Behave! Solid

11 behavioral patterns as comics

Vincent LAURIER

Illustrated by Olivier ETIENNE

Behave! Solid

11 behavioral patterns as comics

Vincent LAURIER and Olivier ETIENNE

This book is for sale at http://leanpub.com/behave-solid

This version was published on 2021-10-19

Tweet This Book!

Please help Vincent LAURIER and Olivier ETIENNE by spreading the word about this book on Twitter!

The suggested tweet for this book is:

I just bought a comic book about design patterns. It sounds interesting. I can’t wait to know more!

The suggested hashtag for this book is #BehaveSolid.

Find out what other people are saying about the book by clicking on this link to search for this hashtag on Twitter:

#BehaveSolid

Contents

Preface

A word from the author

"Behave! Solid" is for me much more than a book!

It was first a concept, then a project, and finally a story.

A concept

2014 was the year I decided to jot down some ideas and various personal projects. I could not help mentioning and thinking about them frequently, but never took any action! So I listed some criteria, approximately ten, on a piece of paper which has since vanished. Each idea was looked into carefully, and only "Solid" was selected!

I could not remember all the requirements, but only the most important at the time: no financial investment, no stock management and a collaborative project, making the best of everyone's skills. I made up my mind! I asked Olivier my best friend to join me on this adventure! He put his drawing skill forward and in my case it was programming. The project was also able to tackle other requirements that I had in mind:

- originality: "A children's comic book about design patterns! Really?"
- never out of date: "Design Patterns: Elements of Reusable Object-Oriented Software", the **G**ang **O**f **F**our's book dates from 1994, and remains a reference to this day!

A project

The idea was good. So we decided to start working on the book.

Regular meetings, sketches, reference images, storyboards, code implementation, book cover's ideas, a title … everything was coming together. Working hard with a good plan. For those who wonder why "Behave! Solid" was chosen as a title, it is a reference to the term "behavioral" in the subtitle. A year later, I could no longer work on the project due to serious health problems. Olivier himself had other priorities.

A story

260 pages written, a 40% illustrated book, "Behave! Solid" stalled in March 2015. We kept in touch from time to time, and talked about it.

In 2019, I was back on my feet and got a job as a programmer. I sometimes shared ideas and concept about this project with my colleagues. The positive feedbacks that I had, made me want to work on it again. I decided to free up some time. I would do it alone and take drawing lessons if I had to!

I then realized I could not do it all by myself. Luckily, Olivier was available again and motivated too. We looked at it with a new pair of eyes.

260 pages was way too long with too much information! A second and strip down version was necessary. It was amazing, as if we never stop working on it in the first place.

October 5th, 2020, as I'm writing these lines, we are planning to release the book by the end of November. So it will finally be completed with many more in the pipeline!

A word from the illustrator

I have always been passionate about programming. But I mostly enjoyed a procedural approach. What mattered to me was to get the job done.
As I was illustrating and was rereading the book, I gradually understood the point of OOP and the purpose of interfaces and abstract classes. I have evolved not only as an artist but also as a programmer.
Our flexible collaboration has allowed me to explore different graphic styles and techniques.
As a whole, I really enjoyed working on this project. I have published my first comic book, fulfilling my childhood dream!

About this book

Who is this book for?

I believe it will be a game changer. It is a book intended for developers, but I challenged myself to keep it simple. It brings a smile to my face when thinking that a toddler could understand the stories, and perhaps later on, discover the underlying concepts. Who knows?

But don't be fooled! it is indeed intended for you, programmers. We will not be covering the basic because we assume that you are familiar with object-oriented programming. Code samples are written in PHP but should be accessible for all coders.

Advanced readers can use this book as a teaching aid; It should be a good mnemonic support for everyone!

Independent chapters

"Solid! behave" includes eleven independent chapters. Each of them focuses on a unique design pattern. In rare cases, we reference other parts of the book, but you are free to read them in any order really.

The children's section

Each chapter begins with a story, followed by questions and answers. These first three sections are accessible for children. However, I invite developers to also read the answers. They are short, and relate to key elements in the story.

Implementation examples

I made the choice not to follow the standards. A "snake_case" notation has been used as opposed to the less readable "CamelCase". Also, I avoided systematic strict typing. This could help those who are not familiar with PHP to only focus on what is important. In summary, clarity and understanding were chosen over standards.

UML diagrams

Following the above, the GOF UML diagrams have been “adapted” on a case-by-case basis. Sometimes the terms come from the GOF, and other times from the comic. The diagrams appear straight after the implementation to summarize and highlight the essentials. However familiar you are with UML notation in general, numbered bullet points are available to help you.

Answers for developers

Questions that were intended for children are now answered by adding technical terms. This allows a deeper understanding of the pattern. Sometimes it leads to a more detailed implementation, and other time it engages into theoretical reflections. I wanted to keep it relatively short, as the aim of the book is to quickly understand the essence of each design pattern.

About the author and the illustrator

PHP developer since 2009, **Vincent Laurier** mainly works with the Symfony framework. Former Math and Physics teacher, he values pedagogy and knowledge sharing. His first book “Behave! Solid”, perfectly illustrates his effort to make all subjects more accessible. Code design, elegance and quality mean a lot to him.

Olivier Etienne is a curious and passionate autodidact. Among other things like programming or music, drawing is a big part of his life, and has been since his childhood. Self-taught he only works with open source software : Blender, Inkscape and Gimp. Today, his talent as an illustrator benefits this educational project, which involves the skills he values the most.

Thanks to

From the author

To Olivier, for his friendship, his positivity and his work.

To my wife, for her support and patience. Thanks to her, the idea became a reality.

To my mother, for her careful proofreading.

To friends, colleagues and family, for their encouragements.

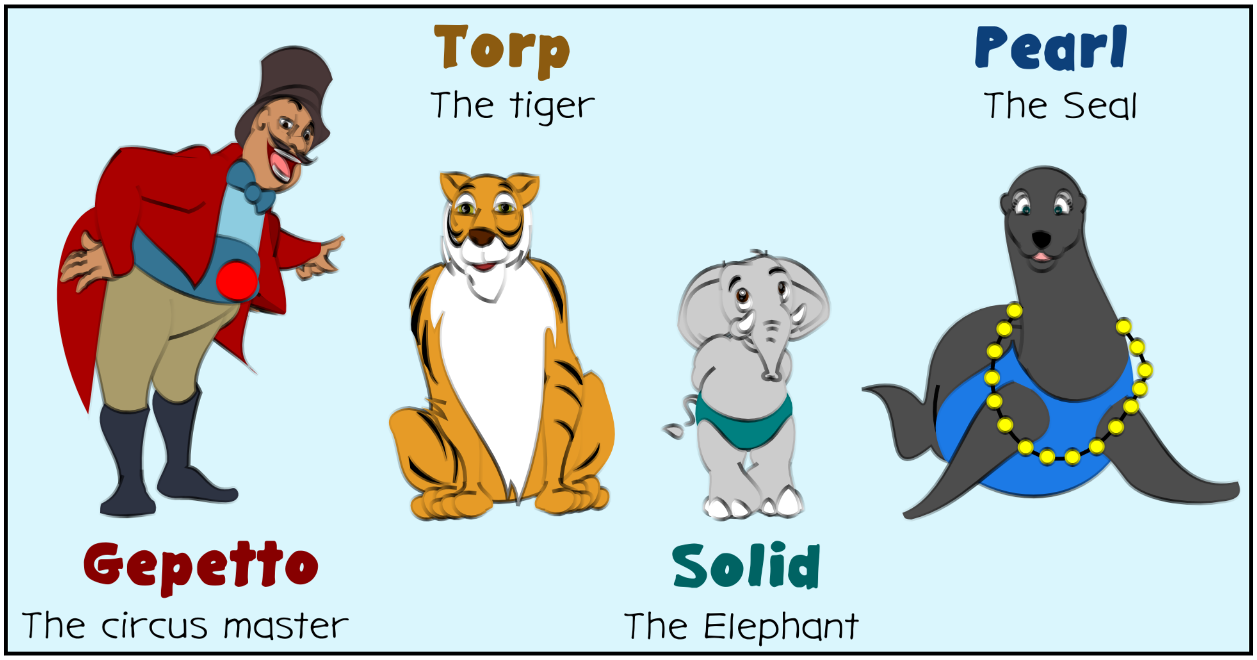
From the illustrator

To Nathan and Jérôme, my art students, who have considerably helped me improve my skills.

To Vincent, for the added value of our friendship in everyday life.

To my family, for their invaluable support.

And now, please welcome … the stars of the show!



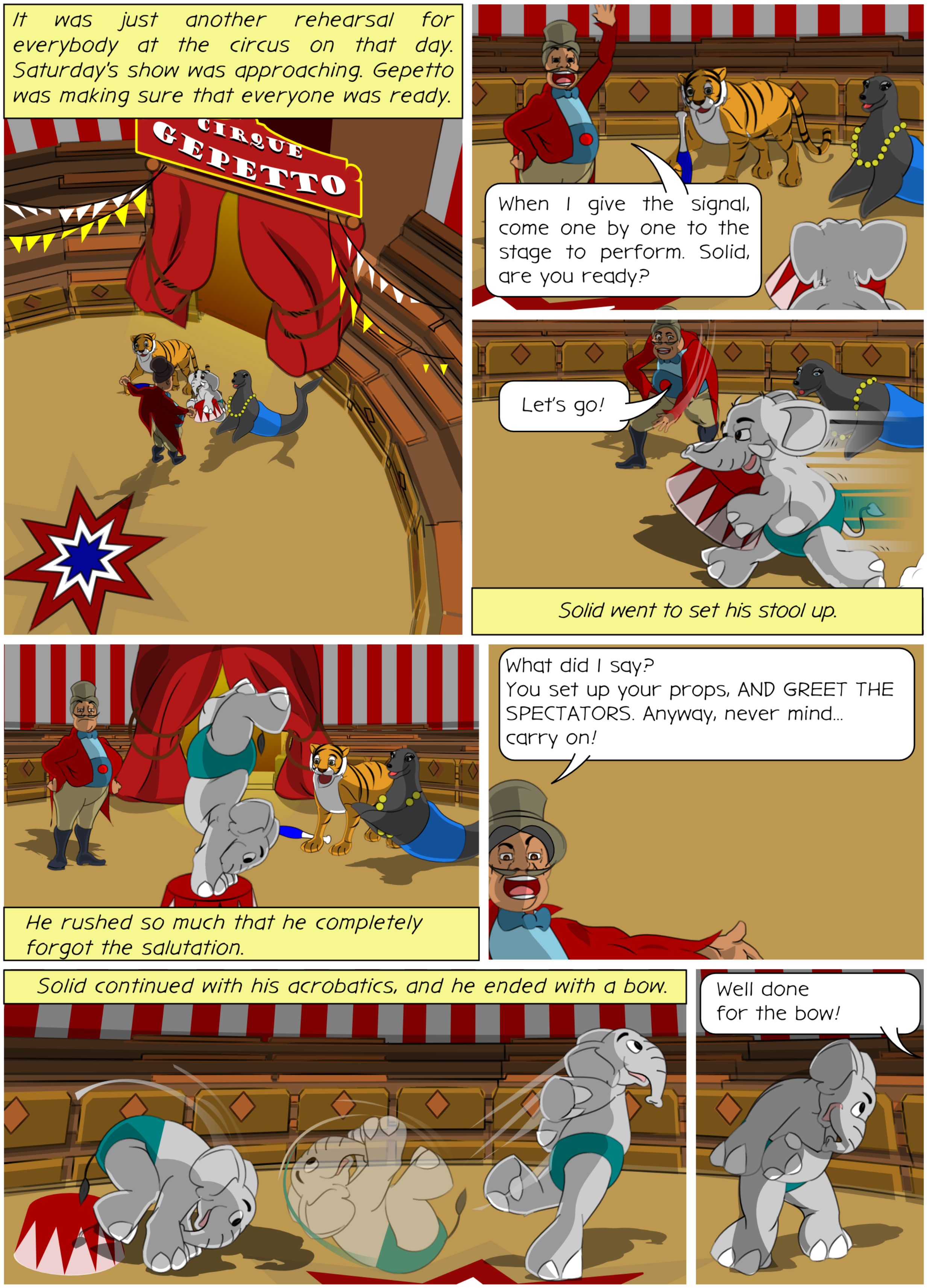
Template method

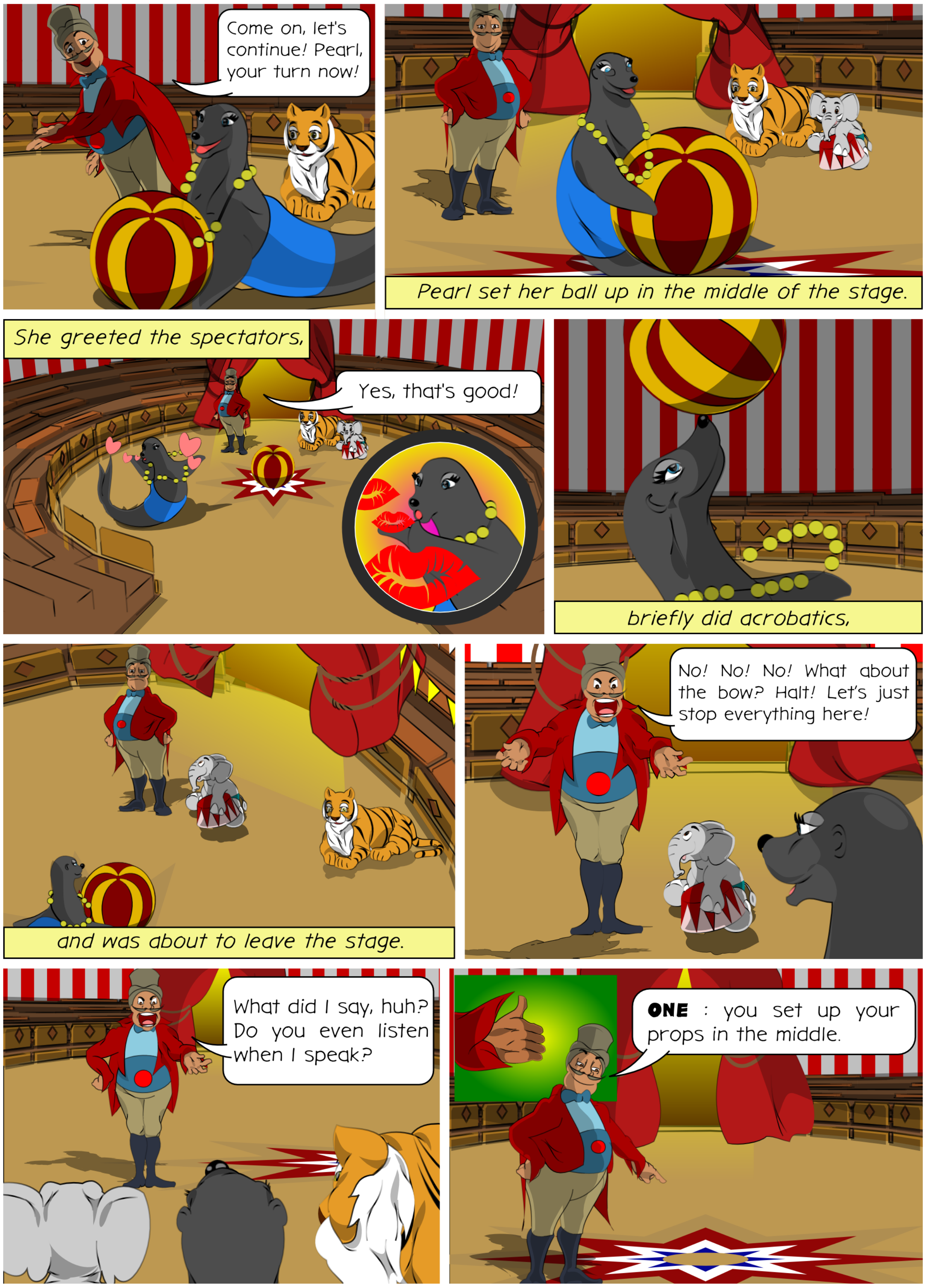
Intent according to the GOF

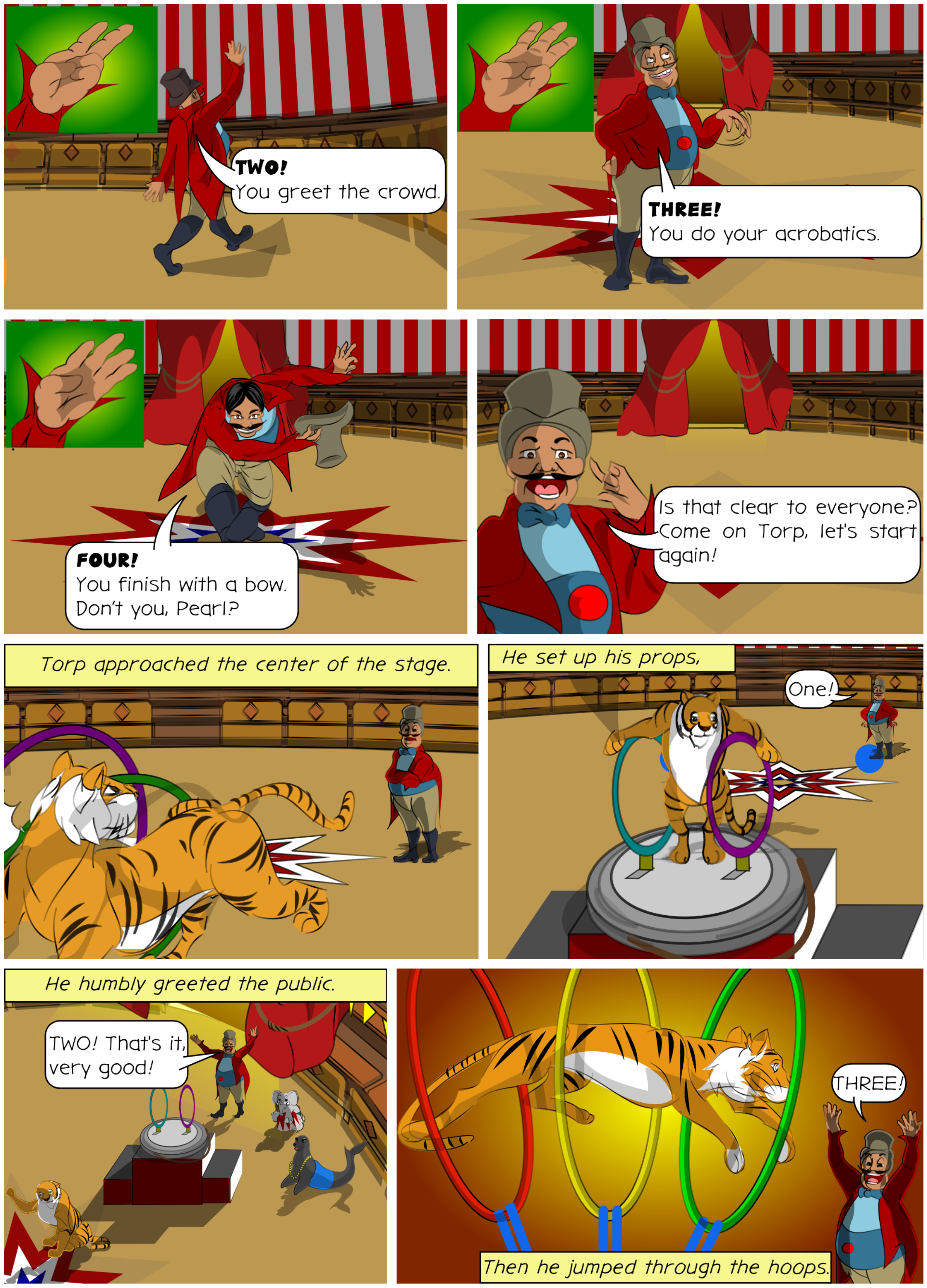
Define the skeleton of an algorithm in an operation, deferring some steps to subclasses. Template Method lets subclasses redefine certain steps of an algorithm without changing the algorithm's structure.

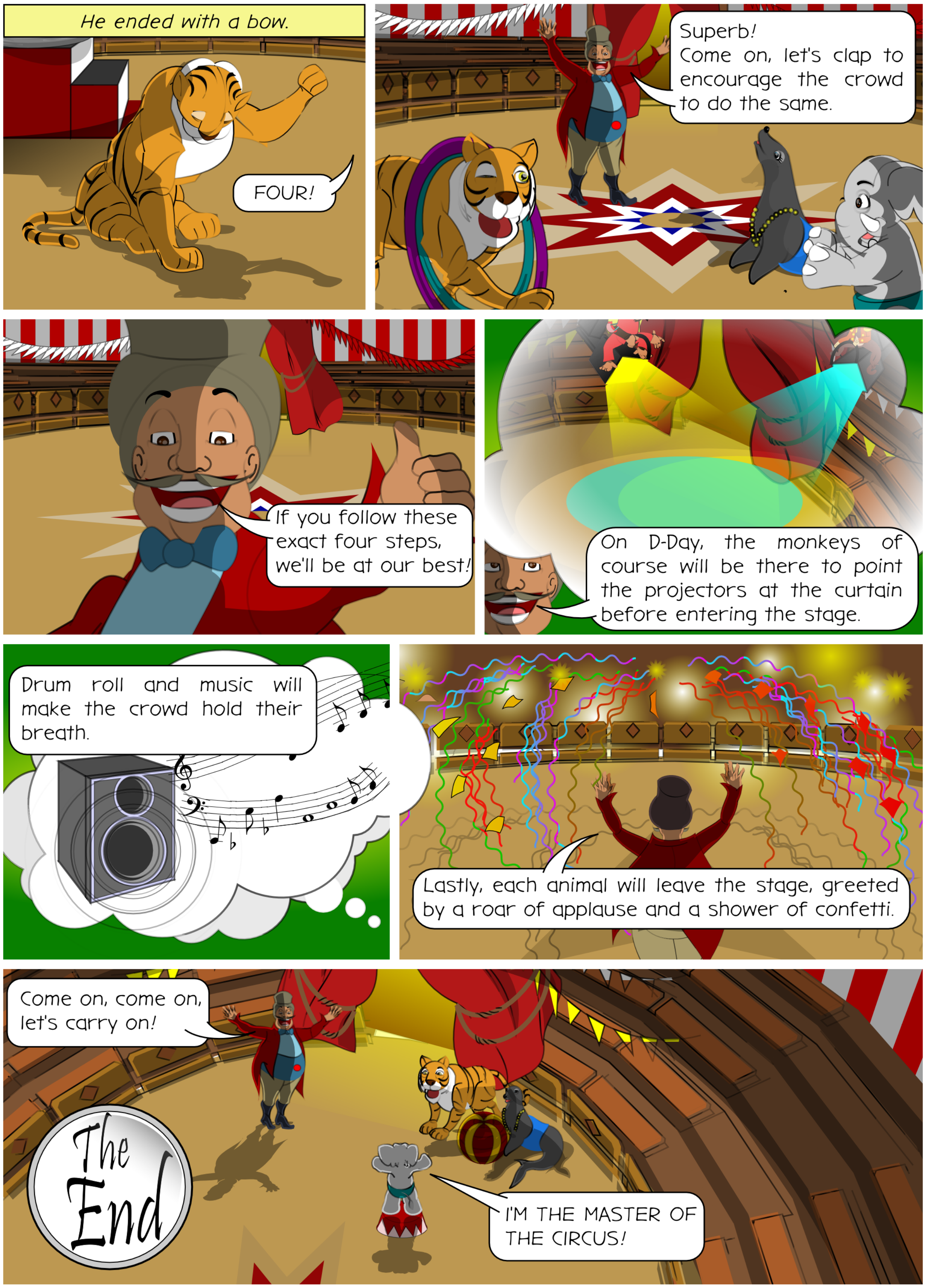
Let's explain this definition with a short illustrated story:

"Gepetto does the show"


It was just another rehearsal for everybody at the circus on that day. Saturday's show was approaching. Gepetto was making sure that everyone was ready.
CIRQUE GEPETTO
When I give the signal, come one by one to the stage to perform. Solid, are you ready?
Let's go!
Solid went to set his stool up.
He rushed so much that he completely forgot the salutation.
What did I say? You set up your props, AND GREET THE SPECTATORS. Anyway, never mind... carry on!
Solid continued with his acrobatics, and he ended with a bow.
Well done for the bow!


Come on, let's continue! Pearl, your turn now!
Pearl set her ball up in the middle of the stage.
She greeted the spectators,
Yes, that's good!
briefly did acrobatics,
and was about to leave the stage.
No! No! No! What about the bow? Halt! Let's just stop everything here!
What did I say, huh? Do you even listen when I speak?
ONE : you set up your props in the middle.


TWO!
You greet the crowd.
THREE!
You do your acrobatics.
FOUR!
You finish with a bow.
Don't you, Pearl?
Is that clear to everyone? Come on Torp, let's start again!
Torp approached the center of the stage.
He set up his props,
One!
He humbly greeted the public.
TWO! That's it, very good!
THREE!
Then he jumped through the hoops.


He ended with a bow.
FOUR!
Superb! Come on, let's clap to encourage the crowd to do the same.
If you follow these exact four steps, we'll be at our best!
On D-Day, the monkeys of course will be there to point the projectors at the curtain before entering the stage.
Drum roll and music will make the crowd hold their breath.
Lastly, each animal will leave the stage, greeted by a roar of applause and a shower of confetti.
Come on, come on, let's carry on!
The End
I'M THE MASTER OF THE CIRCUS!

General questions

1. What is the purpose of today's session at the circus?
2. Why does Gepetto have to lead by example?

Specific questions

3. Which steps the animals need to follow?
4. Does Gepetto own a prop to do acrobatics?
5. Which animal does very well in the practice session?
6. On the big day, there will be little extras compared to the rehearsals. What are they?

In your opinion

7. Look at the last thumbnail. Why do you think Solid is shouting: "I'M THE MASTER OF THE CIRCUS!"?

General questions

1. Today, the goal is to do a practice session for the animals.

2. Gepetto must lead by example because Solid and Pearl have forgotten a step in the process.

Specific questions

3. To do their show, the animals must follow 4 steps: drop off their props, greet the audience, do their acrobatics, then bow.

4. No, Gepetto does not have any props to do acrobatics. It simply points out the location where the animals will have to put theirs.

5. Torp is the one who brilliantly performs, because he follows all the steps.

6. On the big day, there will be light and music when the animals enter the stage. When leaving the stage, confetti and cheering will follow.

In your opinion

7. You might think that Solid is shouting "I AM THE MASTER OF THE CIRCUS!" to tease Gepetto. He's a bit mischievous, and he likes to think he is in control, even if that's not entirely true ...

The analogy with the comic strip

The purpose

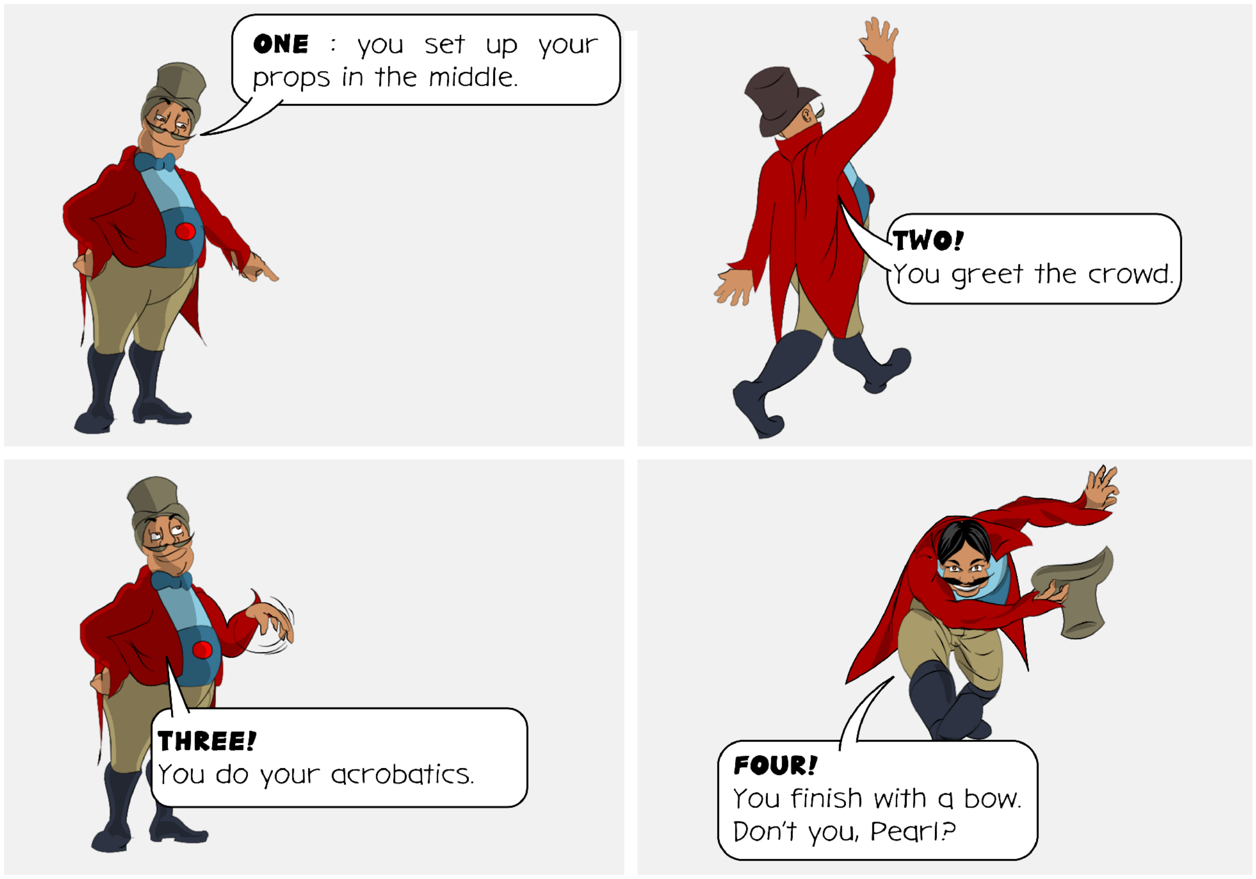
What did I say?
You set up your props, AND GREET THE SPECTATORS. Anyway, never mind... carry on!
No! No! No! What about the bow? Halt! Let's just stop everything here!

At the beginning of the story, Solid and Pearl freely perform on stage. Unfortunately, one of them forgets to greet the audience when entering the stage, while the other does not bow at the end. Therefore, Gepetto feels the need to put some structure in place. This is why he will use **template method** as a design pattern.

If we compare the comics to real-world software development, it is like letting two developers (Solid and Pearl) taking care of implementing a feature. They would only have implemented three steps instead of four in a very specific order. You will therefore have to step into Gepetto's shoes: let's see how!

The main idea

The circus master layouts the foundation by using a "**template method**" which could be called: ***"perform_on_stage"***. It will be the entry point to our class.
Have a look at the four steps below. Find two of them that are concrete, where Gepetto shows (implements) exactly what to do. Try to understand why the other two could be considered abstract.


ONE : you set up your props in the middle.
TWO! You greet the crowd.
THREE! You do your acrobatics.
FOUR! You finish with a bow. Don't you, Pearl?

Have you found them?

Gepetto does not have any props nor does he perform any acrobatics. These will be the two abstract steps to be defined by the animals.

Example of implementation

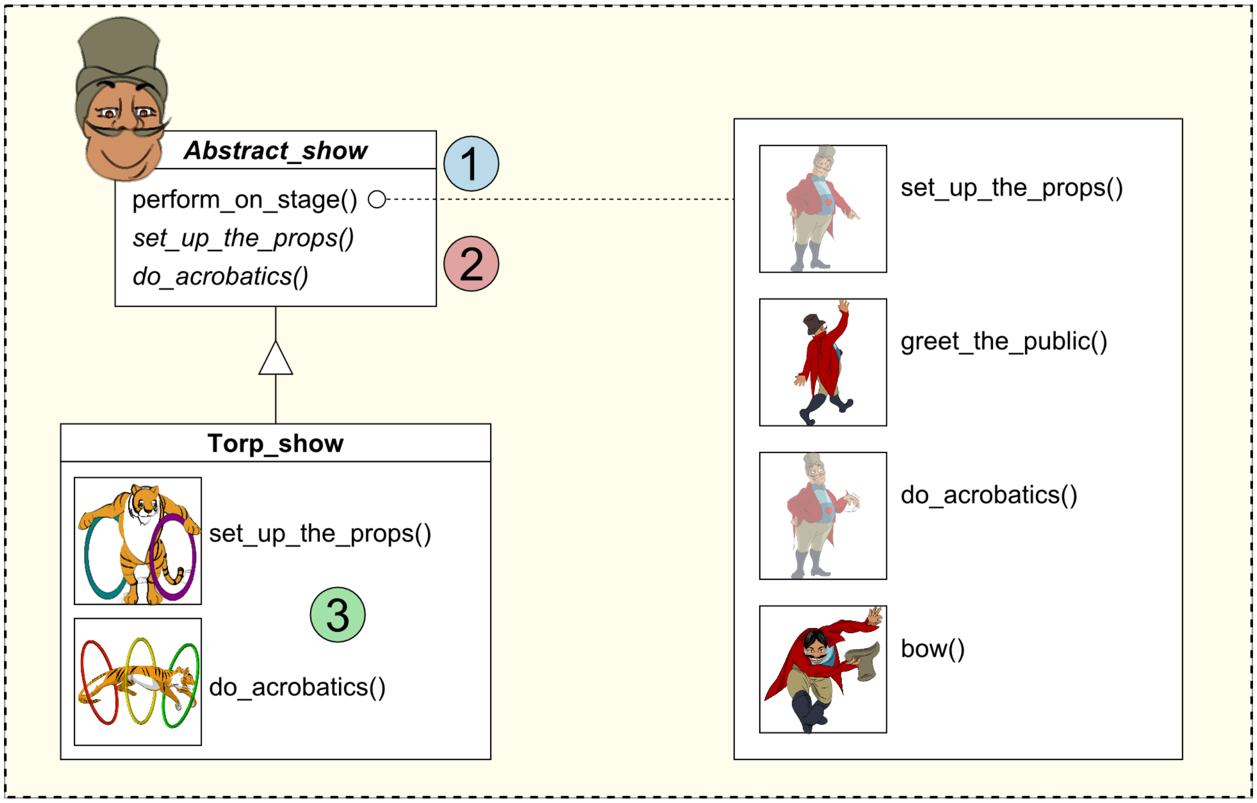
```
abstract class Abstract_show
{
    // Gepetto wants the animals to strictly follow his instructions.
    // Declaring the method as "final" prevents any override.
    // It is nevertheless an entry point of the class due to public visibility.
    final public function perform_on_stage()
    {
        // Here is the skeleton of the algorithm: 4 steps as in the story.
        $this->set_up_the_props();
        $this->greet_the_public();
        $this->do_acrobatics();
        $this->bow();
    }
    // Gepetto has no props, hence no implementation.
    // The keyword "abstract" will force children classes to propose one.
    // We also choose a protected visibility to enable it to do so.
    abstract protected function set_up_the_props();

    private function greet_the_public()
    {
        echo "Going around the stage is necessary to greet the public.";
    }
    // Gepetto does not perform any acrobatics.
    abstract protected function do_acrobatics();

    private function bow()
    {
        echo 'A bow concludes the performance.';
    }
}
// Torp follows the example given by Gepetto by extending the abstract class.
// He only customizes the necessary steps.
class Torp_show extends Abstract_show
{
    protected function set_up_the_props()
    {
        echo 'Torp sets up his hoops.';
    }

    protected function do_acrobatics()
    {
        echo 'He jumps through the hoops. Spendid!';
    }
}
// His performance thus follows the defined sequence.
(new Torp_show())->perform_on_stage();
```

UML modeling



1. Gepetto proposes a template method: perform_on_stage(). A pseudo implementation is visible on the right-hand side.
2. Some steps are abstract in the parent class.
3. Children classes, Torp_show for example, must therefore give a concrete implementation.

Answers for developers

General questions

1) What is the purpose of today's session at the circus?

The goal is to make sure that the animals are ready for the show, in other words, that they will follow perfectly determined steps. So remember the following: it is interesting to **use the template method pattern to set a framework as a guide**.

2) Why does Gepetto have to lead by example?

To prevent animals from making mistakes! Through inheritance, we instantly create rigidity.

Specific questions

3) Which steps the animals need to follow?

They are four of them: set up the props, greet the public, do acrobatics, then bow. They are not part of the public API of the abstract class, but describe the inner workings of the show. You will therefore notice their protected or private visibility in the code example.

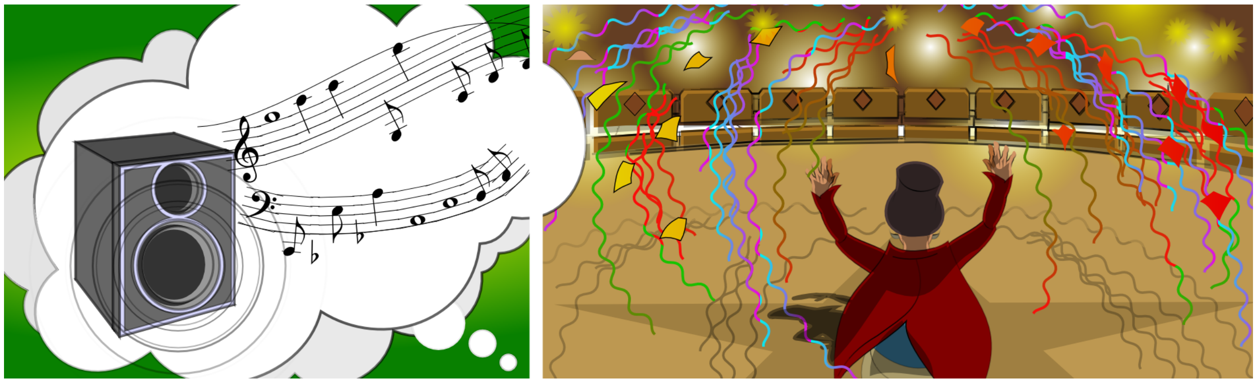
4) Does Gepetto own a prop to do acrobatics?

No! And that's why "set_up_the_props" or "do_acrobatics" are abstract methods. Therefore, the master of the circus himself represents an abstract class.

5) Which animal does very well in the practice session?

Torp is the one who will extend the class proposed by Gepetto. There is an instant benefit! While performing, he has to follow the four steps in the defined order.

6) On the big day, there will be little extras compared to the rehearsals. What are they?


On the big day

There will be music and confetti before and after each performance. This therefore raises a question about "**template method**". How do we proceed later on to add some logic in the workflow? For this to be possible, the abstract class will have to be set up accordingly. Have a look at the sample code below.

```
public function perform_on_stage()
{
   // To anticipate subsequent customization needs,
   // one can plan to add "empty steps" ...
   $this->when_entering_the_stage();

   $this->set_up_the_props();
   $this->greet_the_public();
   $this->do_acrobatics();
   $this->bow();

   // ...at the beginning and at the end!
   $this->when_leaving_the_stage();
}

// These are called "hooks" and do nothing by default,
// but can be overridden to add some logic if necessary.
protected function when_entering_the_stage(){};
protected function when_leaving_the_stage(){};
```

In your opinion

7) Look at the last thumbnail. Why do you think Solid is shouting "I'M THE MASTER OF THE CIRCUS!"?

Inversion of control

Come on, come on, let's carry on!
The End
I'M THE MASTER OF THE CIRCUS!

Solid likes to feel he is in control, being able to do whatever he wants. But the reality is quite different. Gepetto does dictate what happens.
The pattern "**template method**" follows a principle known as "**inversion of control**". To better

understand why, think about Solid as a class in your application, and the circus master, as one from a framework you are using. By extending the framework's class, we free ourselves from a certain amount of work, but there is at least two things that we have no control over:

- the algorithm's workflow, dictated by the framework
- what can be customized, also controlled by the framework

Most curious readers could look for the reason why "**inversion of control**" is also known as **Hollywood principle**.

See appendix page 135

.

Iterator

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/behave-solid.

"Inventory day"

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/behave-solid.

The analogy with the comic strip

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/behave-solid.

Example of implementation

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/behave-solid.

UML modeling

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/behave-solid.

Answers for developers

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/behave-solid.

SPL iterators

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/behave-solid.

Command

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/behave-solid.

"Under the door of the dressing room"

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/behave-solid.

The analogy with the comic strip

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/behave-solid.

Example of implementation

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/behave-solid.

UML Modeling

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/behave-solid.

Answers for developers

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/behave-solid.

The MacroCommand

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/behave-solid.

Interpreter

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/behave-solid.

"Gepetto's sign language"

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/behave-solid.

The analogy with the comic strip

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/behave-solid.

Example of implementation

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/behave-solid.

UML Modeling

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/behave-solid.

Answers for developers

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/behave-solid.

BNF and EBNF

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/behave-solid.

Strategy

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/behave-solid.

"No matter the accessory!"

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/behave-solid.

The analogy with the comic strip

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/behave-solid.

Example of implementation

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/behave-solid.

UML Modeling

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/behave-solid.

Answers for developers

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/behave-solid.

Null Object

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/behave-solid.

Visitor

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/behave-solid.

"Visiting the animal enclosures"

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/behave-solid.

The analogy with the comic strip

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/behave-solid.

Example of implementation

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/behave-solid.

UML Modeling

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/behave-solid.

Answers for developers

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/behave-solid.

Open-closed principle

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/behave-solid.

Chain of responsibilities

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/behave-solid.

"Who's next?"

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/behave-solid.

The analogy with the comic strip

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/behave-solid.

Example of implementation

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/behave-solid.

UML Modeling

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/behave-solid.

Answers for developers

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/behave-solid.

Observer

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/behave-solid.

"Torp is showing off ..."

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/behave-solid.

The analogy with the comic strip

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/behave-solid.

Example of implementation

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/behave-solid.

UML Modeling

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/behave-solid.

Answers for developers

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/behave-solid.

Dependency Inversion Principle

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/behave-solid.

Mediator

"Gepetto coordinates the opening of the park"

The analogy with the comic strip

Example of implementation

UML Modeling

Answers for developers

State

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/behave-solid.

"The circus ring in all its states"

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/behave-solid.

The analogy with the comic strip

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/behave-solid.

Example of implementation

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/behave-solid.

UML Modeling

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/behave-solid.

Answers for developers

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/behave-solid.

Memento

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/behave-solid.

“Photo shoot for Solid”

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/behave-solid.

The analogy with the comic strip

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/behave-solid.

Example of implementation

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/behave-solid.

UML Modeling

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/behave-solid.

Answers for developers

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/behave-solid.

Appendices

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/behave-solid.

The Hollywood principle

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/behave-solid.

Iterator to filter on rewards

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/behave-solid.

Example of implementation for the “Chain” expression

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/behave-solid.

Bibliography

Use of external resources