Dr. Michael Eichberg
Software Technology Group
Department of Computer Science
Technische Universität Darmstadt

Introduction to Software Engineering

# Object-Oriented Thinking

TECHNISCHE
UNIVERSITÄT
DARMSTADT

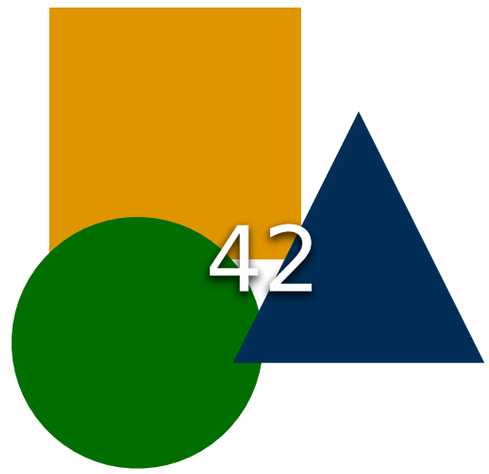
# Object-Oriented Thinking

- Class-Responsibility-Collaboration Cards
  (A very first glimpse of object-oriented analysis and design.)

# A First Glimpse on OO Analysis and Design.



- We want to develop a library for representing vector graphics
- We want to support:
  - Squares
  - Circles
  - Triangles
  - Text




- We want to be able to export figures as PDF and SVG documents

# A Class-Responsibility-Collaboration (CRC) Card



| **Class** | **Collaborations** |
| --- | --- |
| <classname> | ▸ <collaborator A><br>▸ <collaborator B><br>▸ ... |
| **Responsibilities** | |
| ▸ <responsibility A><br>▸ <responsibility B><br>▸ ... | |

other classes

short eng. sentences

# Class-Responsibility-Collaboration (CRC) cards help to discuss object-oriented designs.

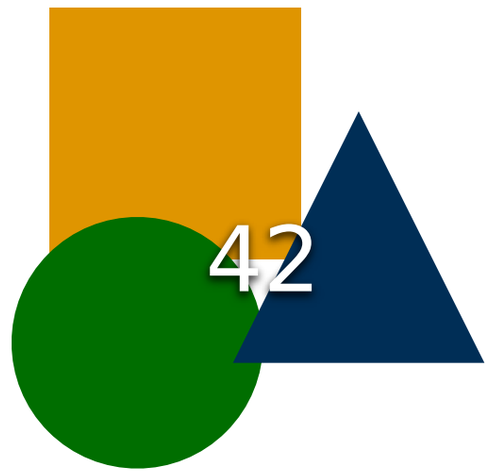


- **The class name of an object creates a vocabulary for discussing the design**; you should spent enough time to find the right words
- **Responsibilities identify problems to be solved**; a responsibility serves as a handle for discussing **potential solutions**; they are expressed using short verb phrases each containing an active verb
- **Collaborators are the objects which will send** or receive **messages** in the course of satisfying responsibilities

# First, we have to identify potential classes.



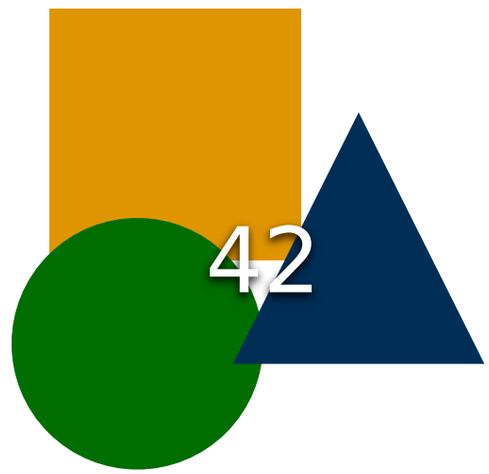
- We want to develop a library for representing vector graphics
- We want to support:
  - Squares
  - Circles
  - Triangles
  - Text
- We want to be able to export figures as PDF and SVG documents



# First, we identify potential classes.



- We want to develop a **library** for representing **vector graphics**
- We want to support:
  - **Square**s
  - **Circle**s
  - **Triangle**s
  - **Text**
- We want to be able to export **figure**s as **PDF** and **SVG *documents***




We start by identifying nouns.

# For each potential class, we create a CRC card.



| Class | Collaborations |
|---|---|
| Text | |
| **Responsibilities** | |

| Class | Collaborations |
|---|---|
| Square | |
| **Responsibilities** | |

| Class | Collaborations |
|---|---|
| Triangle | |
| **Responsibilities** | |

| Class | Collaborations |
|---|---|
| Circle | |
| **Responsibilities** | |

"Setup"

# For each potential class, we create a CRC card.



| Class | Collaborations |
|---|---|
| Figure | |
| **Responsibilities** | |

| Class | Collaborations |
|---|---|
| PDFDocument | |
| **Responsibilities** | |

| Class | Collaborations |
|---|---|
| SVGDocument | |
| **Responsibilities** | |

"Setup"

# We then identify responsibilities and collaborators.



| Class | Collaborations |
|---|---|
| Text | |
| **Responsibilities** | |
| ▸ Maintain the text | |

| Class | Collaborations |
|---|---|
| Square | |
| **Responsibilities** | |
| ▸ Maintain the square's data | |

| Class | Collaborations |
|---|---|
| Triangle | |
| **Responsibilities** | |
| ▸ Maintain the triangle's data | |

| Class | Collaborations |
|---|---|
| Circle | |
| **Responsibilities** | |
| ▸ Maintain the circle's data | |

# We then identify responsibilities and collaborators.



| Class | Collaborations |
|---|---|
| Figure | |
| **Responsibilities** | |
| Maintain a list of "shapes" | |

| Class | Collaborations |
|---|---|
| PDFDocument | |
| **Responsibilities** | |
| Create a ".pdf" file | |

| Class | Collaborations |
|---|---|
| SVGDocument | |
| **Responsibilities** | |
| Create a ".svg" file | |

# We incrementally refine our cards.



| Class | Collaborations |
|---|---|
| Figure | |
| **Responsibilities** | |
| Maintain a list of "shapes" | |

| Class | Collaborations |
|---|---|
| PDFDocument | |
| **Responsibilities** | |
| ~~Create ...~~ | |
| Save a figure to a ".pdf" file | |

| Class | Collaborations |
|---|---|
| SVGDocument | |
| **Responsibilities** | |
| ~~Create ...~~ | |
| Save a figure to a ".svg" file | |

# We incrementally refine our cards.



| Class | Collaborations |
|---|---|
| Figure | |
| **Responsibilities** | |
| Maintain a list of "shapes" | |

| Class | Collaborations |
|---|---|
| ~~PDFDocument~~ | |
| PDFDocBuilder | |
| **Responsibilities** | |
| Save a figure to a ".pdf" file | |

| Class | Collaborations |
|---|---|
| ~~SVGDocument~~ | |
| SVGDocBuilder | |
| **Responsibilities** | |
| Save a figure to a ".svg" file | |

# We then identify responsibilities and collaborators.



| Class | Collaborations |
|---|---|
| Figure | Text<br>Circle<br>Triangle<br>Square |
| **Responsibilities** | |
| Maintain a list of "shapes" | |

How do we support this?

| Class | Collaborations |
|---|---|
| PDFDocBuilder | Figure<br>Text<br>Circle<br>Triangle<br>Square |
| **Responsibilities** | |
| Save a figure to a ".pdf" file | |

| Class | Collaborations |
|---|---|
| SVGDocBuilder | Figure<br>Text<br>Circle<br>Triangle<br>Square |
| **Responsibilities** | |
| Save a figure to a ".svg" file | |

# We then identify responsibilities and collaborators.



| Class | Collaborations |
|---|---|
| Text | |
| **Responsibilities** | |
| ▸ Maintain the text | |

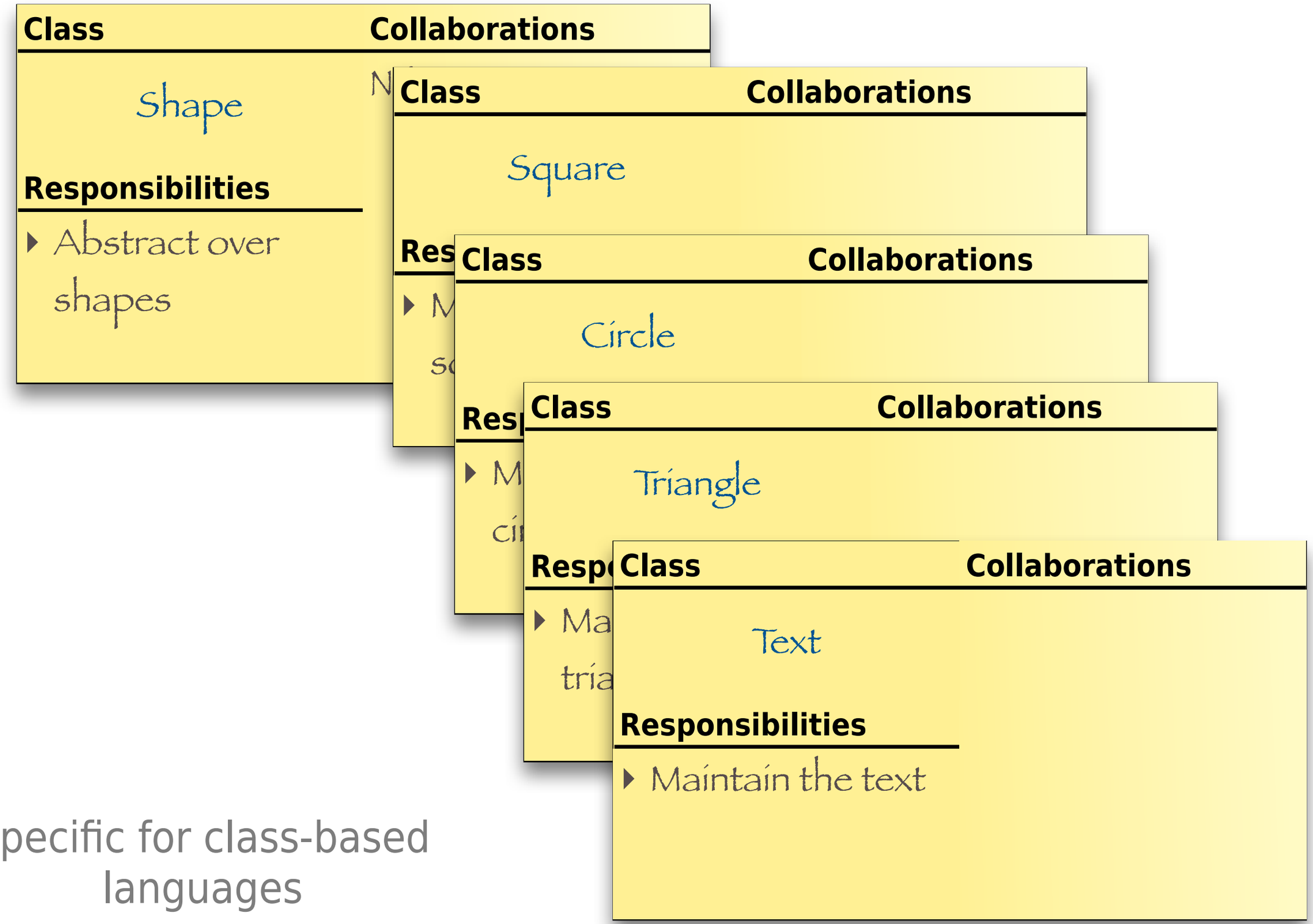
| Class | Collaborations |
|---|---|
| Square | |
| **Responsibilities** | |
| ▸ Maintain the square's data | |

| Class | Collaborations |
|---|---|
| Triangle | |
| **Responsibilities** | |
| ▸ Maintain the triangle's data | |

| Class | Collaborations |
|---|---|
| Circle | |
| **Responsibilities** | |
| ▸ Maintain the circle's data | |

# We then identify responsibilities and collaborators.






Specific for class-based languages

# We then identify responsibilities and collaborators.



**Class** Shape

**Collaborations**

**Responsibilities**

- Abstract over shapes

**Class** Square

**Collaborations**

- Color

**Responsibilities**

- Maintain a square

**Class** Circle

**Collaborations**

- Color

**Responsibilities**

- Maintain a circle

**Class** Triangle

**Collaborations**

- Color

**Responsibilities**

- Maintain a triangle

**Class** Text

**Collaborations**

- Color

**Responsibilities**

- Maintain the text

**Class** Color

**Collaborations**

N/A

**Responsibilities**

- Converts between color spaces

> *We stress the importance of creating objects not to meet mythical future needs, but only under the demands of the moment.*

Kent Beck, Ward Cunningham

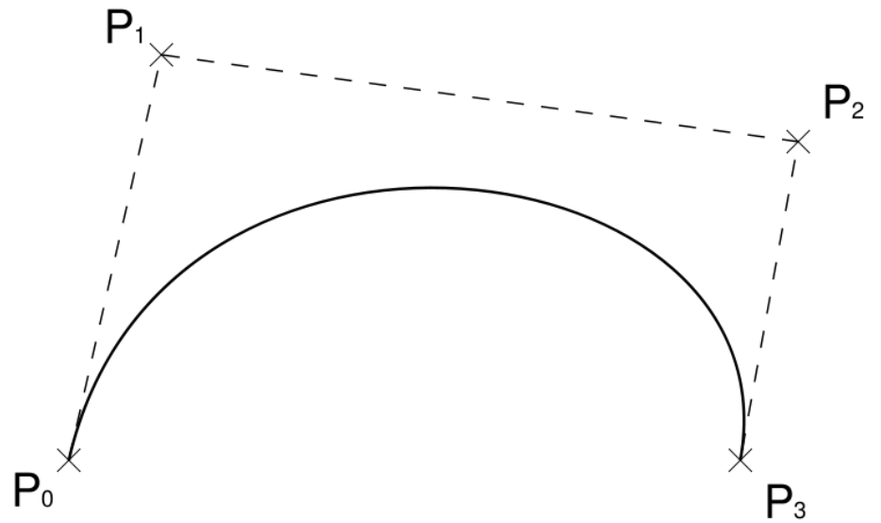
*A Laboratory For Teaching OO-Thinking*

*Proceedings of OOPSLA ’89; ACM Press*

# Evolving the Application



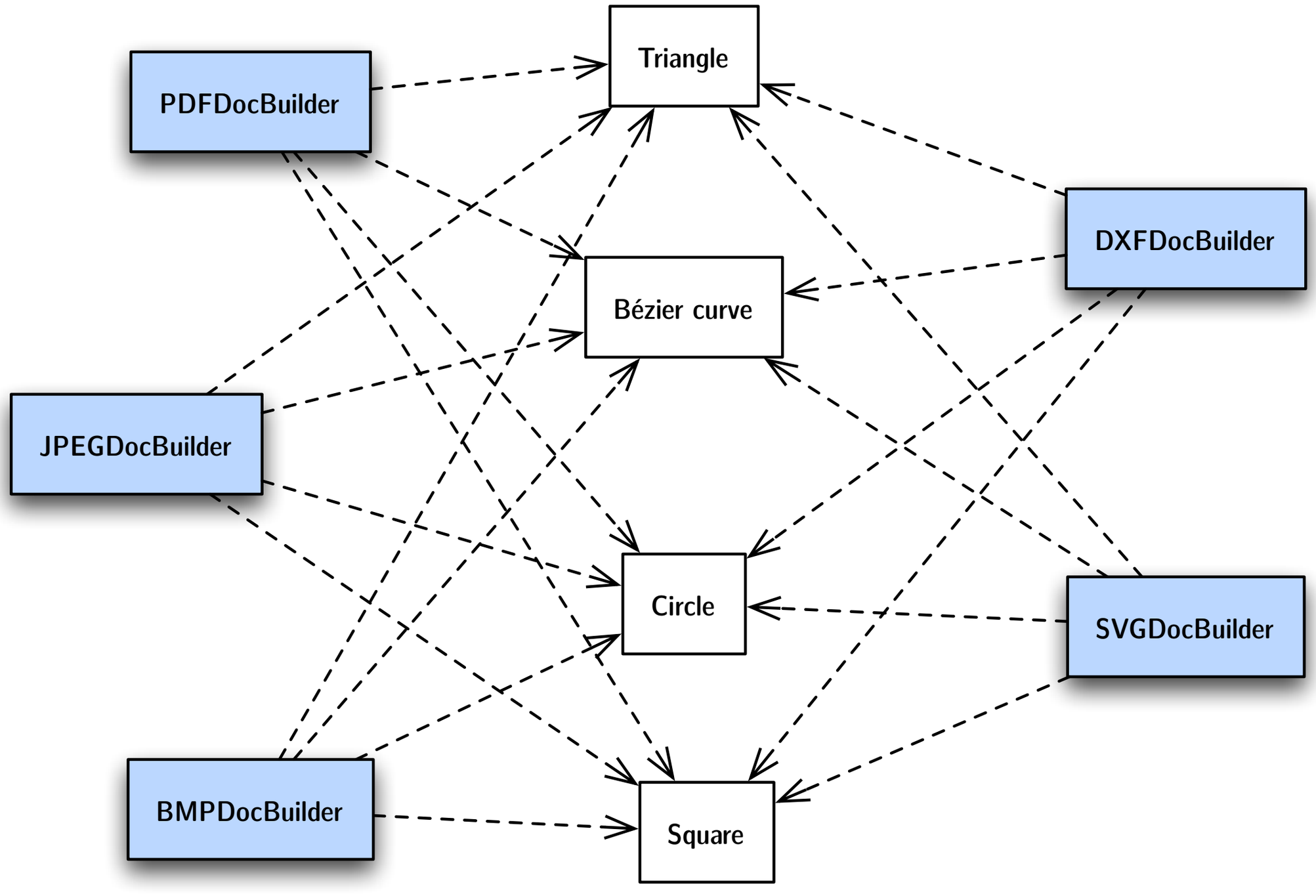
- We want to support further shapes:
  - Stars
  - Bubbles
  - Arrows
  - Bézier curves
  - …
- We want to support further file formats:
  - BMP
  - JPEG
  - DXF
  - …



# Evolving the Application



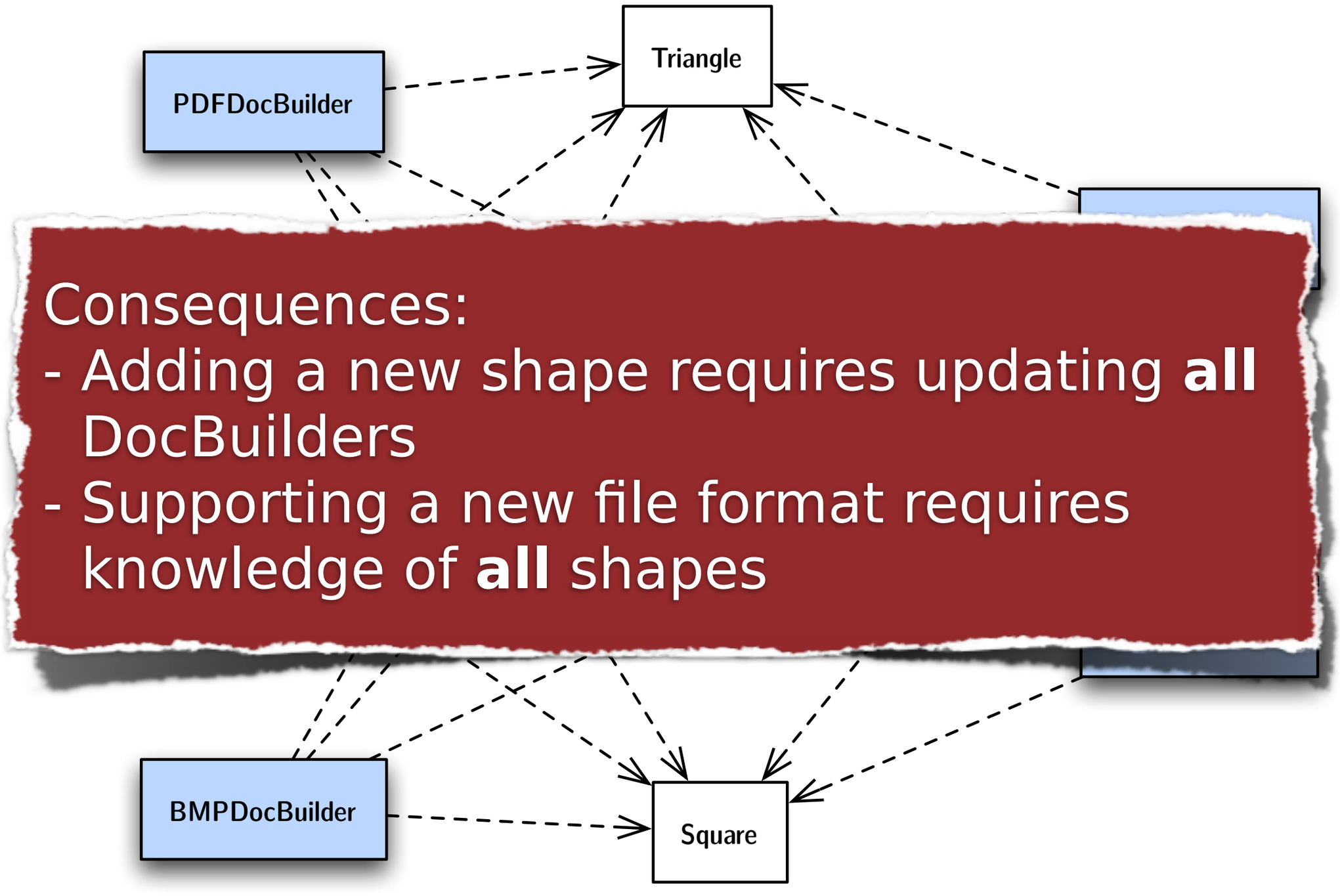
- If we follow the chosen path – i.e, each XDocBuilder collaborates with all shapes – **we end up with this**:



# Evolving the Application

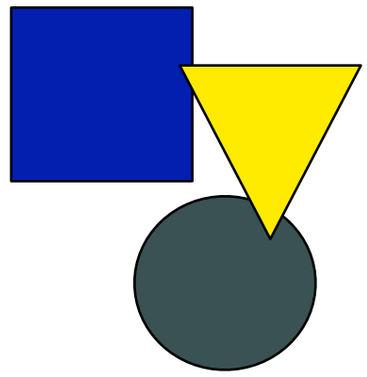


- If we follow the chosen path - i.e, each ...DocBuilder collaborates with all shapes - **we end up with this**:



# An Example SVG file





```
 2 [...]
 3 <svg versi                                    148" width="143pt" height="148pt">
 4  [...]
11  <g>
12   <rect x=185" y="218" width="72" height="
13   <rect x="185" y="218" width="72" height="
14   <ellipse cx="270" cy="327.5" rx="36.00007
15   <ellipse cx="270" cy="327.5" rx="36.00007
16   <path d="M 252 241 L 288 310 L 324 241 Z"
17   <path d="M 252 241 L 288 310 L 324 241 Z"
18  </g>
19 </svg>
```

A circle is represented using an ellipse.

No direct support for triangles.

Each ...DocBuilder will contain quite some domain logic.

# We then identify responsibilities and collaborators.



| Class | Collaborations |
|---|---|
| PDFDocBuilder | ~~Figure~~<br>~~Text~~<br>~~Circle~~<br>~~Triangle~~<br>~~Square~~ |

| Class | Collaborations |
|---|---|
| Figure | |
| **Responsibilities** | |
| Maintain a list of "shapes" | |

This solution does not facilitate software evolution!

We need to think again about the collaborations and responsibilities!

| Responsibilities | Collaborations |
|---|---|
| Save a figure to a ".svg" file | ~~Figure~~<br>~~Text~~<br>~~Circle~~<br>~~Triangle~~<br>~~Square~~ |

# There are some things to consider.



| Class | Collaborations |
| --- | --- |
| Names do matter! | If this list gets too long, it's probably time to split up the class. |
| **Responsibilities** | If you have cyclic collaboration dependencies its time to think about introducing abstractions. |
| If this list gets too long, it's probably time to split up the class. | |
| The responsibilities a class has should be related. | |

# A new class with new responsibilities and collaborators.



| Class | Collaborations |
|---|---|
| Canvas | Color |

**Responsibilities**

- draw text, lines, bézier curves, ellipses and points

… identification of a new artificial class (w.r.t. the domain description)

| Class | Collaborations |
|---|---|
| Text | Color<br>Canvas |

**Responsibilities**

- Maintain the text
- draw itself on a canvas

… using primitive shapes

# We then identify responsibilities and collaborators.



| Class | Collaborations |
|---|---|
| Figure | ▸ Text<br>▸ Circle<br>▸ Triangle<br>▸ Square |
| **Responsibilities** | |
| ▸ Maintain a list of "shapes" | |

| Class | Collaborations |
|---|---|
| PDFDocBuilder | ▸ Canvas |
| **Responsibilities** | |
| ▸ Save figure as ".pdf" file<br>▸ act as a Canvas | |

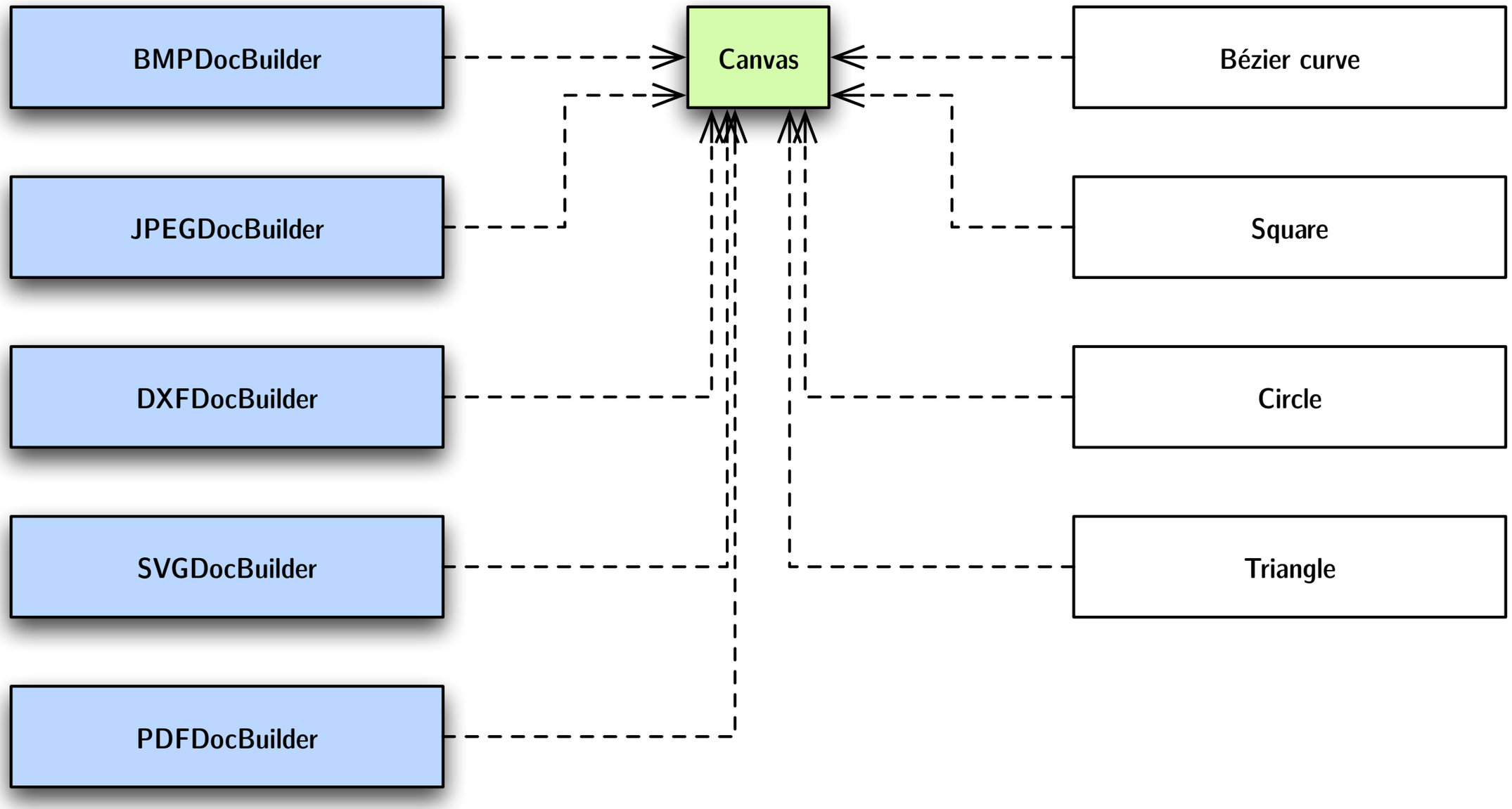
| Class | Collaborations |
|---|---|
| Canvas | ▸ Color |
| **Responsibilities** | |
| ▸ draw text, lines, bézier curves, ellipses and points | |

| Class | Collaborations |
|---|---|
| SVGDocBuilder | ▸ Canvas |
| **Responsibilities** | |
| ▸ Save figure as ".svg" file<br>▸ act as a Canvas | |

# Evolving the Application



- If we follow the new path - i.e, each XDocBuilder is basically a canvas - **we end up with this**:



# “Background Literature”



- The International Conference on Object Oriented Programming, Systems, Languages and Applications (OOPSLA); ACM Press, 1989

**A Laboratory For Teaching Object-Oriented Thinking**

Kent Beck, Apple Computer, Inc.
Ward Cunningham, Wyatt Software Services, Inc.

It is difficult to introduce both novice and experienced procedural programmers to the anthropomorphic perspective necessary for object-oriented design. We introduce CRC cards, which characterize objects by class name, responsibilities, and collaborators, as a way of giving learners a direct experience of objects. We have found this approach successful in teaching novice programmers the concepts of objects, and in introducing experienced programmers to complicated existing designs.

**1. Problem**

The most difficult problem in teaching object-oriented programming is getting the learner to give up the global knowledge of control that is possible with procedural programs, and rely on the local knowledge of objects to accomplish their tasks. Novice designs are littered with regressions to global thinking: gratuitous global variables, unnecessary pointers, and inappropriate reliance on the implementation of other [illegible]

reduces to teaching the design of objects. We focus on design whether we are teaching basic concepts to novices or the subtleties of a complicated design to experienced object programmers.

Rather than try to make object design as much like procedural design as possible, we have found that the most effective way of teaching the idiomatic way of thinking with objects is to immerse the learner in the “object-ness” of the material. To do this we must remove as much familiar material as possible, expecting that details such as syntax and programming environment operation will be picked up quickly enough once the fundamentals have been thoroughly understood.

It is in this context that we will describe our perspective on object design, its concrete manifestation, CRC (for Class, Responsibility, and Collaboration) cards, and our experience using these cards to teach both the fundamentals and subtleties of thinking with objects.

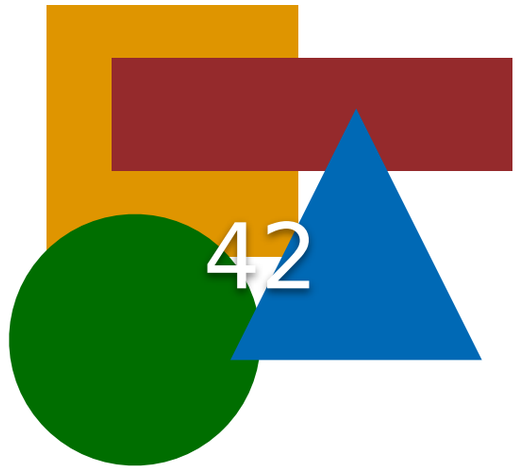
# Object-Oriented Thinking

- The inheritance relationship

# A common pitfall in object-oriented design is the inheritance relation.



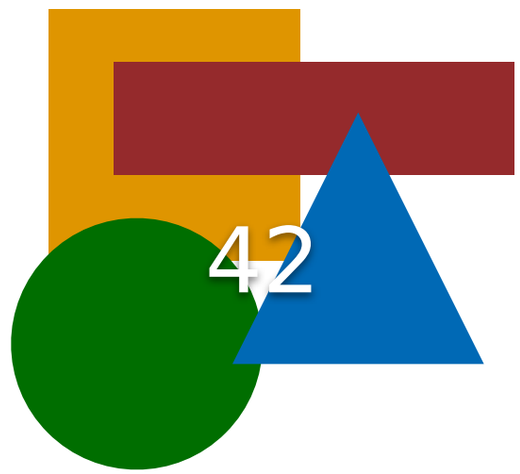
- Now let’s assume we want to further evolve our library and add support for **Rectangle**s…



# A common pitfall in object-oriented design is the inheritance relation.



- Now let's assume we want to further evolve our library and add support for **Rectangle**s…
- Should **Rectangle** inherit from **Square**?
- Should **Square** inherit from **Rectangle**?
- Is there some other solution?

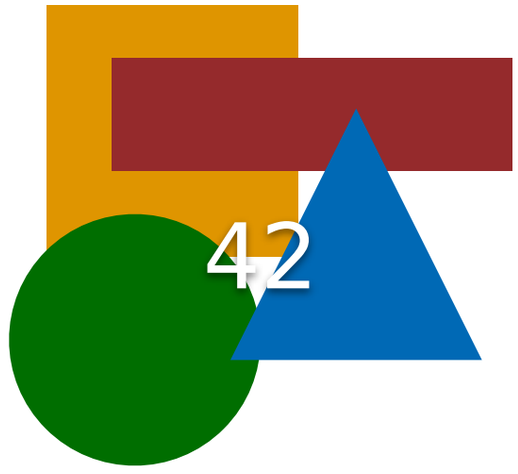


# A common pitfall in object-oriented design is the inheritance relation.



- Now let's assume we want to further evolve our library and add support for **Rectangle**s…
- Should **Rectangle** inherit from **Square**?
- Should **Square** inherit from **Rectangle**?
- Is there some

A first test:
"Is a Rectangle a Square?"



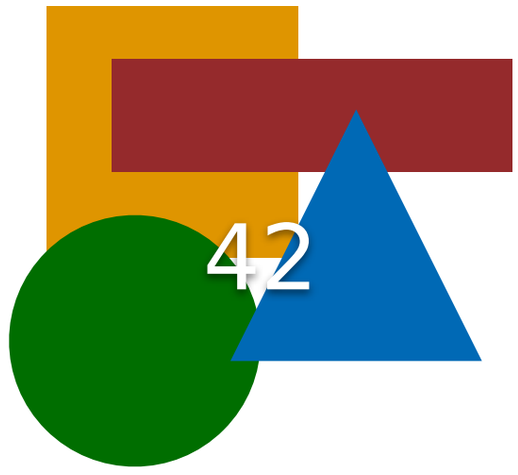
# A common pitfall in object-oriented design is the inheritance relation.



- Now let's assume we want to further evolve our library and add support for **Rectangle**s…
- ~~Should **Rectangle** inherit from **Square**?~~
- Should **Square** inherit from **Rectangle**?
- Is there some

A first test:
"Is a Rectangle a Square?"

No.



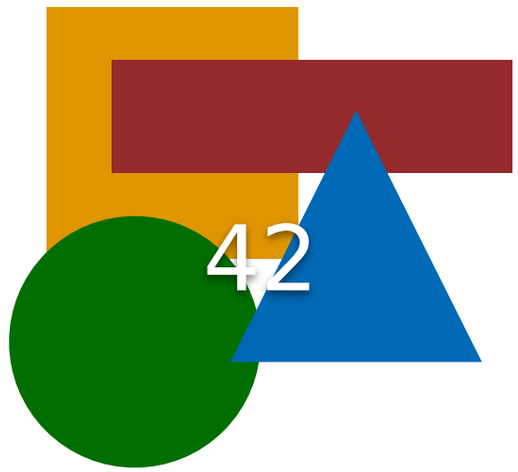
# A common pitfall in object-oriented design is the inheritance relation.



- Now let's assume we want to further evolve our library and add support for **Rectangle**s…
- ~~Should **Rectangle** inherit from **Square**?~~
- Should **Square** inherit from **Rectangle**?
- Is there som

A first test:
"Is a Square a Rectangle"?

Well… yes, but … how about a Square's behavior?



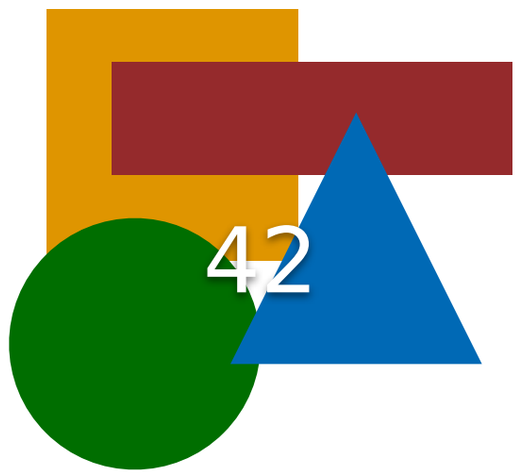
# A common pitfall in object-oriented design is the inheritance relation.



- Now let's assume we want to further evolve our library and add support for **Rectangle**s…
- ~~Should **Rectangle** inherit from **Square**?~~
- ~~Should **Square** inherit from **Rectangle**?~~
- Is there som

A first test:
"Is a Square a Rectangle"?

Well… yes, but … how about a Square's behavior?



# Object-Oriented Thinking

- Summary

TECHNISCHE UNIVERSITÄT DARMSTADT

# A large number of Design Heuristics and Design Principles exists that help you to design “better” programs.



- Low Coupling
- High Cohesion
- Single Responsibility Principle
- Don’t repeat yourself
- No cyclic dependencies
- Liskov Substitution Principle
- Open-Closed Principle
- …

# The goal of this lecture is to enable you to systematically carry out small(er) software projects that produce well-designed software.

- Identifying and (re-)distributing responsibilities among objects / classes is one of the major tasks when designing and evolving object-oriented programs.
- Having classes with well identified responsibilities facilitates the comprehension, maintenance and evolution of the software.