



Assignment 8

Felix Friedrich, Lars Widmer
TA lecture, Informatics II D-BAUG
April 9, 2014

Concerning the Exam

Assignments Relevance

- The topics from the assignments contribute for **at least 50%** of the exam.
- Using just the mandatory part of the assignments, you can already reach a mark of 4.0 or higher. So even, if you only do the mandatory exercises, you can pass this part of the exam.
- The not mandatory parts of the assignments rather cover the marks above 4.0 up to 6.0. Thus the more not mandatory questions you solve, the better your grade gets.

Concerning the Exam

Not mandatory Questions

- For the rest of the semester there will be less not mandatory questions.
- The aim of the not mandatory parts was to account for differences among the students. Such that if you're not so much into programming yet, you can omit them until your skills have improved. They are clearly meant as an offer not as a burden.
- Personally I like not mandatory stuff because it's usually fun. Stuff "I really have to do" is much more difficult to enjoy.

Concerning the Exam

Summaries

We will provide the summaries Oskar has written on the website after the easter break.

Exam Questions

As soon as possible, long before the exam we will provide you with a set of example questions which could be part of the final exam.

“Präsenzstunden” Today

In the **standard** room

- **HIL E15.2**
- **15:00 - 18:00**
- **Timon Gehr** (*arriving 15:45*)
- **Lei Zhong** (*arriving 15:00*)

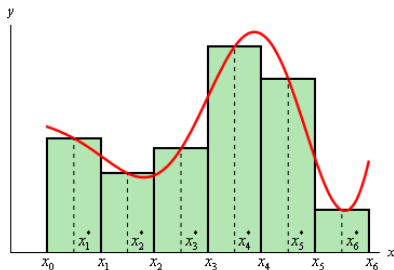
Lei joins Timon

Of course Lei keeps on correcting the hand-ins of his group but he won't hold his own TA lecture in future. Instead he joins Timon for the presence hours.

Outline

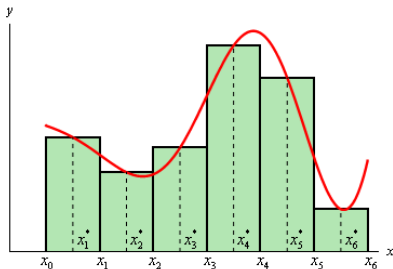
- 1 Know How
 - Rectangle Rule
 - Trapezoidal Rule
- 2 Prediscussion Assignment 8
 - Matlab Functions
 - Matlab Vectors (Arrays)
 - Time Measurement
- 3 Postdiscussion Assignment 7
 - Vehicles Diagram
 - Figures Class Structure
 - Painting Figures

Rectangle Rule



Source: <http://tutorial.math.lamar.edu>

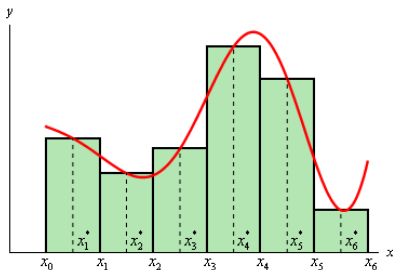
Rectangle Rule



Source: <http://tutorial.math.lamar.edu>

We divide the interval into a given number of slices. Of each slice we take the function value at the center as the height of the according **rectangle**. The sum of the areas of all rectangles is our **approximation** of the integral.

Rectangle Rule



Source: <http://tutorial.math.lamar.edu>

We divide the interval into a given number of slices. Of each slice we take the function value at the center as the height of the according **rectangle**. The sum of the areas of all rectangles is our **approximation** of the integral.

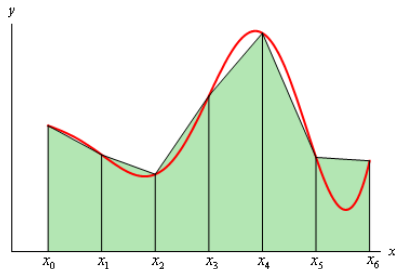
Explanations and the code are in the lecture slides. You can look at the code, when you're stuck while programming.

Otherwise try to implement it from the top of your head.

Outline

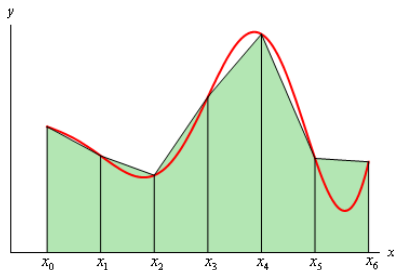
- 1 Know How
 - Rectangle Rule
 - Trapezoidal Rule
- 2 Prediscussion Assignment 8
 - Matlab Functions
 - Matlab Vectors (Arrays)
 - Time Measurement
- 3 Postdiscussion Assignment 7
 - Vehicles Diagram
 - Figures Class Structure
 - Painting Figures

Trapezoidal Rule



Source: <http://tutorial.math.lamar.edu>

Trapezoidal Rule



Source: <http://tutorial.math.lamar.edu>

Same as before, we divide the interval into a given number of slices. But now we take the function values at the beginning and at the end of the interval. Using the two heights we calculate the area of the according **trapezoid**.

The sum of the areas of all trapezoids is our approximation of the integral. Again try to implement this method from the top of your head.

Outline

- 1 Know How
 - Rectangle Rule
 - Trapezoidal Rule
- 2 Prediscussion Assignment 8
 - Matlab Functions
 - Matlab Vectors (Arrays)
 - Time Measurement
- 3 Postdiscussion Assignment 7
 - Vehicles Diagram
 - Figures Class Structure
 - Painting Figures

Function in Matlab

The Code for the function:

$$f(x) = \frac{\sqrt{x^6 - x^3 + 12}}{\ln((x^2 + 4) \cdot (x^2 - 2) + 10)} \quad (1)$$

Function in Matlab

The Code for the function:

$$f(x) = \frac{\sqrt{x^6 - x^3 + 12}}{\ln((x^2 + 4) \cdot (x^2 - 2) + 10)} \quad (1)$$

... looks like this in Matlab:

```
1 function res = f(x)
2     res = sqrt(x^6-x^3+12) / log((x^2+4)*(x^2-2)+10);
3 end
```

Matlab Function Definitions

```
1 function res = f(x)
2     res = sqrt(x^6-x^3+12) / log((x^2+4)*(x^2-2)+10);
3 end
```

Names:

- “res” is ...

Matlab Function Definitions

```
1 function res = f(x)
2     res = sqrt(x^6-x^3+12) / log((x^2+4)*(x^2-2)+10);
3 end
```

Names:

- “res” is ... the name we gave to the return value.
- “f” is ...

Matlab Function Definitions

```
1 function res = f(x)
2     res = sqrt(x^6-x^3+12) / log((x^2+4)*(x^2-2)+10);
3 end
```

Names:

- “res” is ... the name we gave to the return value.
- “f” is ... the function name.
This means we **have to save the file** as “f.m”.
- “x” is ...

Matlab Function Definitions

```
1 function res = f(x)
2     res = sqrt(x^6-x^3+12) / log((x^2+4)*(x^2-2)+10);
3 end
```

Names:

- “res” is ... the name we gave to the return value.
- “f” is ... the function name.
This means we **have to save the file** as “f.m”.
- “x” is ... the parameter name.
It's the input value we pass to the function.

Matlab Function Usage

```
1 function res = f(x)
2     res = sqrt(x^6-x^3+12) / log((x^2+4)*(x^2-2)+10);
3 end
```

We call this method in the **Command Window** like this:

```
1 >> f(100)
2 ans =
3     5.4286e+04
```

Matlab Function Usage

```
1 function res = f(x)
2     res = sqrt(x^6-x^3+12) / log((x^2+4)*(x^2-2)+10);
3 end
```

We call this method in the **Command Window** like this:

```
1 >> f(100)
2 ans =
3     5.4286e+04
```

With a **semicolon** after the call, the result isn't printed:

```
1 >> f(100);
2 >>
```

Outline

- 1 Know How
 - Rectangle Rule
 - Trapezoidal Rule
- 2 Prediscussion Assignment 8
 - Matlab Functions
 - Matlab Vectors (Arrays)
 - Time Measurement
- 3 Postdiscussion Assignment 7
 - Vehicles Diagram
 - Figures Class Structure
 - Painting Figures

Matlab Vectors

A vector in Matlab is like an array in Java. Nevertheless the brackets differ a bit.

Creating a vector:

```
1 >> test = [1 2 42 7]
2 test =
3      1      2     42      7
```

Matlab Vectors

A vector in Matlab is like an array in Java. Nevertheless the brackets differ a bit.

Creating a vector:

```
1 >> test = [1 2 42 7]
2 test =
3      1      2     42      7
```

Using a vector:

```
1 >> test(3)
2 ans =
3     42
4 >>
```


Matlab Vectors

Using a vector:

```
1 >> test(3)
2 ans =
3     42
4 >>
```

Indexing

Be aware that arrays in Matlab are indexed starting with 1 and not 0, as in Java. So the first element of vector `vec` in Matlab is element `vec(1)` $\sin(!)$

Outline

- 1 Know How
 - Rectangle Rule
 - Trapezoidal Rule
- 2 Prediscussion Assignment 8
 - Matlab Functions
 - Matlab Vectors (Arrays)
 - Time Measurement
- 3 Postdiscussion Assignment 7
 - Vehicles Diagram
 - Figures Class Structure
 - Painting Figures

Measuring Execution Time

It's quite simple:

```
1 tic  
2 f(42);  
3 toc
```

Example on one line:

```
1 >> tic; f(42); toc  
2 Elapsed time is 0.000079 seconds.
```

Measuring Execution Time

Example on one line:

```
1 >> tic; f(42); toc
2 Elapsed time is 0.000079 seconds.
```

And don't forget: **Output is slow!**

```
1 >> tic; f(42), toc
2 ans =
3     4.9551e+03
4 Elapsed time is 0.000635 seconds.
```

Measuring Execution Time

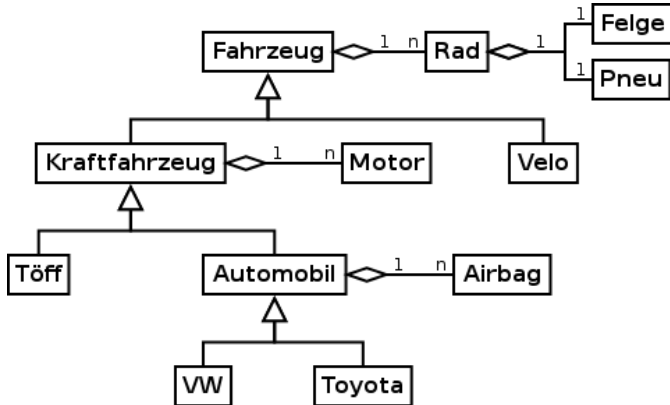
As we know from exercise 4.3 g) the **execution time in Java** is measured this way:

```
1 long time = System.currentTimeMillis();  
2 double res = test.f(42);  
3 long duration = System.currentTimeMillis() - time;
```

Outline

- 1 Know How
 - Rectangle Rule
 - Trapezoidal Rule
- 2 Prediscussion Assignment 8
 - Matlab Functions
 - Matlab Vectors (Arrays)
 - Time Measurement
- 3 Postdiscussion Assignment 7
 - Vehicles Diagram
 - Figures Class Structure
 - Painting Figures

Solution, Vehicles Diagram



Airbag for Motorcycles



Airbag for Motorcycles

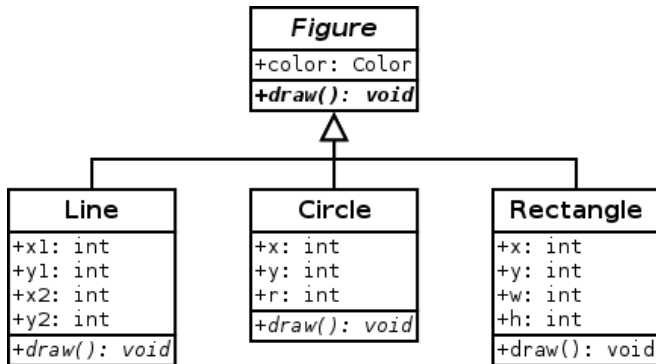


It can be argued, that some of today's motorcycles actually have an airbag. Or an airbag is even contained in the hightech suit for motorcyclists.

Outline

- 1 Know How
 - Rectangle Rule
 - Trapezoidal Rule
- 2 Prediscussion Assignment 8
 - Matlab Functions
 - Matlab Vectors (Arrays)
 - Time Measurement
- 3 Postdiscussion Assignment 7
 - Vehicles Diagram
 - **Figures Class Structure**
 - Painting Figures

Given Class Diagram



Superclass Figure

```
1 class Figure {  
2     Color color;  
3     public Figure(Color c) {  
4         color = c;  
5     }  
6     public void draw() {  
7         System.out.println("Figure_ "+color);  
8     }  
9 }
```

Subclass Line

```
1 class Line extends Figure {
2     int x1, x2, y1, y2;
3     public Line(Color c,
4                 int x1, int y1, int x2, int y2) {
5         super(c); // reuse Figure-constructor
6         this.x1 = x1;
7         this.y1 = y1;
8         this.x2 = x2;
9         this.y2 = y2;
10    }
11    public void draw() {
12        System.out.println("Line " + color + " ("
13                            + x1 + "/" + y1 + ") to (" + x2 + "/" + y2 + ")");
14    } // saving space
```

Subclass Circle

```
1 class Circle extends Figure {
2     int x, y, r;
3     public Circle(Color c,int x,int y,int r) {
4         super(c); // reuse Figure-constructor
5         this.x = x;
6         this.y = y;
7         this.r = r;
8     }
9     public void draw() {
10        System.out.println("Circle_" + color +
11            "_at_(" + x + "/" + y + ")_radius_" + r);
12    }
13 }
```

Subclass Rectangle

```
1 class Rectangle extends Figure {  
2     int x, y, w, h;  
3     public Rectangle(Color c,  
4         int x,int y,int w,int h) {  
5         super(c); // reuse Figure-constructor  
6         this.x = x;  
7         this.y = y;  
8         this.w = w;  
9         this.h = h;  
10    }  
11    public void draw() {  
12        System.out.println("Rectangle "+color  
13            +" at (" +x+ "/" +y+ ") size "+w+"x"+h+ " ");  
14    } }
```

Usage in main

```
1 public static void main(String[] args) {  
2     LinkedList<Figure> figures =  
3         new LinkedList<Figure>();  
4     figures.add(new Figure(Color.PINK));  
5     figures.add(new Line(Color.BLUE,10,10,20,20));  
6     figures.add(new Line(Color.RED,20,10,10,20));  
7     figures.add(new Circle(Color.GREEN,30,30,10));  
8     figures.add(  
9         new Rectangle(Color.YELLOW,20,20,20,10));  
10    for (Figure f : figures) {  
11        f.draw();  
12    }  
13 }
```


Output of main

- 1 Figure [r=255,g=175,b=175]
- 2 Line [r=0,g=0,b=255] from (10/10) to (20/20)
- 3 Line [r=255,g=0,b=0] from (20/10) to (10/20)
- 4 Circle [r=0,g=255,b=0] at (30/30) with radius 10
- 5 Rectangle [r=255,g=255,b=0] at (20/20) size 20x10

Text shrunked: "java.awt.Color" ommitted.

Outline

- 1 Know How
 - Rectangle Rule
 - Trapezoidal Rule
- 2 Prediscussion Assignment 8
 - Matlab Functions
 - Matlab Vectors (Arrays)
 - Time Measurement
- 3 Postdiscussion Assignment 7
 - Vehicles Diagram
 - Figures Class Structure
 - Painting Figures

class Line extends Figure

```
1 private int x1, x2, y1, y2;  
2 public Line(Color c, int x1, int y1, int x2, int y2) {  
3     super(c); // reuse constructor of Figure  
4     this.x1 = x1;  
5     this.x2 = x2;  
6     this.y1 = y1;  
7     this.y2 = y2;  
8 }
```

We have to call the `super`-constructor explicitly (line 3), since `Figure` has no default-constructor (without parameters).

class Line extends Figure

```
1 public void draw(EasyGraphics graph) {  
2     if ((x2-x1) > (y2-y1)) {  
3         for (int i=x1; i<x2; ++i) {  
4             graph.set(i, (i-x1)*(y2-y1)/(x2-x1)+y1,  
5                     color.getRGB());  
6         }  
7     } else {  
8         for (int i=y1; i<y2; ++i) {  
9             graph.set((i-y1)*(x2-x1)/(y2-y1)+x1, i,  
10                    color.getRGB());  
11        }  
12    }  
13    graph.repaint();  
14 }
```

class Circle extends Figure

Class structure omitted, you can find the full solution on the website. The basic concept is to use a *sin*-function for the x-value and a *cos*-function for the y-value.

```
1 public void draw(EasyGraphics graph) {  
2     for (int i=0; i<r*Math.PI*2; ++i) {  
3         graph.set(x+(int)(Math.cos((double)i/r)*r),  
4                 y+(int)(Math.sin((double)i/r)*r),  
5                 color.getRGB());  
6     }  
7     graph.repaint();  
8 }
```

class Circle extends Figure

```
1 graph.set(x+(int)(Math.sin((double)i/r)*r),  
2           y+(int)(Math.cos((double)i/r)*r),  
3           color.getRGB());
```

Common pitfalls are problems with Java's **integer computations** and **type casts**. If something doesn't work as expected, always make sure to consistently work with double-values only. If you need an `int` in the end, cast it just before passing back the value. Only when your algorithm works as you wish, you can finally optimize the use of `double`.

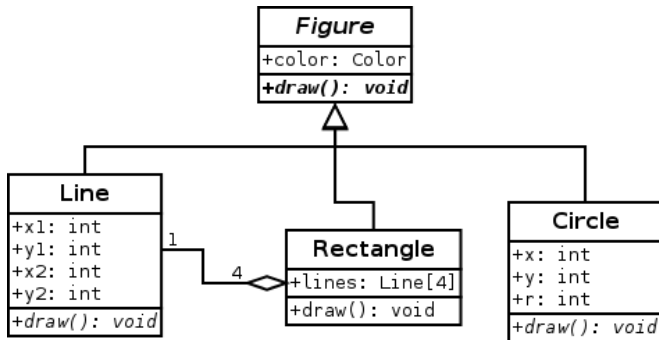
class Rectangle extends Figure

```
1 private Line[] lines; // Rectangle has 4 lines!
2
3 public Rectangle(Color c,
4                 int x,int y,int w,int h) {
5     super(c);
6     lines = new Line[4];
7     lines[0] = new Line(c,x,y,x+w,y);
8     lines[1] = new Line(c,x+w,y,x+w,y+h);
9     lines[2] = new Line(c,x,y+h,x+w,y+h);
10    lines[3] = new Line(c,x,y,x,y+h);
11 }
```

Being lazy, I use four Line objects to form a Rectangle.

Rectangle has Lines

Being lazy, I use four Line-objects to form a Rectangle.
Therefore the class diagram now looks like this:



class Rectangle extends Figure

```
1 public void draw(EasyGraphics graph) {  
2     for (Line l : lines) {  
3         l.draw(graph);  
4     }  
5     graph.repaint();  
6 }
```

Drawing the Rectangle is easy, since we just have to call the draw-methods of the four aggregated Lines.

main-Method

I decided to keep my Figures in a LinkedList.
At first I add 20 Lines, all starting from the same origin.

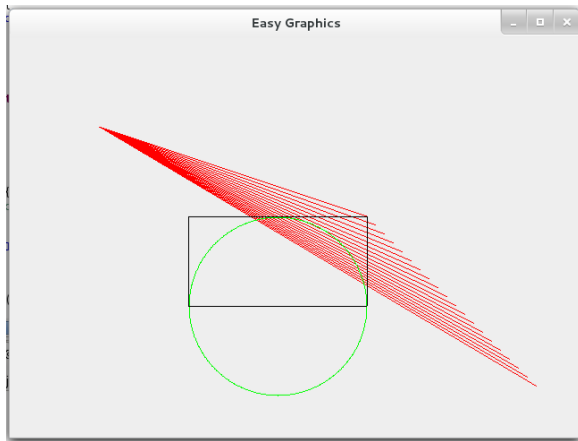
```
1 public static void main(String[] args) {  
2     LinkedList<Figure> figures;  
3     figures = new LinkedList<Figure>();  
4     for (int i=0; i<20; ++i) {  
5         figures.add(new Line(Color.RED,  
6             100,100,400+i*10,200+i*10));  
7     }  
8     ...  
}
```

main Method

Further I add a Circle and a Rectangle.
Finally we call the draw-Methods in a loop.

```
1      ...
2      figures.add(new Circle(Color.GREEN,
3                          300,300,100));
4      figures.add(new Rectangle(Color.BLACK,
5                          200,200,200,100));
6      EasyGraphics graph = new EasyGraphics();
7      for (Figure f : figures) {
8          f.draw(graph);
9      }
10 }
```

And thats how this example looks like



Questions, please

Please

- Feedback?
- Additions?
- Remarks?
- Wishes?
- ...



All the Best!

