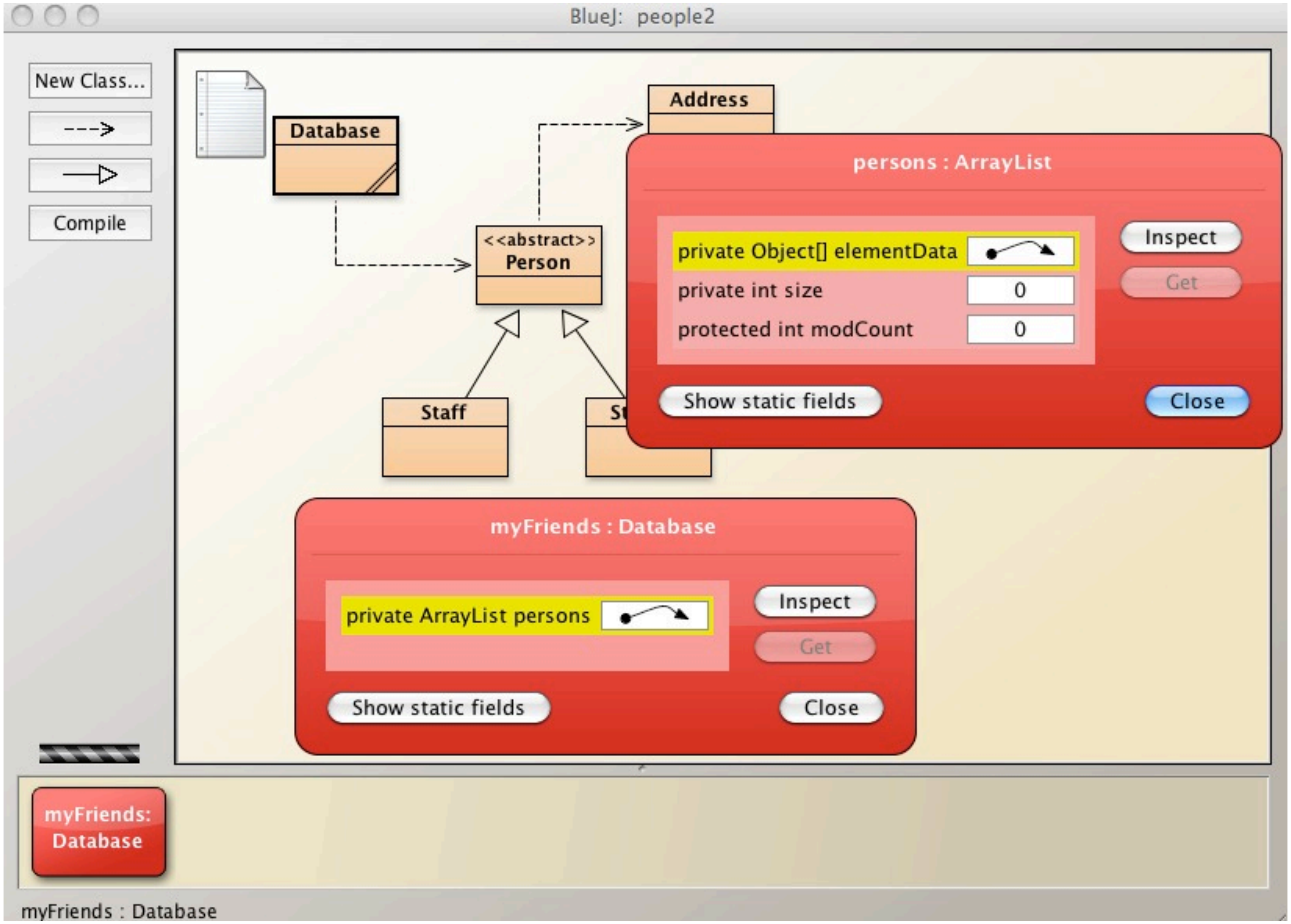
- Intuitive representation of OO (simplified UML diagram)
- OO and code are directly linked
- Runtime workbench for rapid prototyping and debugging, e.g., create instances via context menu

Weaknesses/Limitations

- It's an IDE but not as rich as, for example, Eclipse.

http://bluej.org/


BlueJ: people2
New Class...
Compile
Database
Address
<<abstract>>
Person
Staff
Student
Initialising virtual machine... Done.



persons : ArrayList

private Object[] elementData	
private int size	0
protected int modCount	0

myFriends : Database

private ArrayList persons	

myFriends : Database

Processing

Environment for creating images, animations, and interactions. Initially developed to serve as a software sketchbook and to teach fundamentals of computer programming within a visual context

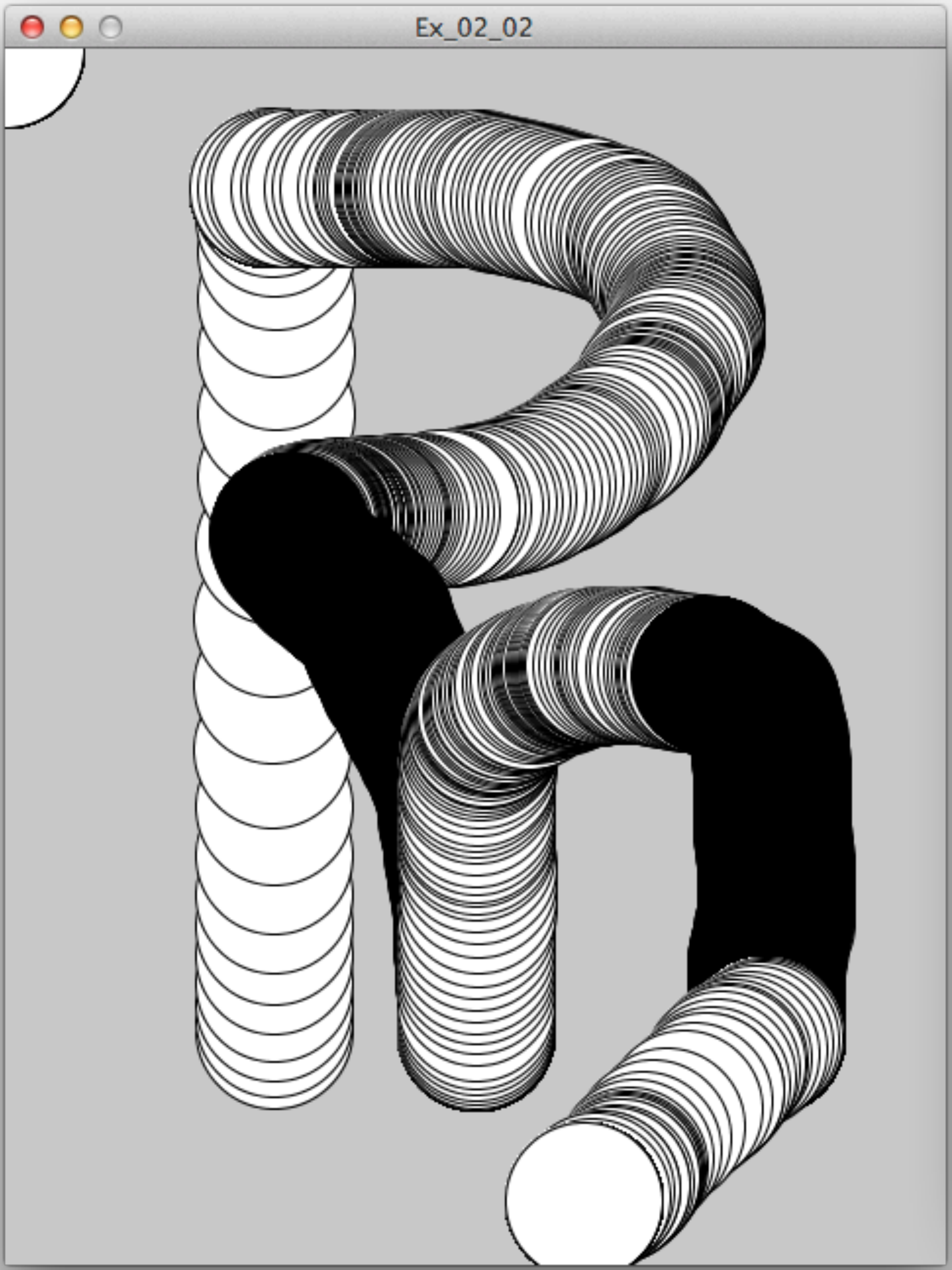
Strengths

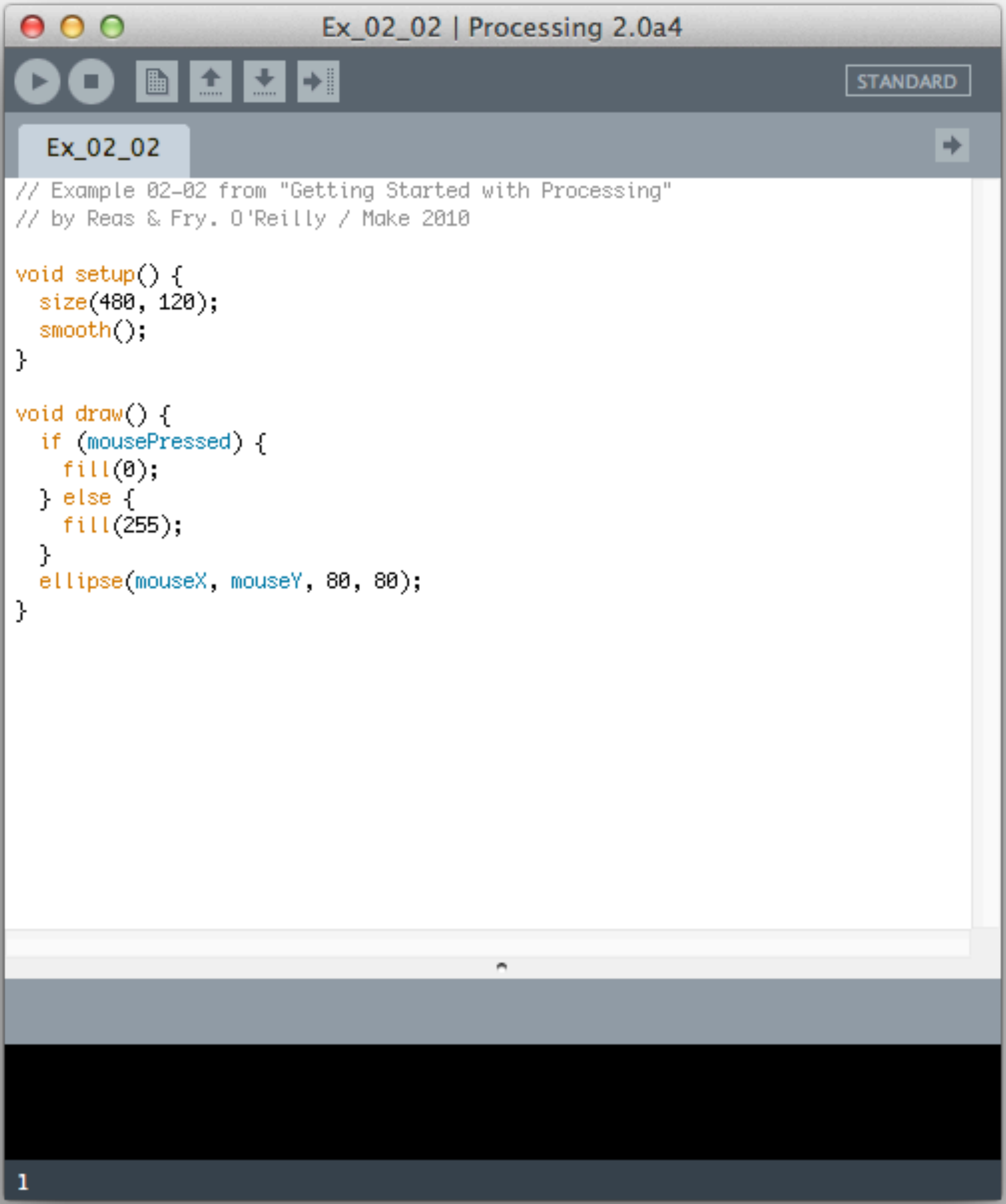
- Visual and interactive programs using 2D, 3D, or PDF output
- Sketches run online or can be easily exported as double-clickable apps
- Fair amount of documentation and tutorials

Weaknesses/Limitations

- Full Java syntax

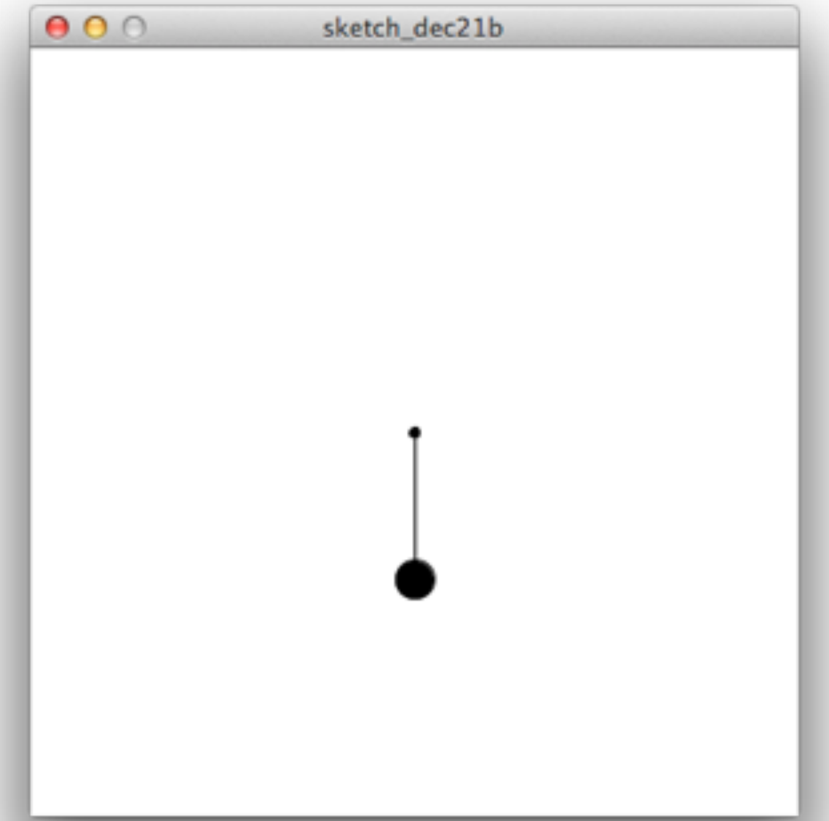
http://processing.org/





```
// Example 02-02 from "Getting Started with Processing"
// by Reas & Fry. O'Reilly / Make 2010

void setup() {
  size(480, 120);
  smooth();
}

void draw() {
  if (mousePressed) {
    fill(0);
  } else {
    fill(255);
  }
  ellipse(mouseX, mouseY, 80, 80);
}
```

```
import traer.physics.*;

ParticleSystem physics;

Particle p;
Particle anchor;
Spring s;

void setup()
{
  size( 400, 400 );
  smooth();
  fill( 0 );
  ellipseMode( CENTER );

  physics = new ParticleSystem( 1, 0.05 );

  p = physics.makeParticle( 1.0, width/2, height/2, 0 );
  anchor = physics.makeParticle( 1.0, width/2, height/2, 0 );
  anchor.makeFixed();
  s = physics.makeSpring( p, anchor, 0.5, 0.1, 75 );
}

void mousePressed()
{
   p.makeFixed();
   p.position().set( mouseX, mouseY, 0 );
}

void mouseDragged()
{
  p.position().set( mouseX, mouseY, 0 );
}

void mouseReleased()
{
   p.makeFree();
}

void draw()
{
  physics.tick();

  background( 255 );

  line( p.position().x(), p.position().y(), anchor.position().x(), anchor.position().y() );
  ellipse( anchor.position().x(), anchor.position().y(), 5, 5 );
  ellipse( p.position().x(), p.position().y(), 20, 20 );
}
```



DrRacket (DrScheme)

A functional programming language based on Lisp/Scheme

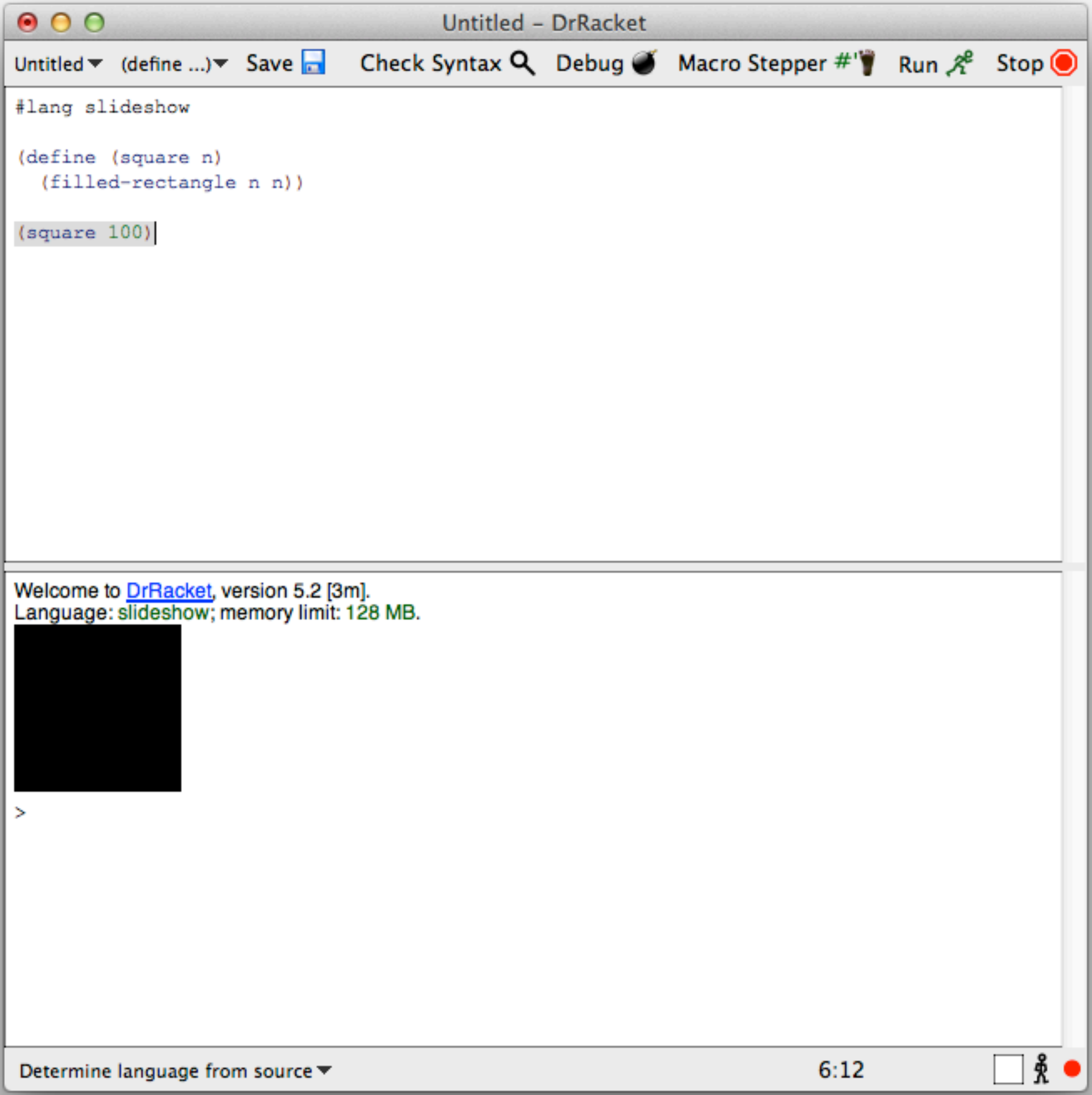
Strengths

- Simple syntax
- Interactive scripting mode
- Especially suited for teaching (tail-)recursion

Weaknesses/Limitations

- Functional programming vs. OOP
- Hard to get (appealing) visual results

http://racket-lang.org



```
#lang slideshow

(define (square n)
  (filled-rectangle n n))

(square 100)
```

```
Welcome to DrRacket, version 5.2 [3m].
Language: slideshow; memory limit: 128 MB.
>
```

Groovy

An agile and dynamic language for the Java Virtual Machine that builds upon the strengths of Java but has additional power features inspired by languages like Python, Ruby and Smalltalk

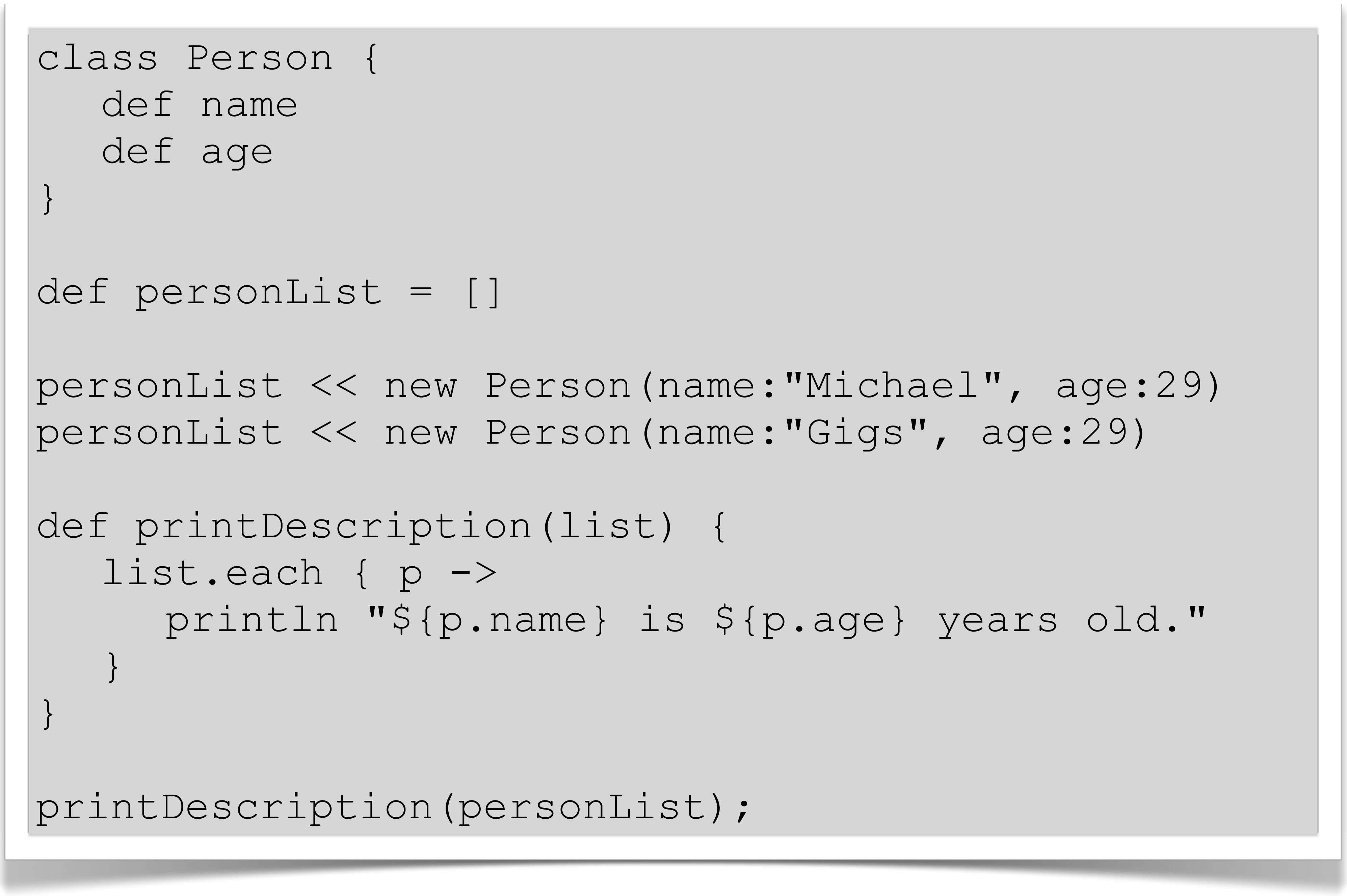
Strengths

- Simplified Java syntax
- Interactive scripting mode vs. full applications

Weaknesses/Limitations

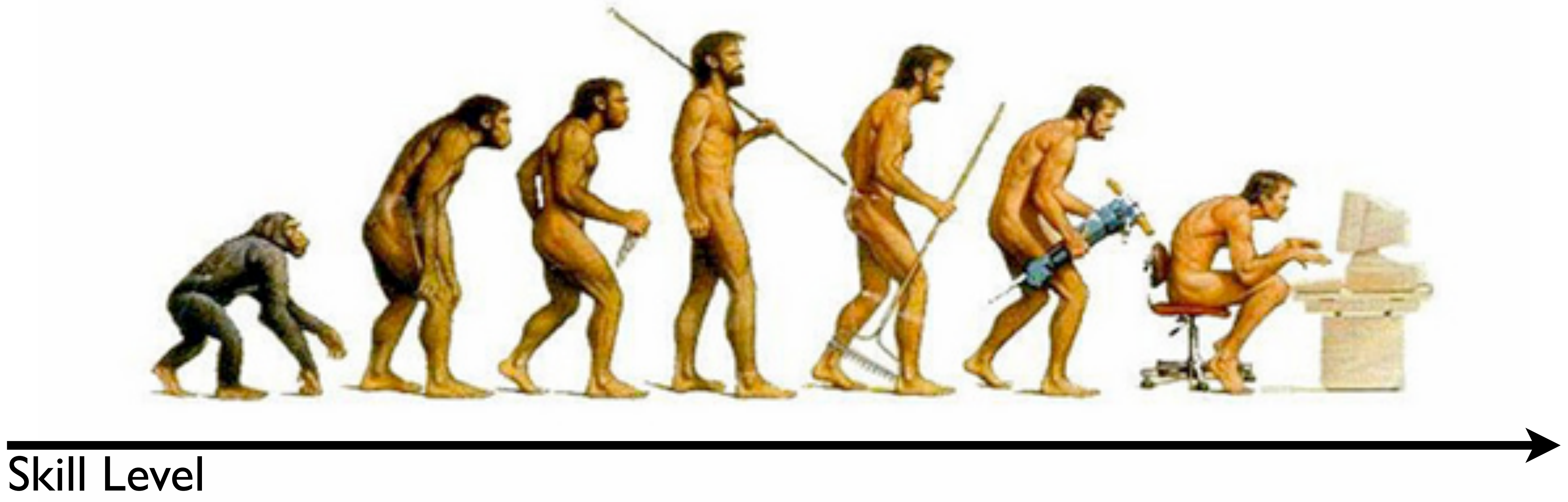
- Hard to produce visual results
- Still incorporates a lot of syntax to learn
- Compiler error messages are confusing for beginners

http://groovy.codehaus.org/



```
class Person {
   def name
   def age
}

def personList = []

personList << new Person(name:"Michael", age:29)
personList << new Person(name:"Gigs", age:29)

def printDescription(list) {
   list.each { p ->
      println "${p.name} is ${p.age} years old."
   }
}

printDescription(personList);
```

Conclusion



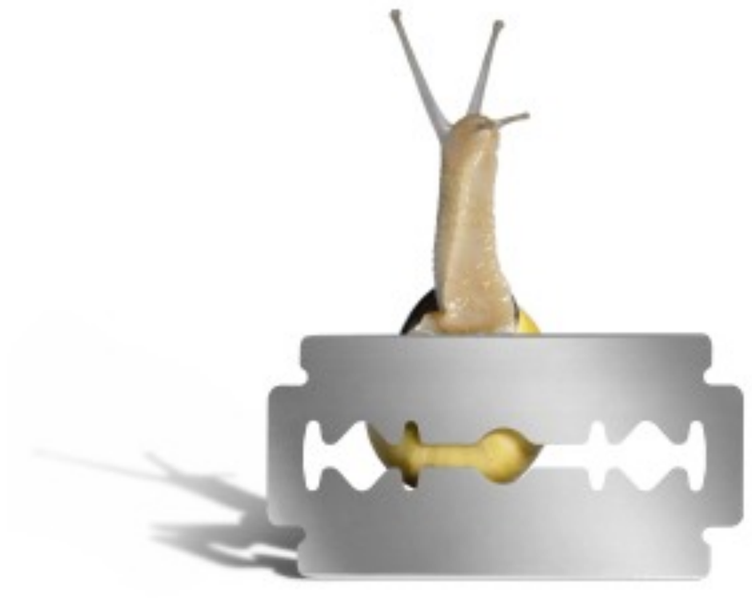
Discussion

Academic Value ⇔ Relevancy to Practice

Is Java the right language for teaching Programming?

Simplicity ⇔ Appealing Presentation

It is all about keeping our pupils/students motivated



Focus ⇔ Breadth

In products, concepts, languages, environments, methods, etc.

Objects-first ⇔ Algorithms/Imperative Programming

Thinking in Objects is (surprisingly) difficult