

Go places with
System Workers!

4D Summit 2023 presentation



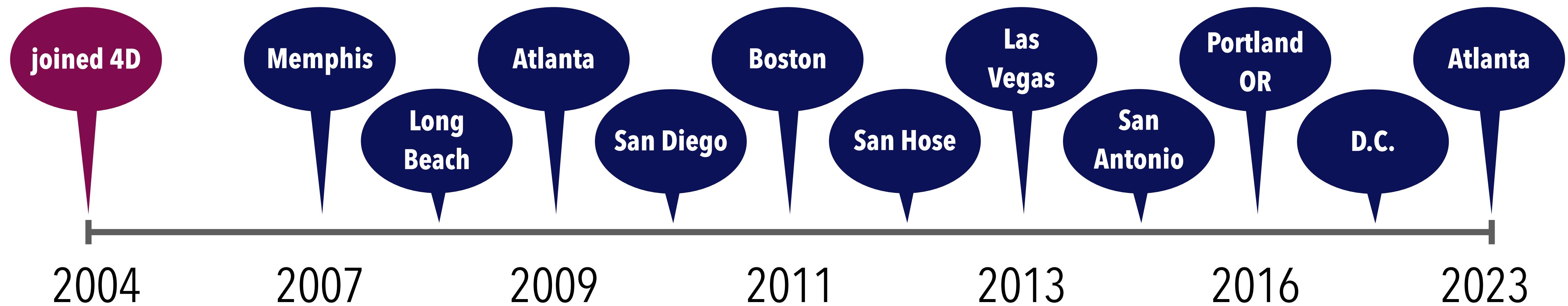
Atlanta, USA (October 3rd - 4th, 2023) and Paris, France (October 17th - 18th, 2023)



Keisuke Miyako

Technical Account Manager

Introduction



Atlanta, USA (October 3rd - 4th, 2023) and Paris, France (October 17th - 18th, 2023)

Introduction



first 4D version: 2004
prior programming knowledge: BASIC
4D is easy to learn, fun to code!



2004

2007

2009

2011

2013

2016

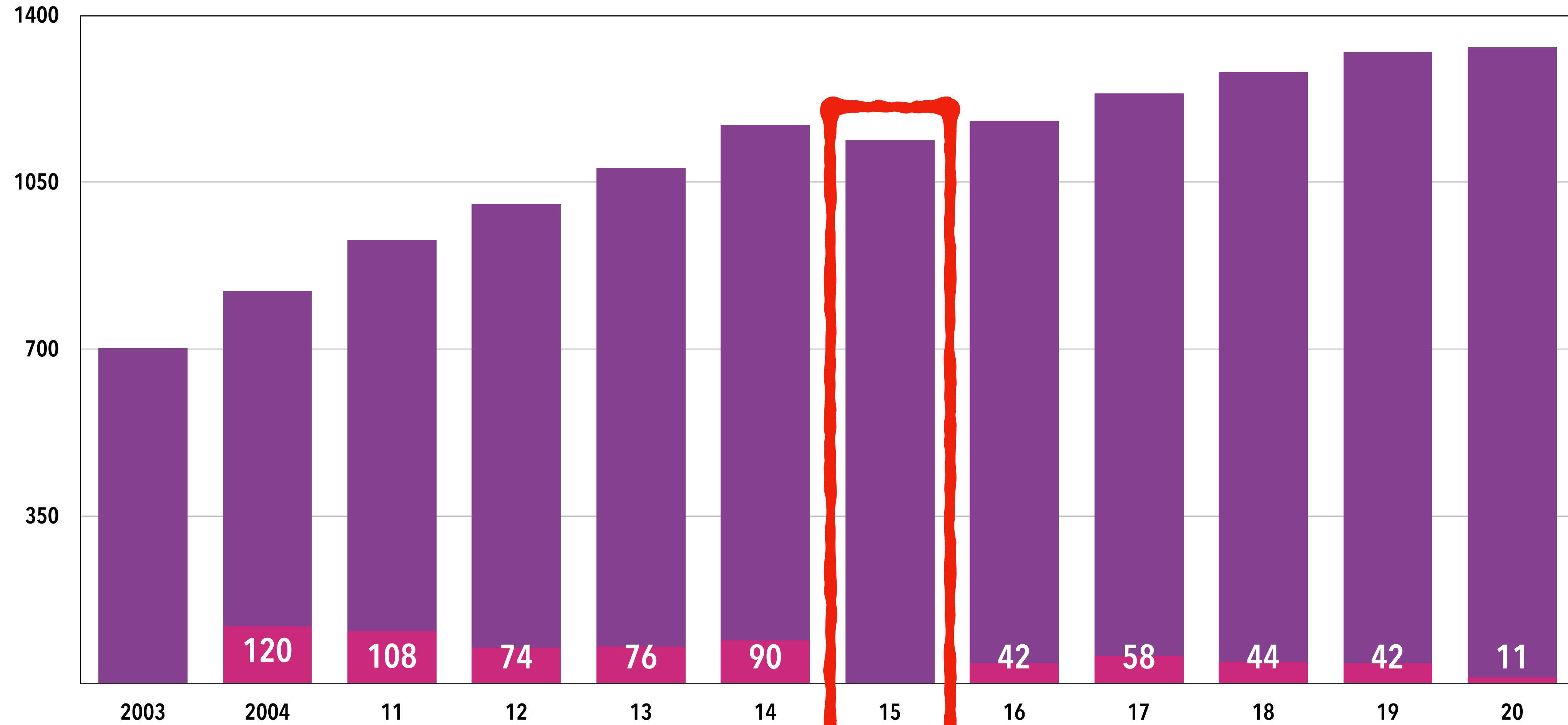
2023

Atlanta



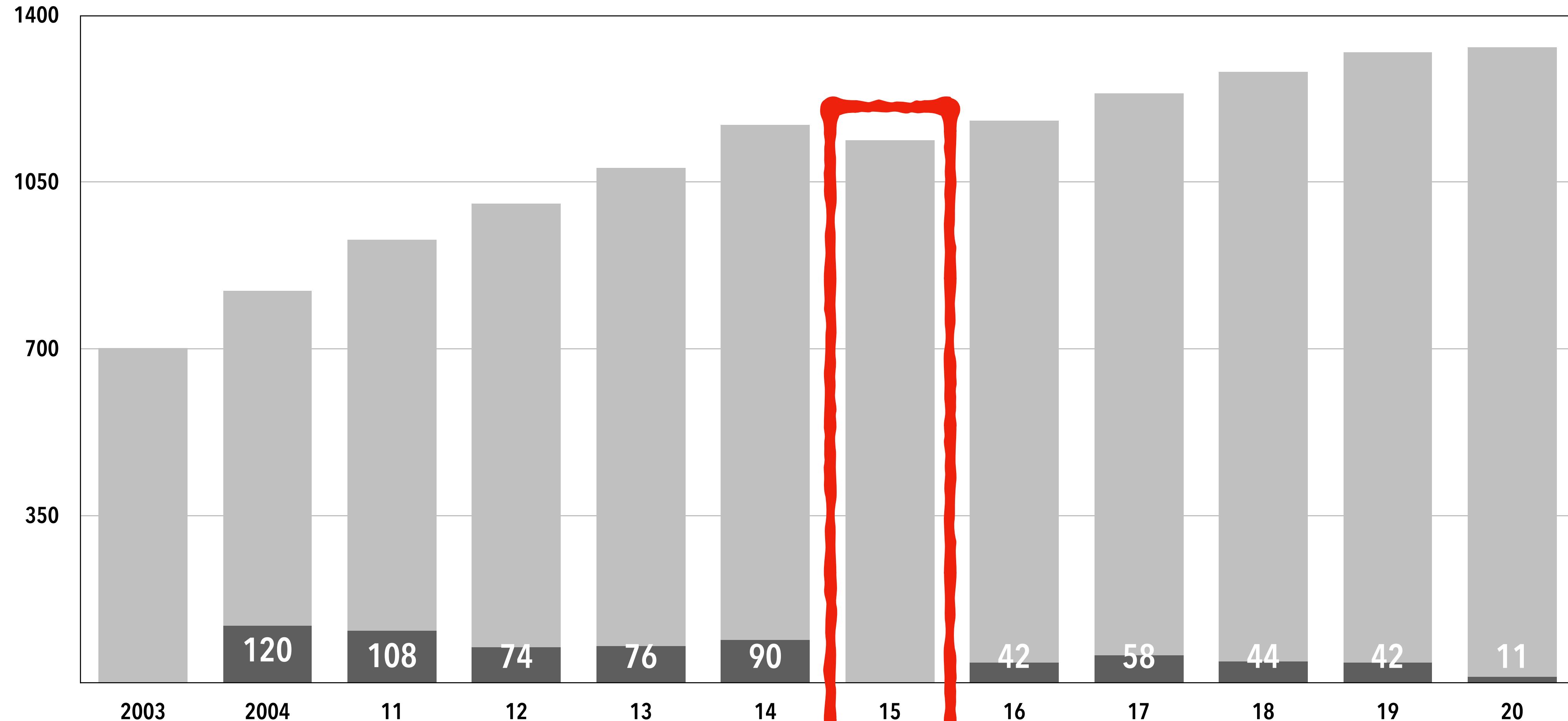
Atlanta, USA (October 3rd - 4th, 2023) and Paris, France (October 17th - 18th, 2023)

4D commands by version



Atlanta, USA (October 3rd - 4th, 2023) and Paris, France (October 17th - 18th, 2023)

4D commands by version



paradigm shift



Atlanta, USA (October 3rd - 4th, 2023) and Paris, France (October 17th - 18th, 2023)

Object

paradigm shift

v14: C_OBJECT, ARRAY OBJECT, JSON

v15: Object Fields

v16: Commands to access object attributes

v17: Object Notation, ORDA

v18: Project Mode, formula object

v19: Class API for object definition

v20: Blob object, property declaration



paradigm shift

v14: C_OBJECT, ARRAY OBJECT, JSON

v15: Object Fields

v16: Commands to access object attributes

v17: Object Notation, ORDA

v18: Project Mode, formula object

v19: Class API for object definition

v20: Blob object, property declaration



Procedural, Functional, Object Oriented

more commands the better!

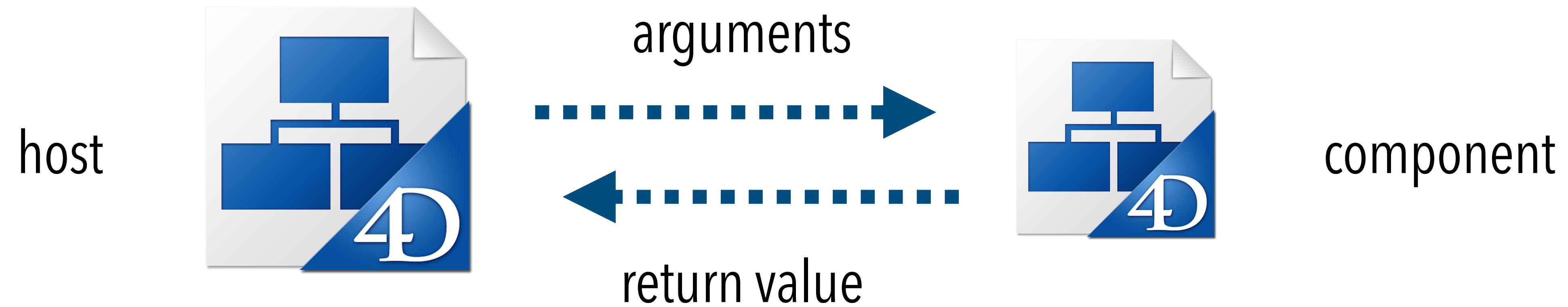


sequence of imperative commands

state (current record, current selection, process variables) is shared implicitly

Procedural, Functional, Object Oriented

fewer process variables the better!



dependency purely based on **arguments** and **return value**

state (current record, current selection, process variables) is not shared

Object

paradigm shift

v14: C_OBJECT, ARRAY OBJECT, JSON

v15: Object Fields

v16: Commands to access object attributes

v17: Object Notation, ORDA

v18: Project Mode, formula object

v19: Class API for object definition

v20: Blob object, property declaration



Object - Generic Data Structure

record

Student	
ID	2^{32}
firstName	A
lastName	A
englishLevel	2^{16}
finalExam	A
ranking	2^{16}
score	2^{32}

object

Student: {

ID:1,
 firstName: William,
 lastName: Shakespeare,
 englishLevel: 1,
 finalExam: pass,
 ranking: 1,
 score: 1616
}

alternative to...

- key/value pair
- associative arrays
- XML
- BLOB
- list properties

Object - Generic Data Structure

record

Student	
ID	2 ³²
firstName	A
lastName	A
englishLevel	2 ¹⁶
finalExam	A
ranking	2 ¹⁶
score	2 ³²

object

Formula creates a formula object based upon the *formulaExp* expression. *formulaExp* can be as simple as a single value or complex, such as a project method with parameters.

Formula

This

game changer!

You can specify the object on which the formula is executed. The properties of the object can then be accessed via the **This** command.

Object - Generic Data Structure with Code

record

Student	
ID	2 ³²
firstName	A
lastName	A
englishLevel	2 ¹⁶
finalExam	A
ranking	2 ¹⁶
score	2 ³²

object

Student: {

ID: 1,

firstName: William,

lastName: Shakespeare,

englishLevel: 1,

finalExam: pass,

eligibilityForTransfer(school),

ranking: 1,

score: 1616

}

alternative to...

- key/value pair
- associative arrays
- XML
- BLOB
- list properties

Procedural, Functional, Object Oriented

abstraction

```
This.capitalize(This.trim(This.firstName))
```

Inheritance kind of...

Person

```
.firstName:="William"
```

```
.lastName:="Shakespeare"
```

Student is based on **Person**

```
.englishLevel:=1
```

```
.finalExam:="pass"
```

encapsulation kind of...

Student

```
._ranking:=1
```

```
._score:=1616
```

polymorphism

Student

```
This.register:=Formula(registerStudent)
```

Teacher

```
This.register:=Formula(registerTeacher)
```



Procedural, Functional, Object Oriented

fewer commands, more classes!



an object contains **data** and **code** (properties and functions)

dynamic state is represented within an **instance** of an object

Procedural, Functional, Object Oriented

4D embraces all 3 paradigms!

procedural

functional

object oriented

user interface

global scope

generic code library

sharable module

asynchronous

event-driven



Procedural, Functional, Object Oriented

4D embraces all 3 paradigms!

procedural

user interface

global scope

11 new commands since v19

1. Create deployment license
2. **SET HELP MENU**
3. Last errors
4. **OBJECT SET SUBFORM CONTAINER VALUE**
5. **OBJECT Get subform container value**
6. Copy parameters
7. Get license usage
8. **WP DELETE TEXT BOX**
9. **WP New text box**
10. **WP Get data context**
11. **WP SET DATA CONTEXT**



Procedural, Functional, Object Oriented

4D embraces all 3 paradigms!

functional

generic code library
sharable module



4D Progress

4D SVG

4D ViewPro

4D Widgets



4D Go Mobile



Atlanta, USA (October 3rd - 4th, 2023) and Paris, France (October 17th - 18th, 2023)

Procedural, Functional, Object Oriented

4D embraces all 3 paradigms!

object oriented

**asynchronous
event-driven**

new built-in classes

FileHandle
HTTPRequest
SystemWorker
WebSocketServer/Connection

classic implementation

Open document, SEND PACKET, etc.
HTTP Get, HTTP Request
LAUNCH EXTERNAL PROCESS
WEB SEND RAW DATA



Atlanta, USA (October 3rd - 4th, 2023) and Paris, France (October 17th - 18th, 2023)

Go places with
System Workers!

4D Summit 2023 presentation

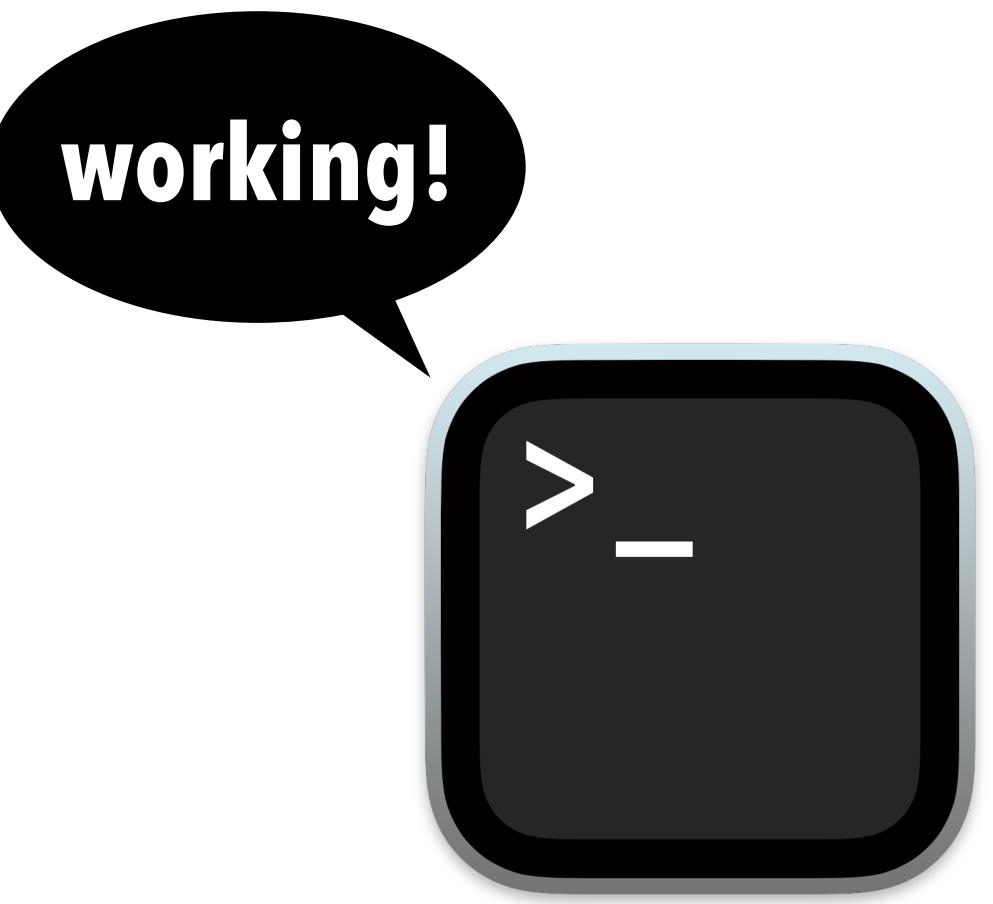
System Worker

- **object-oriented** implementation of **LAUNCH EXTERNAL PROCESS**
 - **efficient**: communication takes place in the **background**
 - **asynchronous**: process **streams of data**
 - **interactive**: respond to **command line prompts** and make **consecutive calls**

LAUNCH EXTERNAL PROCESS



4D process



external process

Why does it matter?



Atlanta, USA (October 3rd - 4th, 2023) and Paris, France (October 17th - 18th, 2023)

Progress Indicator - Command Line Interface

standard error stream



Progress Indicator - LAUNCH EXTERNAL PROCESS

standard error stream



System Worker is asynchronous!

- designed for time-consuming operations
 - **display progress information** in real time
 - **process large data** in smaller chunks
 - **abort command** if necessary

Execution Context

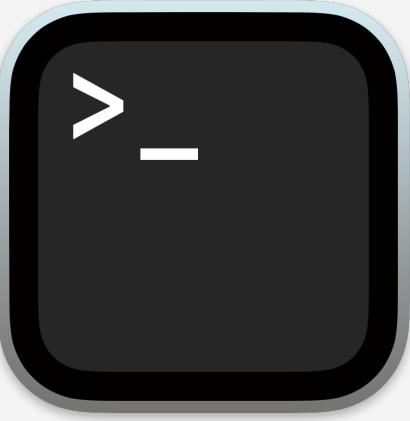
Execution Context

New process

```
$worker:=4D.SystemWorker.new()
```

abort

background thread



execution context does not exist anymore!

Execution Context

CALL WORKER

```
$worker:=4D.SystemWorker.new()
```

background thread



Execution Context

DIALOG

```
$worker:=4D.SystemWorker.new()
```

background thread



Demonstration

SystemWorker needs execution context

Recap



Atlanta, USA (October 3rd - 4th, 2023) and Paris, France (October 17th - 18th, 2023)

Recap

- external process is executed in **background thread**
- background thread invokes **SystemWorker instance**
- SystemWorker instance needs **execution context** (WORKER or DIALOG)
 - implement **callback functions** for asynchronous execution
 - implement **buffer** to process partial responses
 - implement **task queue** to consecutive execution



Inheritance



Atlanta, USA (October 3rd - 4th, 2023) and Paris, France (October 17th - 18th, 2023)

Inheritance

object oriented programming

generic

specific



Base class to implement
any command line interface

Subclass of **_CLI** to execute
the **7z** command line interface

Inheritance

object oriented programming

generic

specific



methods

.escape()

.quote()

implement the **7z**
command line
interface

methods

[.terminate()
.add()
.extract()

properties

controller

executableName

currentDirectory

executablePath

EOL

name

executableFile

platform

Usage: 7z <command> [<switches>...] <archive_name>
[<file_names>...] [@listfile]

Inheritance

object oriented programming

generic

specific



properties

controller
currentDirectory
EOL
executableFile

executableName
executablePath
name
platform

methods

.escape()
.quote()

implement the **curl** command line interface

.terminate()
.get()
.post()

Usage: curl [options...] <url>

Inheritance

object oriented programming

generic



Classes



_CLI.4dm

specific



_CLI_Controller.4dm



_Worker_Controller.4dm



_Form_Controller.4dm

Inheritance

object oriented programming

generic

specific



Classes



_CLI.4dm



SevenZip.4dm



_CLI_Controller.4dm



_Worker_Controller.4dm



_Form_Controller.4dm



_SevenZip_Form_Controller.4dm

Inheritance

object oriented programming

generic

specific



Classes



_CLI.4dm



cURL.4dm



_CLI_Controller.4dm



_Worker_Controller.4dm



_Form_Controller.4dm



_cURL_Form_Controller.4dm

Demonstration

SystemWorker and class inheritance

Recap



Atlanta, USA (October 3rd - 4th, 2023) and Paris, France (October 17th - 18th, 2023)

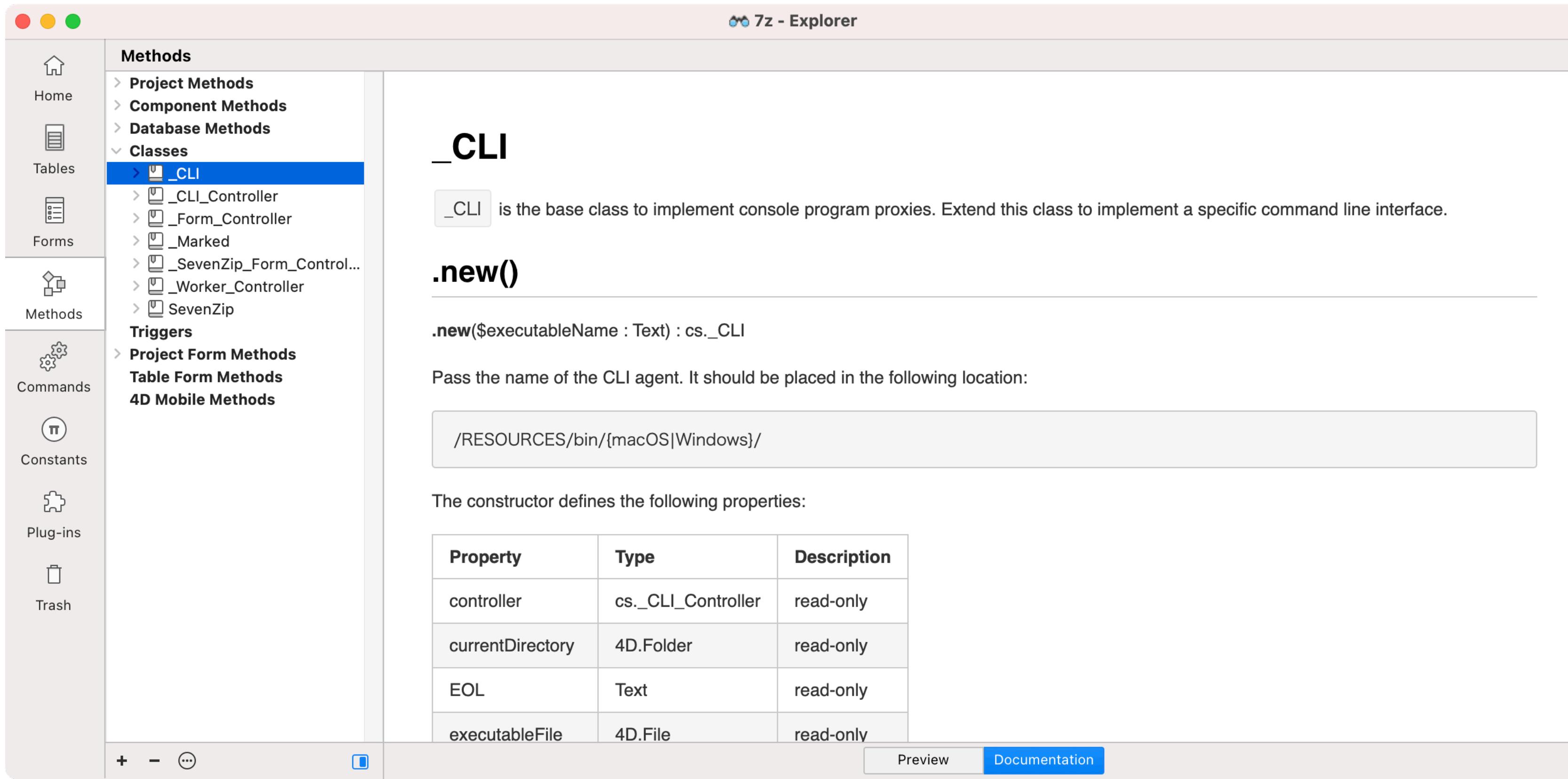
Recap

- **platform agnostic**
 - convert File/Folder object to and from platform path
 - take care of folder separator, end-of-line, UNIX execute bit, file extension
- **flexible API**
 - take advantage of **Variant, Copy parameters**
 - class documentation



Inheritance

Explain the class hierarchy to other developers (and to your future self)

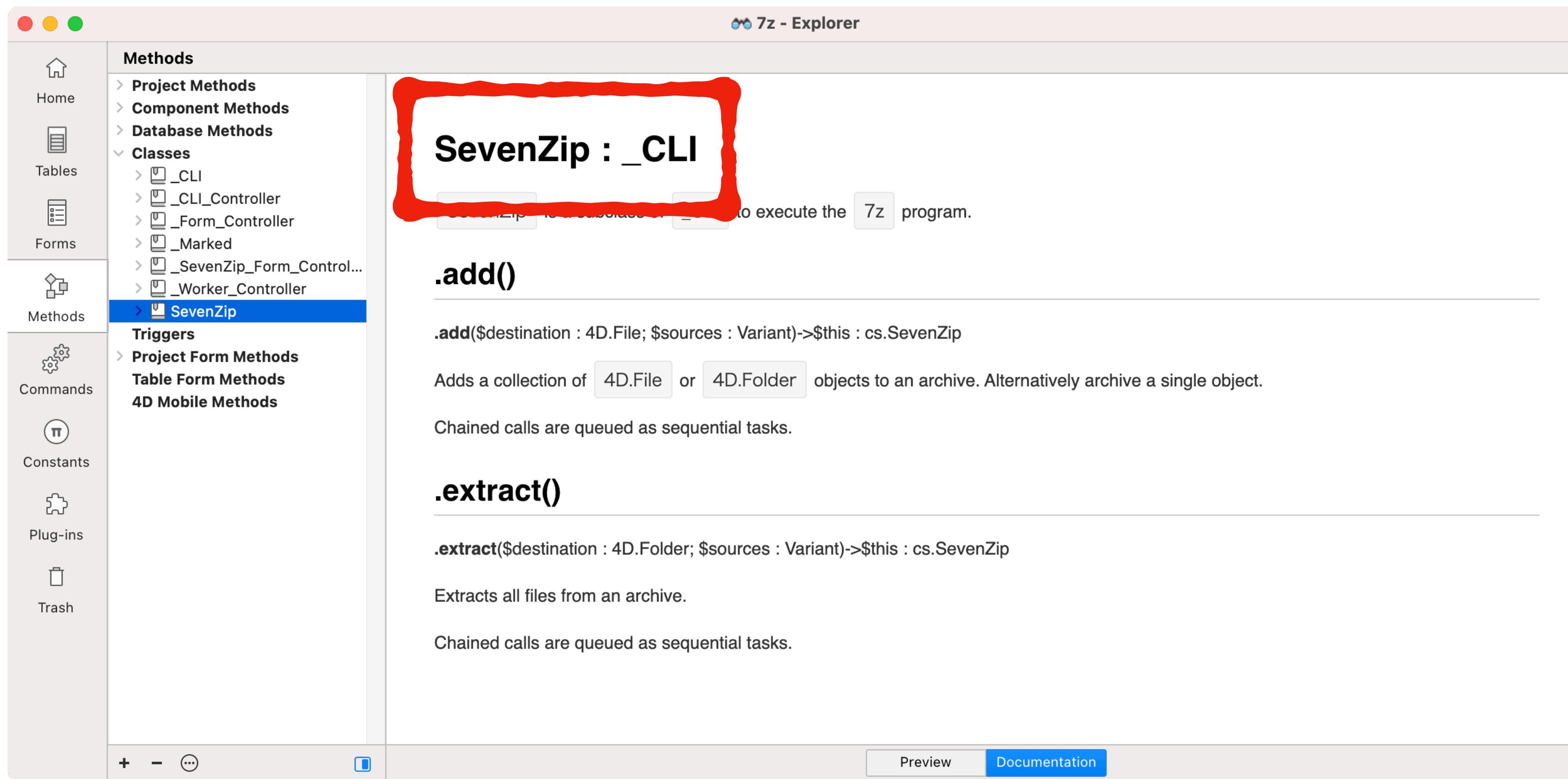


The screenshot shows the 4D Explorer application interface. On the left is a sidebar with icons for Home, Tables, Forms, Methods, Commands, Constants, Plug-ins, and Trash. The 'Methods' section is expanded, showing Project Methods, Component Methods, Database Methods, Classes, Triggers, Project Form Methods, Table Form Methods, and 4D Mobile Methods. Under 'Classes', the `_CLI` class is selected and highlighted with a blue background. The main pane displays the `_CLI` class documentation. The title is `_CLI`. A description states: `_CLI` is the base class to implement console program proxies. Extend this class to implement a specific command line interface. A method `.new()` is shown with the signature `.new($executableName : Text) : cs._CLI`. It is noted that Pass the name of the CLI agent. It should be placed in the following location: `/RESOURCES/bin/{macOS|Windows}/`. Below this, it says The constructor defines the following properties:

Property	Type	Description
controller	cs._CLI_Controller	read-only
currentDirectory	4D.Folder	read-only
EOL	Text	read-only
executableFile	4D.File	read-only

Inheritance

Explain the class hierarchy to other developers (and to your future self)



Abstraction



Atlanta, USA (October 3rd - 4th, 2023) and Paris, France (October 17th - 18th, 2023)

Abstraction

```
$worker:=4D.SystemWorker.new($command,$options)
```

what it this?



Abstraction

`$worker:=4D.SystemWorker.new($command;$options)`



controller

.onResponse

.timeout

.onData

.dataType

.onDataError

.encoding

.onError

.variables

.onTerminate

.currentDirectory

.hideWindow



Abstraction

`$worker:=4D.SystemWorker.new($command;$options)`



controller

Class Constructor

Function `onResponse(4D.SystemWorker; Object)`

Function `onData (4D.SystemWorker; Object)`

Function `onDataError (4D.SystemWorker; Object)`

Function `onError (4D.SystemWorker; Object)`

Function `onTerminate (4D.SystemWorker; Object)`

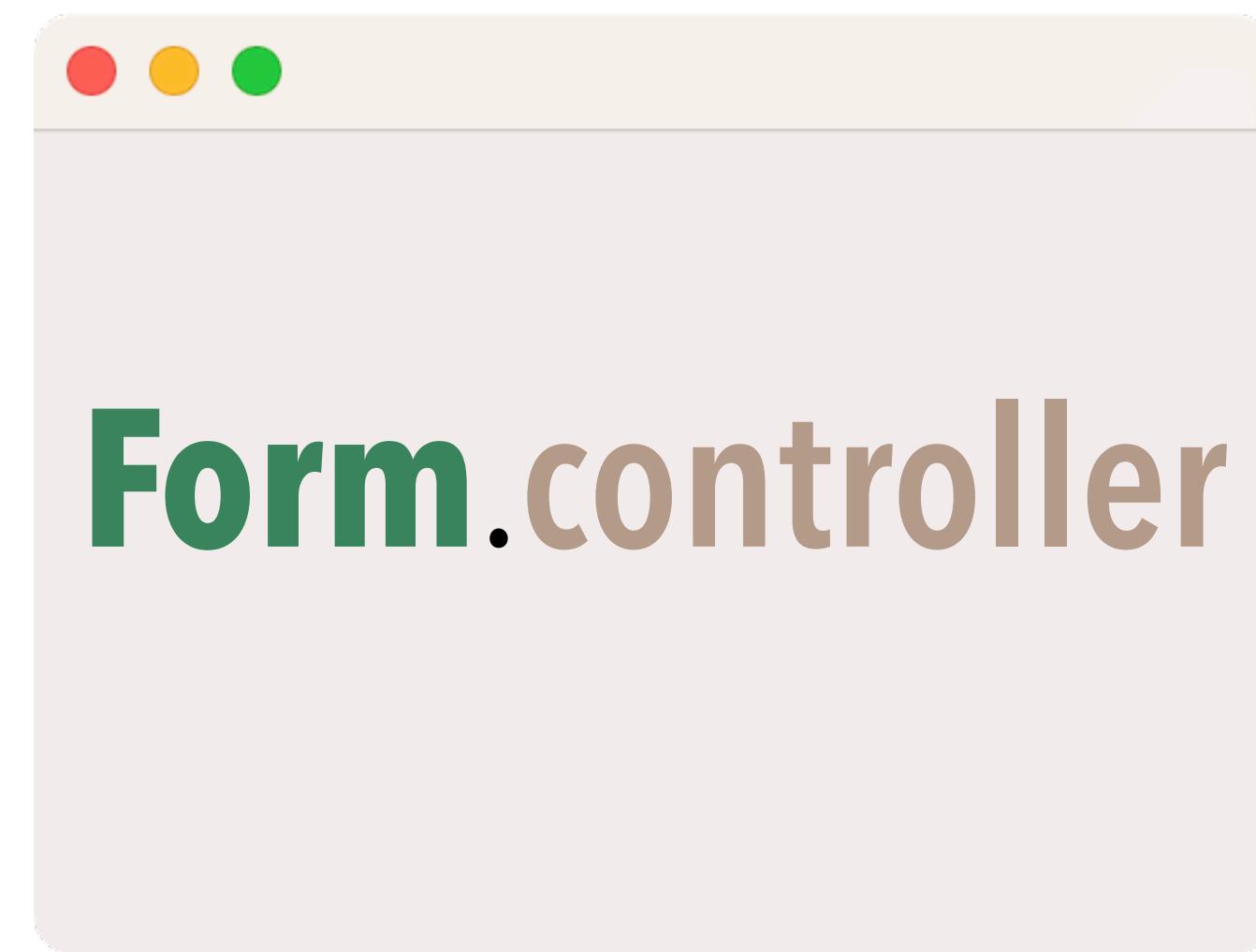


Abstraction

```
$worker:=4D.SystemWorker.new($command;$controller)
```



WORKER



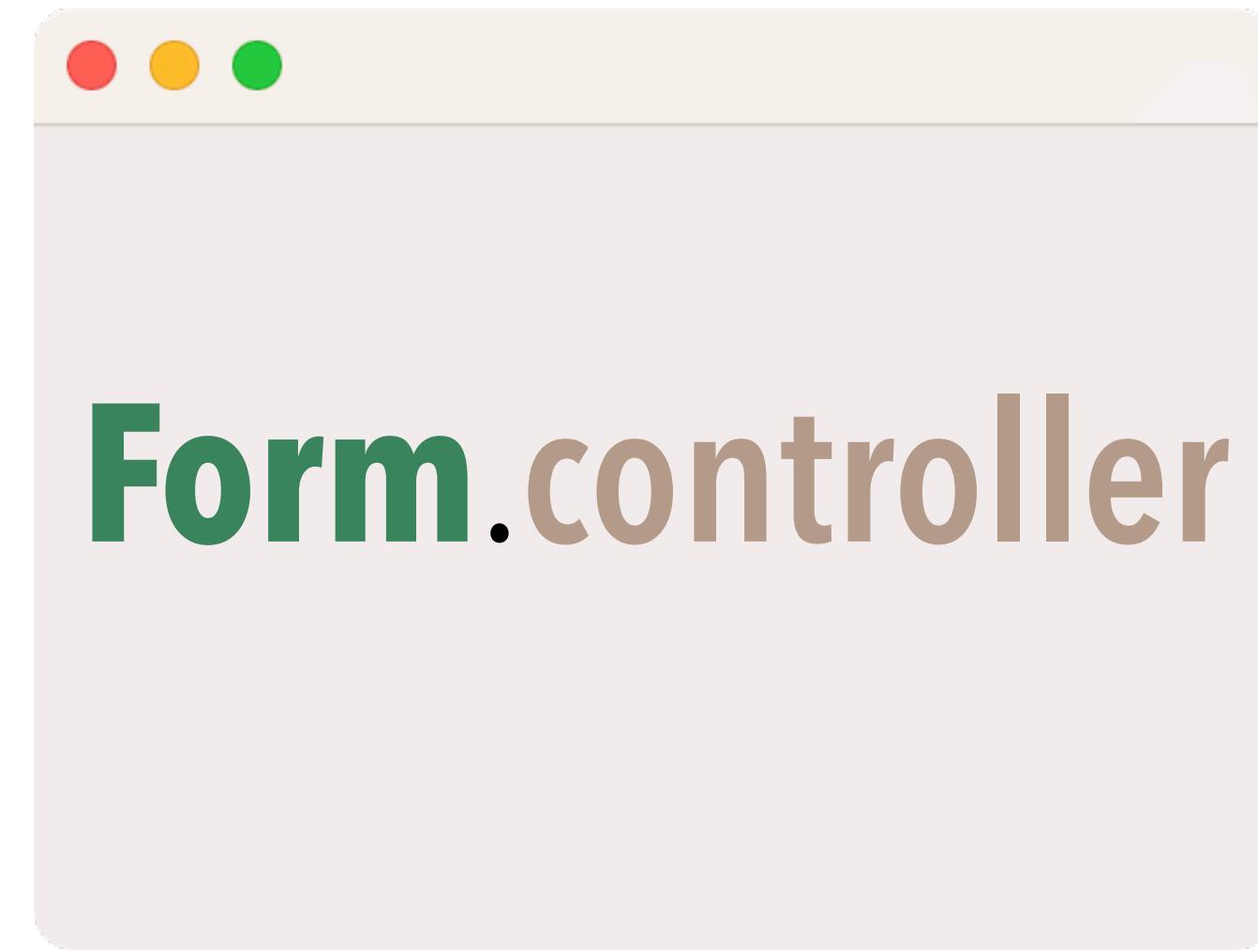
DIALOG

Abstraction

`$worker:=4D.SystemWorker.new($command;$controller)`



WORKER



DIALOG is like a WORKER with UI layer

Abstraction

`$worker:=4D.SystemWorker.new($command;$controller)`

`$controller`

WORKER

`$controller`

`+`

`Form`

DIALOG is like a WORKER with UI layer



Abstraction

object oriented programming

generic

specific



Base class to implement
any controller



Subclass for FORM



Subclass for FORM for 7z



Subclass for WORKER

Demonstration

DIALOG is like a WORKER with UI layer

Recap



Atlanta, USA (October 3rd - 4th, 2023) and Paris, France (October 17th - 18th, 2023)

Recap

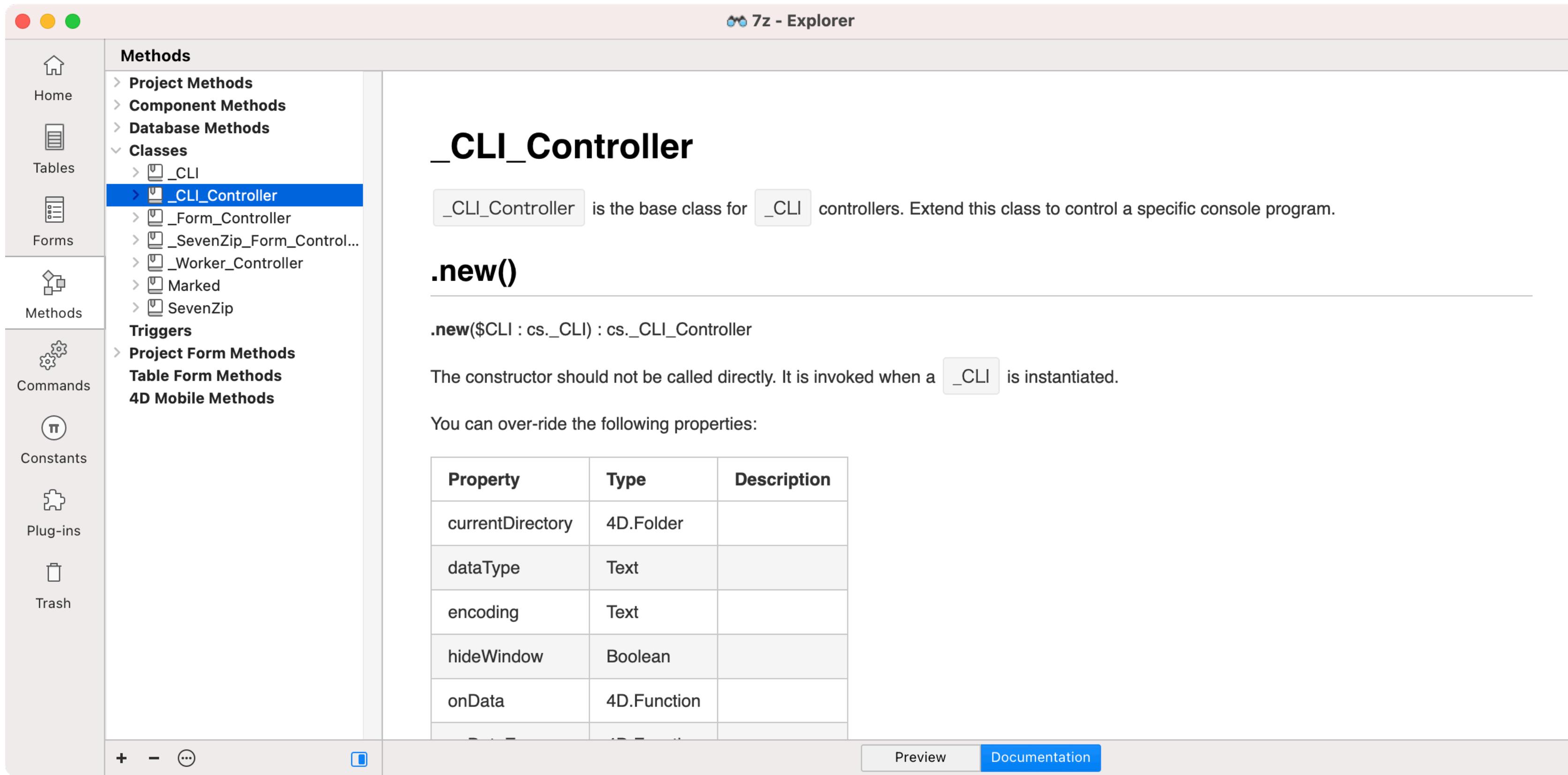
- **abstract super-class** for both WORKER and DIALOG
- **common sub-class** suitable for any form
- **specialised sub-sub-class** for specific forms



Atlanta, USA (October 3rd - 4th, 2023) and Paris, France (October 17th - 18th, 2023)

Abstraction

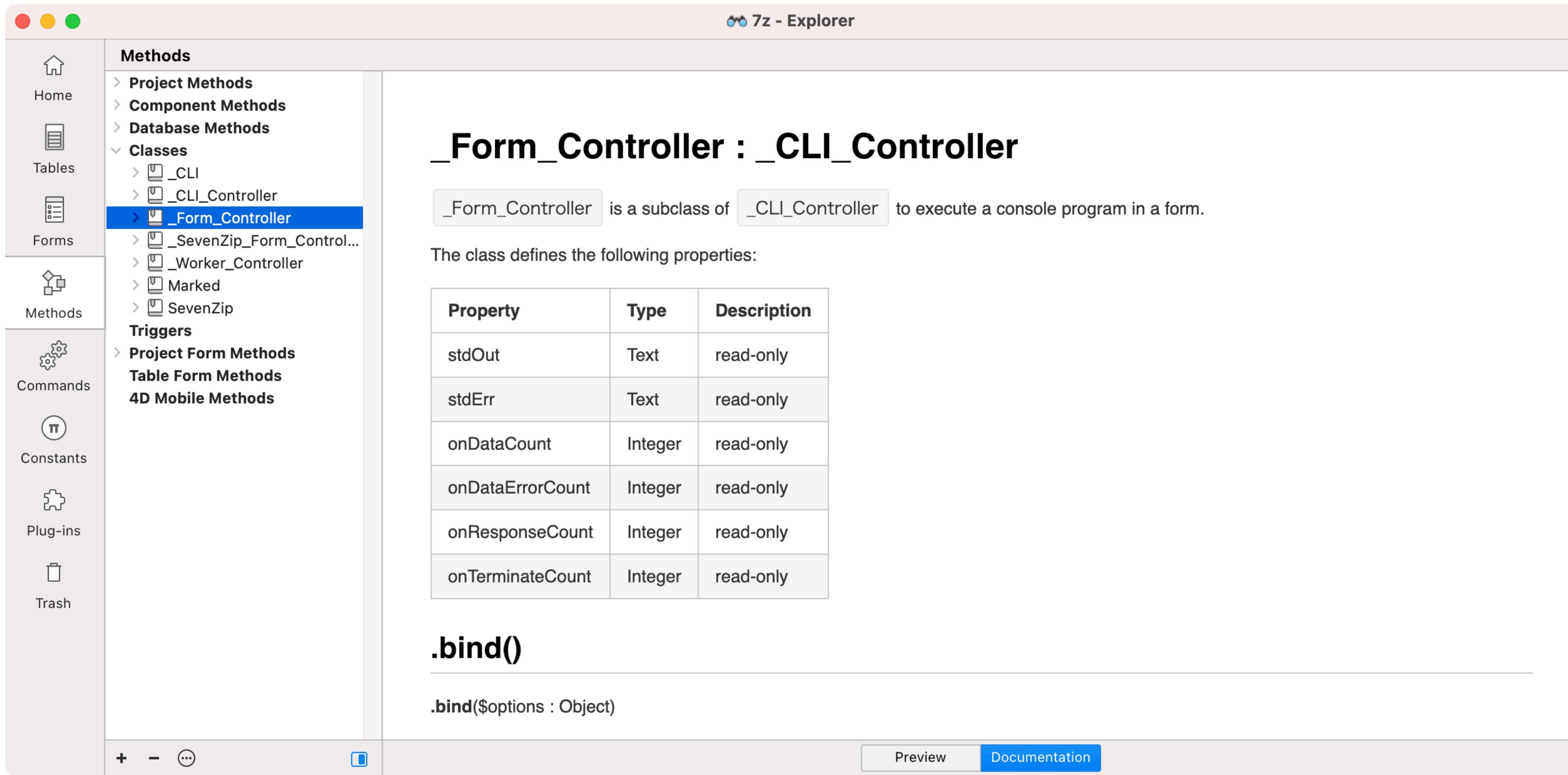
Explain the class hierarchy to other developers (and your future self)



The screenshot shows the 4D Explorer application interface. On the left is a sidebar with icons for Home, Tables, Forms, Methods, Commands, Constants, Plug-ins, and Trash. The 'Methods' section is expanded, showing Project Methods, Component Methods, Database Methods, Classes, Triggers, Project Form Methods, Table Form Methods, and 4D Mobile Methods. Under 'Classes', '_CLI_Controller' is selected and highlighted with a blue background. The main pane displays the documentation for '_CLI_Controller'. The title is '**_CLI_Controller**'. A description states: '_CLI_Controller' is the base class for '_CLI' controllers. Extend this class to control a specific console program. Below this is a section for the constructor '**.new()**' with the signature '.new(\$CLI : cs._CLI) : cs._CLI_Controller'. A note says: The constructor should not be called directly. It is invoked when a '_CLI' is instantiated. There is also a list of over-ridable properties: currentDirectory (4D.Folder), dataType (Text), encoding (Text), hideWindow (Boolean), and onData (4D.Function). At the bottom of the main pane are 'Preview' and 'Documentation' buttons.

Abstraction

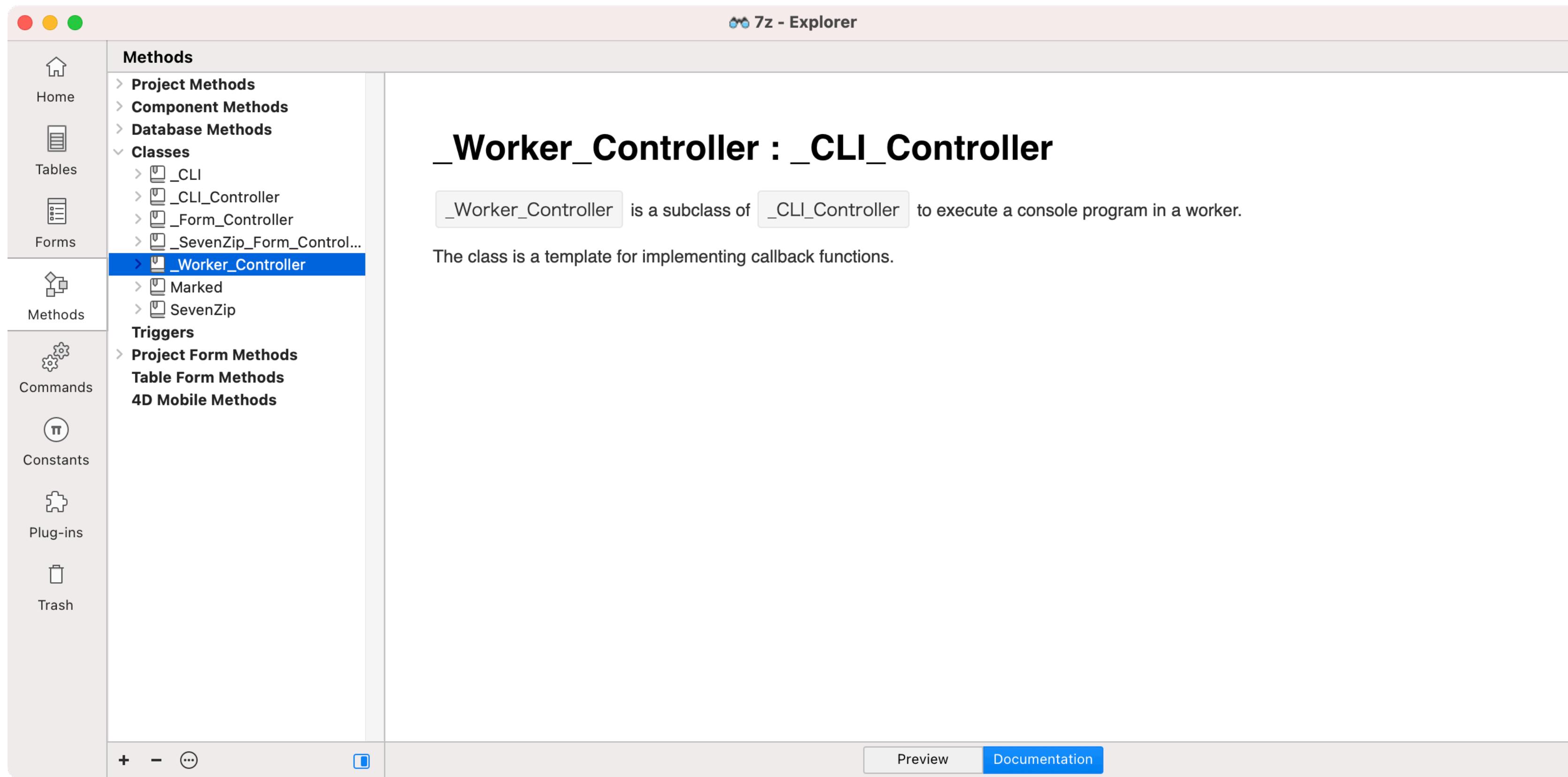
Explain the class hierarchy to other developers (and your future self)



The screenshot shows the 4D Explorer application interface. The left sidebar contains navigation links: Home, Tables, Forms, Methods, Commands, Constants, Plug-ins, and Trash. The main pane displays the class hierarchy under the 'Methods' section. The tree view shows 'Project Methods', 'Component Methods', 'Database Methods', 'Classes' (expanded), '_CLI' (expanded), '_CLI_Controller' (expanded), and '_Form_Controller' (selected). Other visible classes include '_SevenZip_Form_Controller...', '_Worker_Controller', 'Marked', and 'SevenZip'. Below the tree, sections for 'Triggers', 'Project Form Methods', 'Table Form Methods', and '4D Mobile Methods' are listed. The right pane details the selected class, **_Form_Controller : _CLI_Controller**. It states that **_Form_Controller** is a subclass of **_CLI_Controller** to execute a console program in a form. It also lists properties: stdOut (Text, read-only), stdErr (Text, read-only), onDataCount (Integer, read-only), onDataErrorCode (Integer, read-only), onResponseCount (Integer, read-only), and onTerminateCount (Integer, read-only). A method section for **.bind()** is shown with the signature **.bind(\$options : Object)**. At the bottom, there are 'Preview' and 'Documentation' buttons.

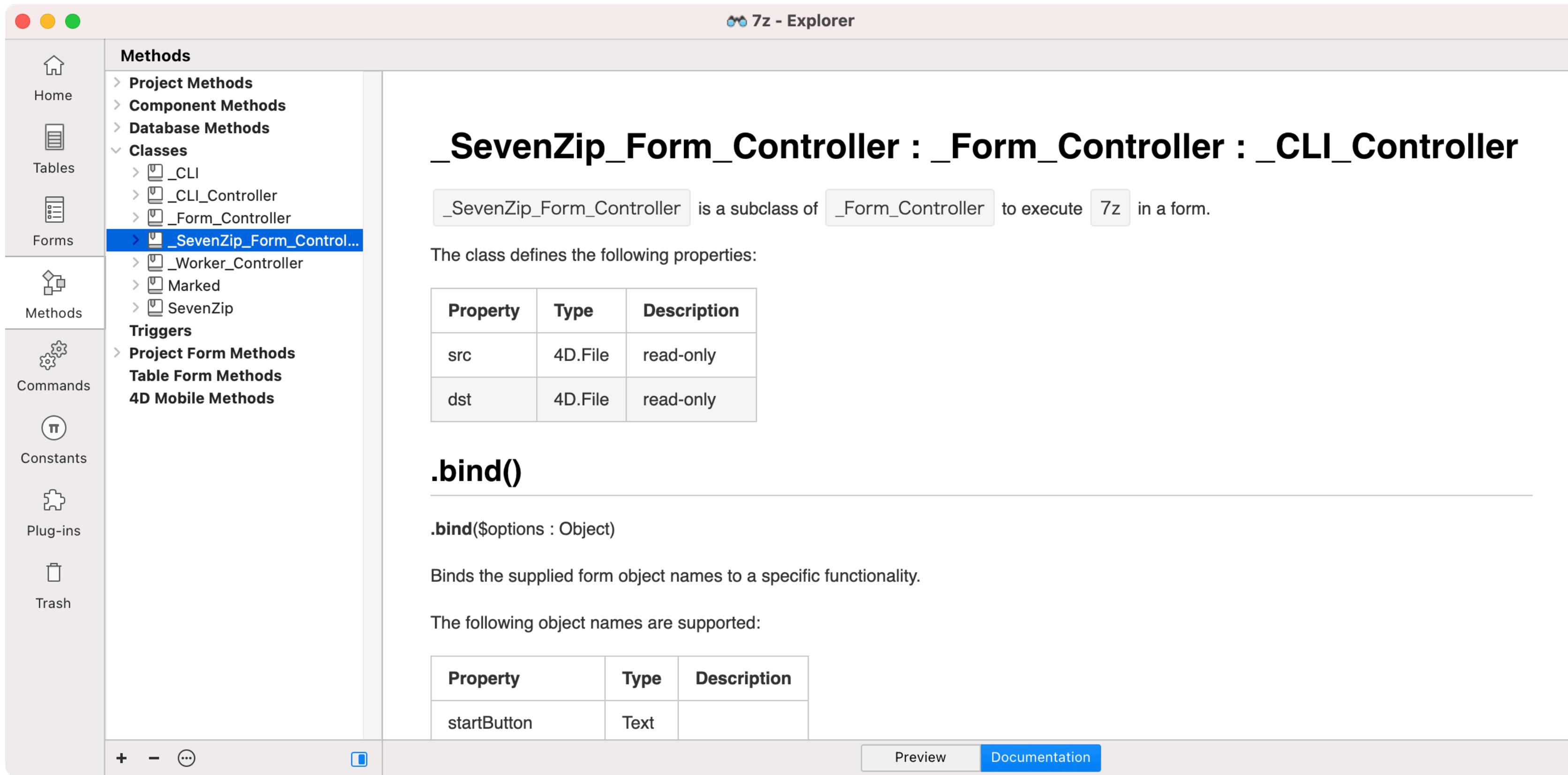
Abstraction

Explain the class hierarchy to other developers (and your future self)



Abstraction

Explain the class hierarchy to other developers (and your future self)



The screenshot shows the 4D Explorer interface with the title bar "7z - Explorer". The left sidebar contains navigation links: Home, Tables, Forms, Methods, Commands, Constants, Plug-ins, and Trash. The main pane displays the class hierarchy under the "Methods" section:

- > Project Methods
- > Component Methods
- > Database Methods
- > Classes
 - > _CLI
 - > _CLI_Controller
 - > _Form_Controller
 - > **_SevenZip_Form_Controller**
- > Triggers
- > Project Form Methods
- Table Form Methods
- 4D Mobile Methods

The class **_SevenZip_Form_Controller** is selected and highlighted with a blue background. The right pane provides documentation for this class:

_SevenZip_Form_Controller : _Form_Controller : _CLI_Controller

_SevenZip_Form_Controller is a subclass of **_Form_Controller** to execute **7z** in a form.

The class defines the following properties:

Property	Type	Description
src	4D.File	read-only
dst	4D.File	read-only

.bind()

.bind(\$options : Object)

Binds the supplied form object names to a specific functionality.

The following object names are supported:

Property	Type	Description
startButton	Text	

Buttons at the bottom of the right pane include "Preview" and "Documentation".

Encapsulation



Atlanta, USA (October 3rd - 4th, 2023) and Paris, France (October 17th - 18th, 2023)

Encapsulation

```
Class constructor($executableName : Text; $controller : 4D.Class)
```

```
This._name:=0B Class(This).name
```

```
Case of
```

```
: (Is macOS)
```

```
    This._platform:="macOS"
```

```
    This._executableName:=$executableName
```

```
    This._EOL:="\n"
```

```
: (Is Windows)
```

```
    This._platform:="Windows"
```

```
    This._executableName:=$executableName+".exe"
```

```
    This._EOL:="\r\n"
```

```
End case
```



Encapsulation

Class constructor(\$executableName : Text; \$controller : 4D.Class)

This._name

This._platform

This._executableName

This._EOL

This._platform

This._executableName

This._EOL

Encapsulation

Class constructor(\$executableName : **Text**; \$controller : **4D.Class**)

This._name:=0B Class(This).name

Function get name() -> \$name : Text

\$name:=This._name

 **get name**
read-only property



Encapsulation

- underscore-prefixed members are **excluded** from **autocomplete**
 - the purpose is to **filter** private or internal properties and methods in **development**
 - available in native method editor and LSP compatible external editor
- hiding doesn't protect data or restrict access
 - values are accessible from **debugger**

Expression	Value
▀ \$marked : Marked	{"_platform":"macOS","_cont...,"area":"Marked","onEvent":{}}
> \$marked._contentsFolder : Folder	Folder: /Applications/4D v20.0/100522/4D.app/Contents/
> \$marked._documentationFolder : Folder	Folder: /PACKAGE/Documentation/
> \$marked._indexPageName : Text	"index.html"
> \$marked._libraryRootFolder : Folder	Folder: /Applications/4D v20....se/Resources/documentation/
> \$marked._platform : Text	"macOS"
> \$marked._rootFolderName : Text	"documentation"
> \$marked.area : Text	"Marked"
> \$marked.onEvent : Function	<<User Function: _Marked._onEvent >>
> \$marked.url : File	File: /Applications/4D v20.0/1...ces/documentation/index.html

Encapsulation

 **constructor**

 **private**

 **createTempFolder**

 **getDocumentationFile**

 **getFormFile**

 **getIndexPage**

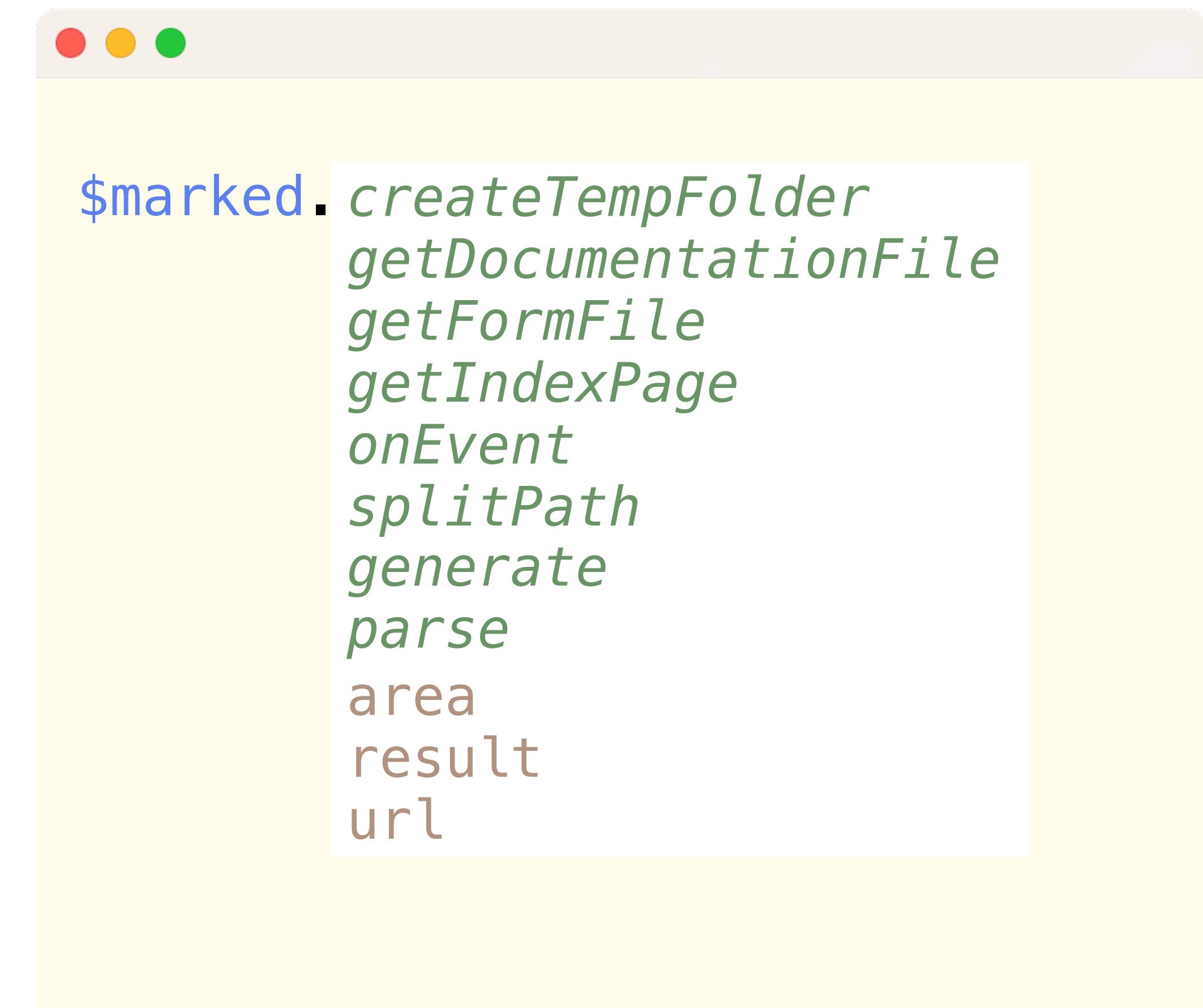
 **onEvent**

 **splitPath**

 **public**

 **generate**

 **parse**



Encapsulation

 **constructor**

 **private**

 **_createTempFolder**

 **_getDocumentationFile**

 **_getFormFile**

 **_getIndexPage**

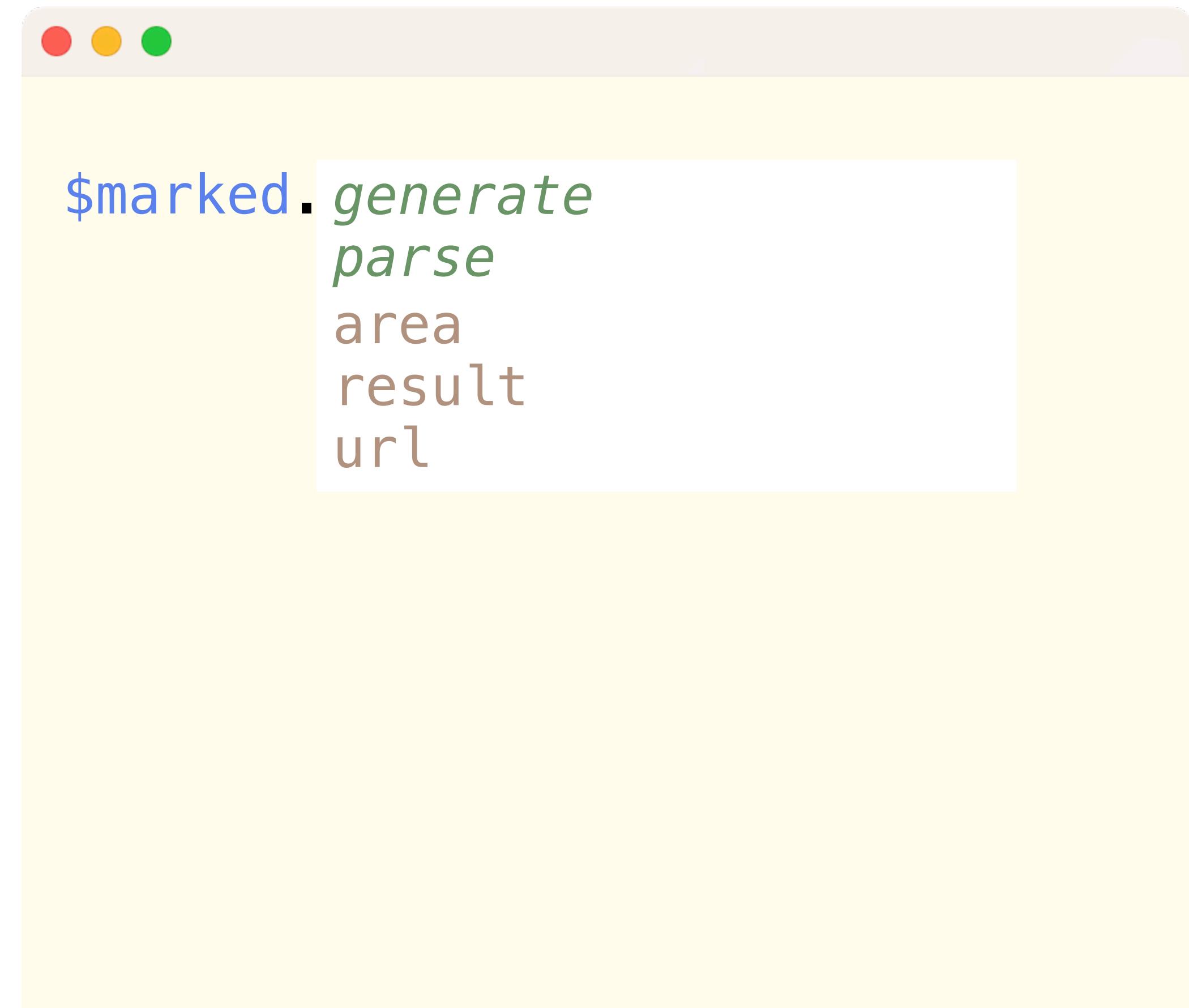
 **_onEvent**

 **_splitPath**

 **public**

 **generate**

 **parse**



Encapsulation

object oriented programming

_CLI

base abstract class

cannot be used directly

_Worker_Controller

extended generic class

can be used with any CLI

SevenZip

public class

constructor instantiates **default controller**

_CLI_Controller

base generic class

can be used directly (**default controller**)

_Form_Controller

extended generic class

can be used with any CLI

_SevenZip_Form_Controller

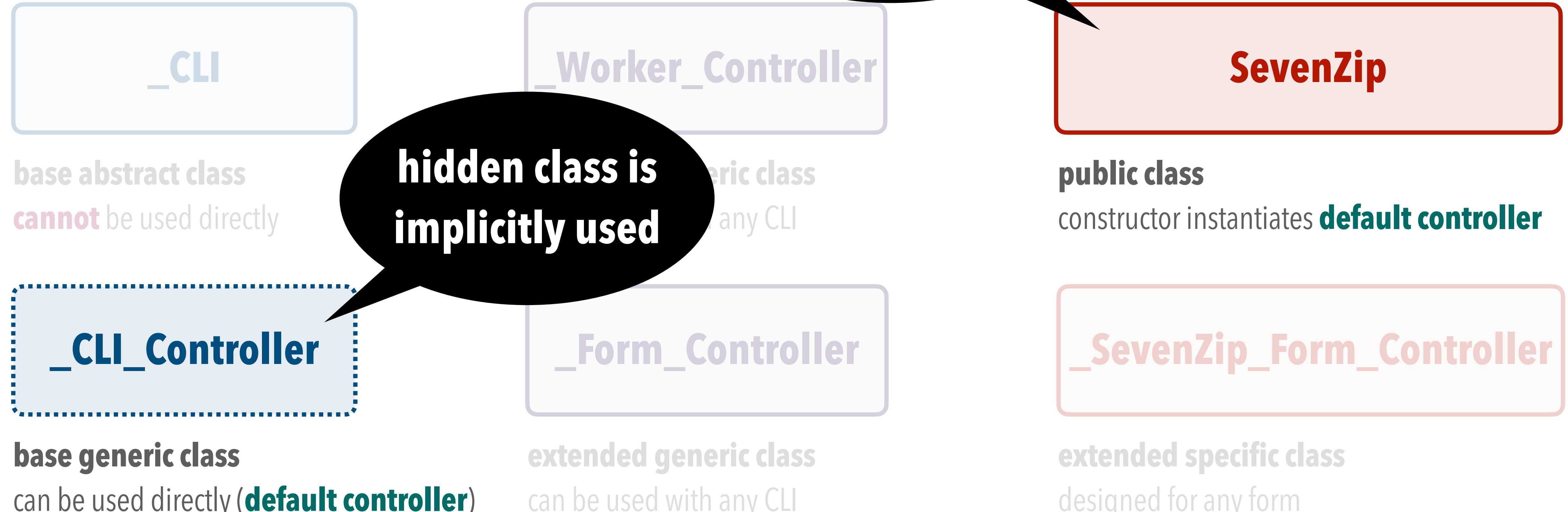
extended specific class

designed for any form



Encapsulation

object oriented programming



Encapsulation

private classes

The screenshot shows the 4D mobile application interface. On the left, a sidebar navigation menu includes Home, Tables, Forms, Methods, Commands, Constants, Plug-ins, and Trash. The Methods section is expanded, showing Project Methods, Component Methods, Database Methods, and Classes. The Classes section is also expanded, listing _CLI, _CLI_Controller, _Form_Controller, _Marked, _SevenZip_Form_Controller, and _Worker_Controller. The main content area displays the details for the _CLI class. It shows the class name, a note that it is the base class, a new() method with a parameter (\$executableName), and a constructor definition (/RESOURCES/bin/[...]). A table below lists properties: controller, currentDirectory, EOL, and executableFile. To the right of the main content is a sidebar titled "Methods" which lists Project Methods, Component Methods, Database Methods, and Classes, with the Classes section expanded to show the same list of classes as the sidebar.

_CLI

_CLI is the base class

.new()

.new(\$executableName)

Pass the name of the CLI

/RESOURCES/bin/[...]

The constructor defines the following properties:

Property	Type
controller	Object
currentDirectory	String
EOL	Text
executableFile	String

Methods

- > Project Methods
- > Component Methods
- > Database Methods
- > Classes
 - > _CLI
 - > _CLI_Controller
 - > _Form_Controller
 - > _Marked
 - > _SevenZip_Form_Controller...
 - > _Worker_Controller

Tables

Forms

Methods

Commands

Triggers

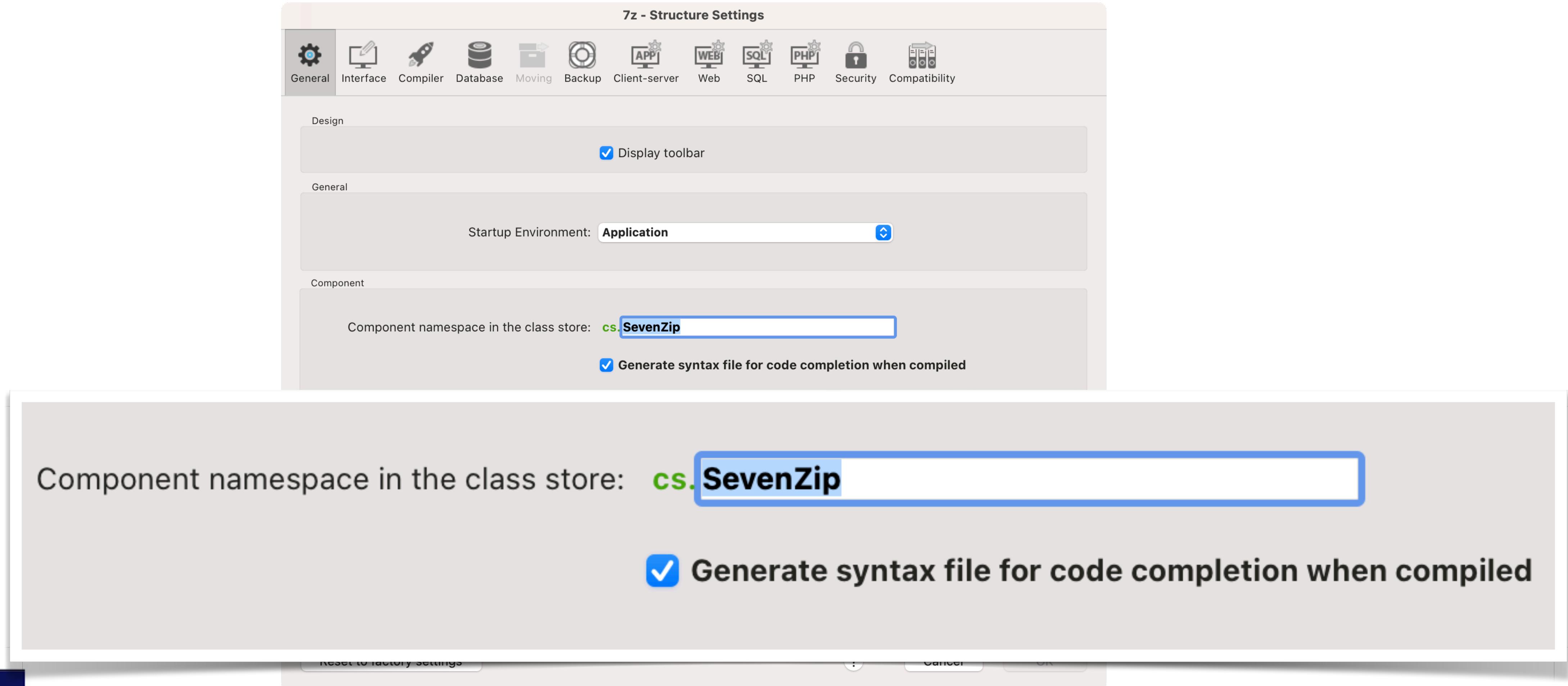
> Project Form Methods

Table Form Methods

4D Mobile Methods

Encapsulation

component namespace

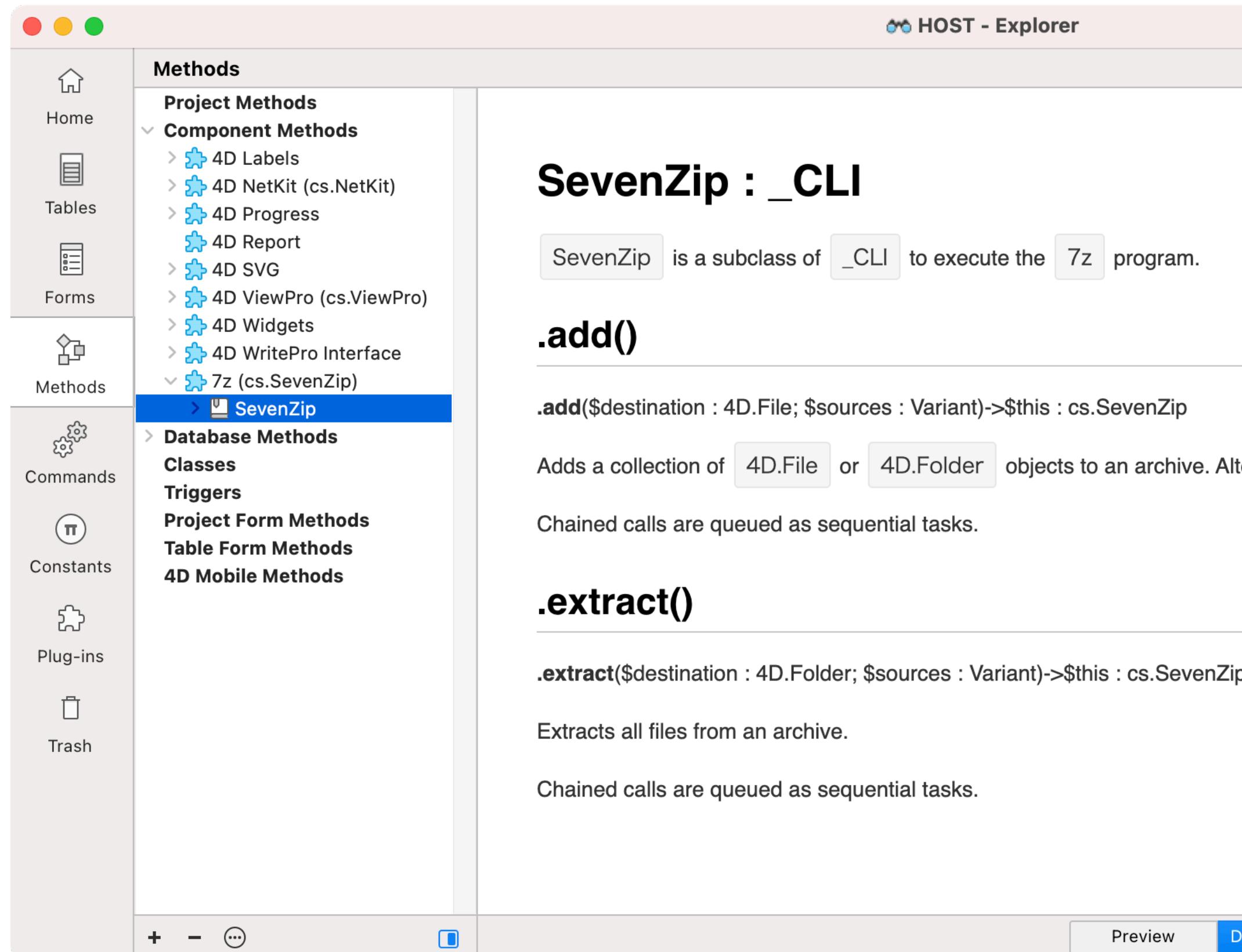


Component namespace in the class store: `cs.SevenZip`

Generate syntax file for code completion when compiled

Encapsulation

public class



The screenshot shows the 4D Explorer interface with the title bar "HOST - Explorer". The left sidebar has icons for Home, Tables, Forms, Methods, Commands, Constants, Plug-ins, and Trash. The "Methods" section is expanded, showing "Project Methods" and "Component Methods". Under "Component Methods", there is a list of components: 4D Labels, 4D NetKit (cs.NetKit), 4D Progress, 4D Report, 4D SVG, 4D ViewPro (cs.ViewPro), 4D Widgets, 4D WritePro Interface, and 7z (cs.SevenZip). The "7z (cs.SevenZip)" item has its submenu expanded, with "SevenZip" highlighted. The main pane displays the "SevenZip : _CLI" class. It defines "SevenZip" as a subclass of "_CLI" to execute the "7z" program. It contains two methods: ".add()" and ".extract()". The ".add()" method adds a collection of 4D.File or 4D.Folder objects to an archive. The ".extract()" method extracts all files from an archive.

Methods

Project Methods

Component Methods

- > **4D Labels**
- > **4D NetKit (cs.NetKit)**
- > **4D Progress**
- > **4D Report**
- > **4D SVG**
- > **4D ViewPro (cs.ViewPro)**
- > **4D Widgets**
- > **4D WritePro Interface**
- > **7z (cs.SevenZip)**

Database Methods

Classes

Triggers

Project Form Methods

Demonstration

hidden class, property, method + Marked.js

Recap



Atlanta, USA (October 3rd - 4th, 2023) and Paris, France (October 17th - 18th, 2023)

Recap

- **access control:** 4D does **not** have private, public, protected
 - underscore prefix **hides** information
- **read-only:** a property can have a **get function** only
- **public:** share component classes with **namespace** (also available in binary mode)

Polymorphism



Atlanta, USA (October 3rd - 4th, 2023) and Paris, France (October 17th - 18th, 2023)

Inheritance

object oriented programming

generic

specific

_CLI_Controller

.execute

push CLI to queue

_Form_Controller

.execute

push CLI to queue

_SevenZip_Form_Controller

.execute

push CLI to queue

_Worker_Controller

.execute

push CLI to queue

methods and properties are
inherited from super-class



Inheritance

object oriented programming

generic

specific

_CLI_Controller

.complete (r/o)

True if queue is empty

_Form_Controller

.complete (r/o)

True if queue is empty

_SevenZip_Form_Controller

.complete (r/o)

True if queue is empty

_Worker_Controller

.complete (r/o)

True if queue is empty

methods and properties are
inherited from super-class

Inheritance

object oriented programming

generic

specific

`_CLI_Controller`

`_Form_Controller`

`_SevenZip_Form_Controller`



`.bind`
specify start/stop buttons

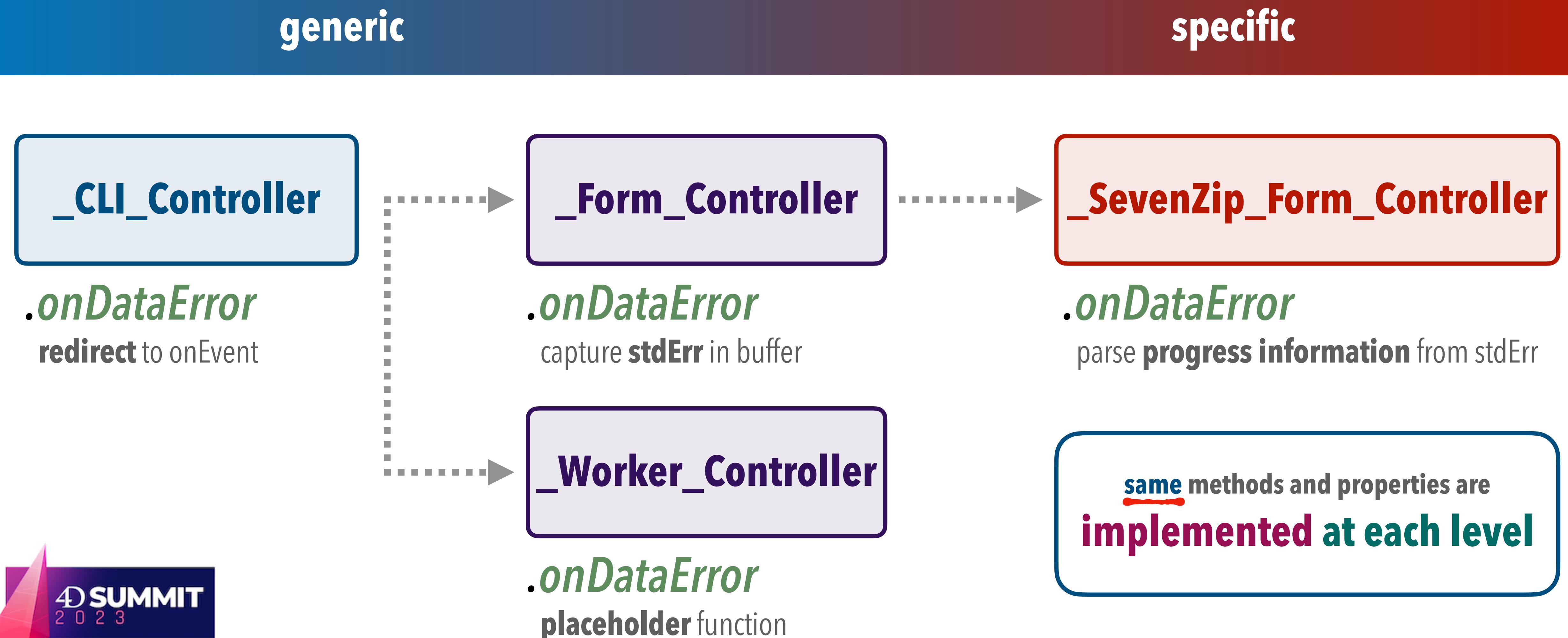
`_Worker_Controller`

`.progress` (r/o)
progress information in percentage

more methods and properties are
implemented at each level

Polymorphism

object oriented programming



Demonstration

System Worker and polymorphism

Recap



Atlanta, USA (October 3rd - 4th, 2023) and Paris, France (October 17th - 18th, 2023)

Recap

- **callback**: different needs for WORKER vs DIALOG
 - DIALOG has **form objects**; WORKER does not
 - DIALOG must check for the **Form** object
- **progress**: real-time update for DIALOG; log recording for WORKER
- **interruption**: stop button for DIALOG; timeout option for WORKER



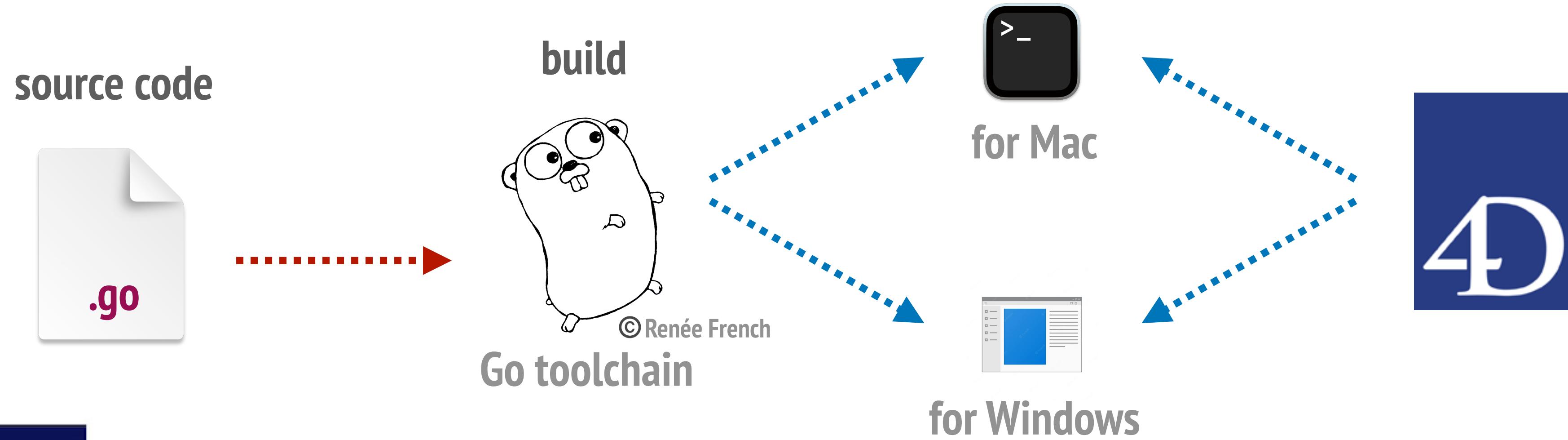
Extra

Go Primer

The Go Programming Language

<https://golang.org/>

- **modern**: statically typed, compiled high-level language
- **cross-platform**: supports Windows and macOS
- **portable**: generates statically linked binary executable



The Go Programming Language

<https://golang.org/>

- **popular:** active developer community
 - tutorials, libraries, examples
- **performance:** high-speed execution, efficient memory usage
- **formulaic:** consistent language design, ideal for AI assisted code

To convert JSON data to Apache Parquet format in Go, you can use the `parquet-go` library. Unfortunately, as of my knowledge cutoff in September 2021, there is no official Go implementation of Apache Parquet. However, there are third-party libraries like `parquet-go` available for this purpose. To get started, you'll need to...

write me code in go that
converts json to apache paquet



Install Go - Windows

<https://go.dev/doc/install>

installer option

Download (1.20.7)

package manager option



Install Go - macOS

<https://go.dev/doc/install>

installer option

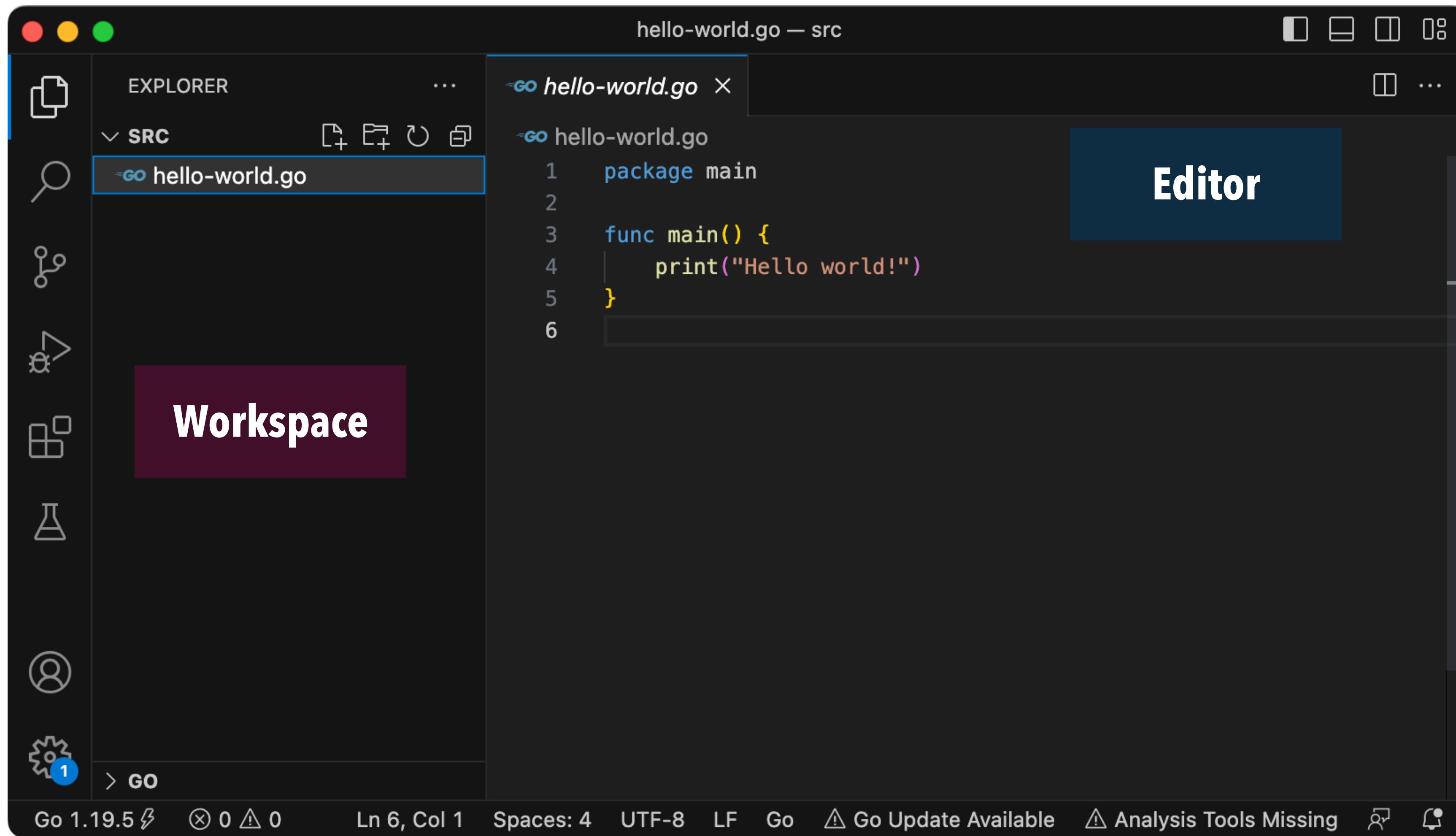
Download (1.20.7)

package manager option



Homebrew

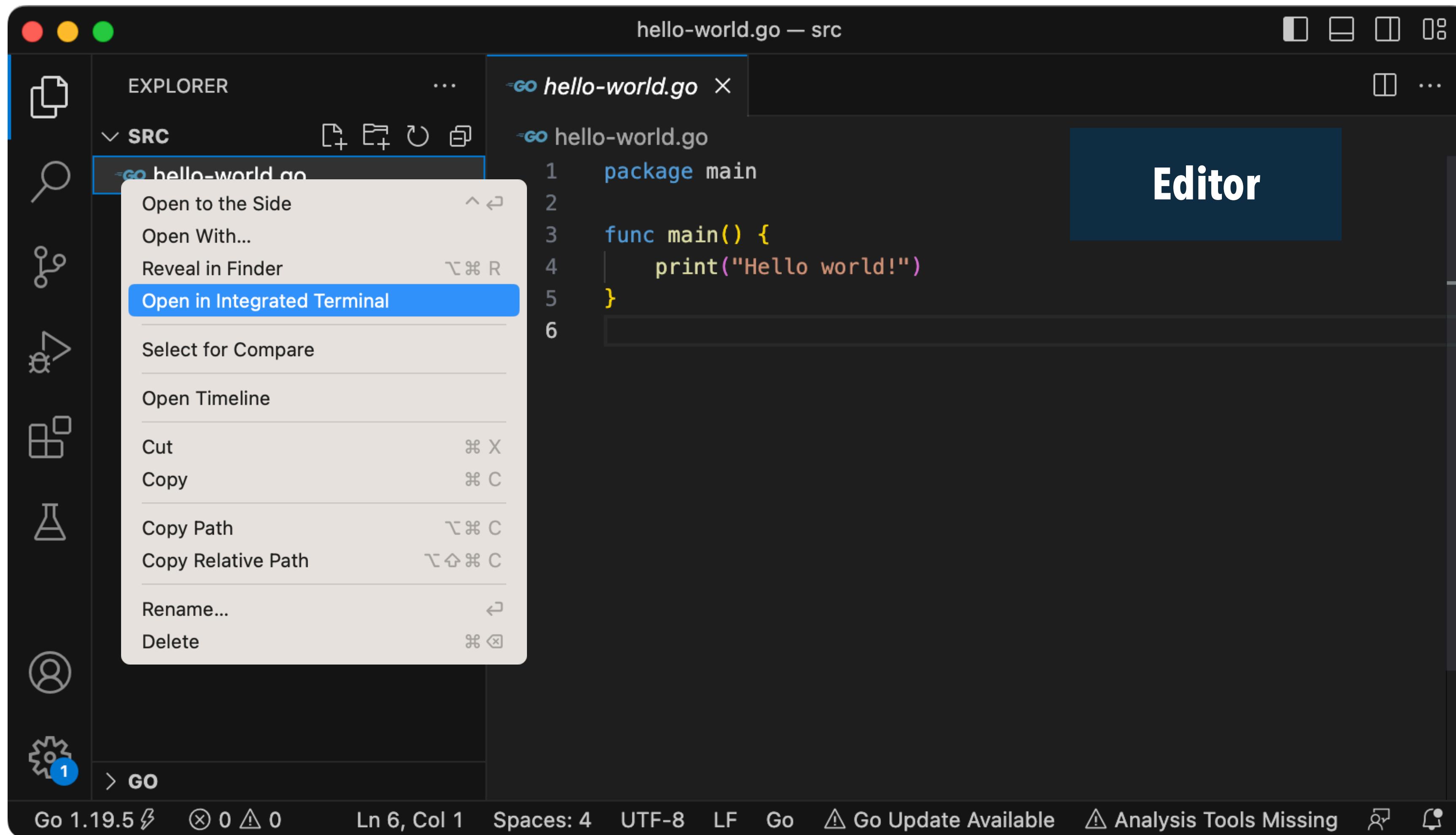
Go for Visual Studio Code



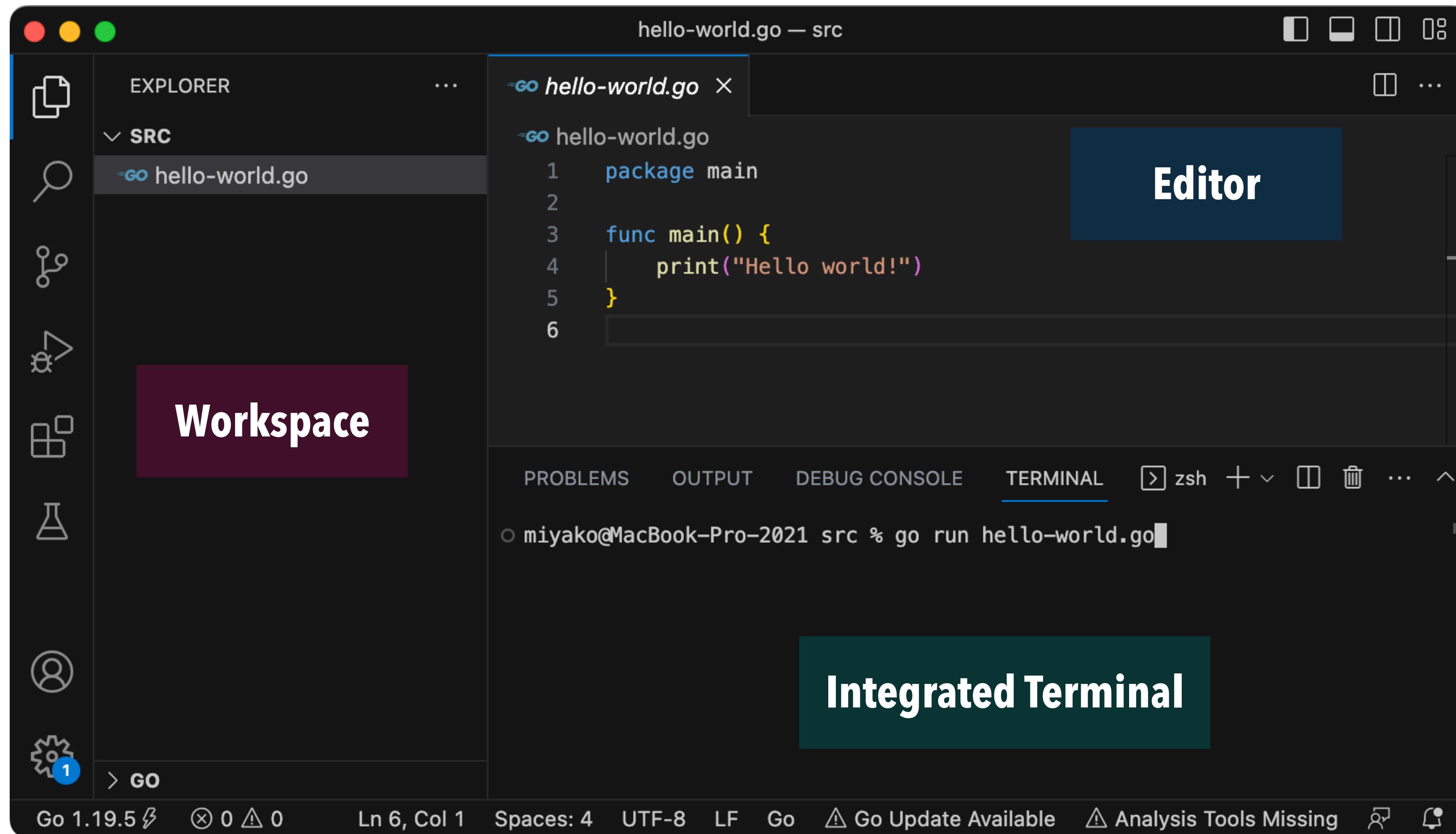
4D SUMMIT
2023

Atlanta, USA (October 3rd - 4th, 2023) and Paris, France (October 17th - 18th, 2023)

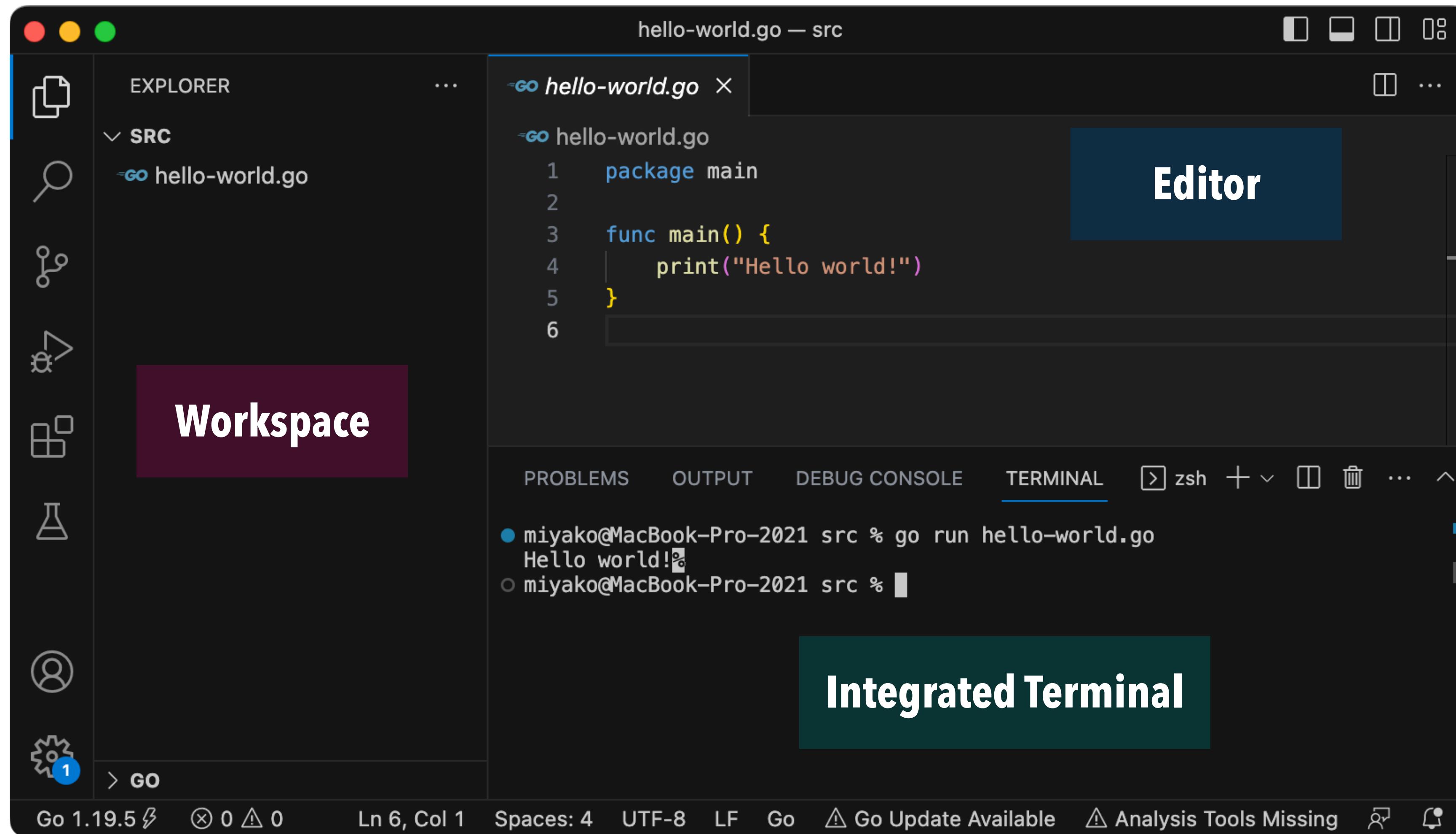
Go for Visual Studio Code



Go for Visual Studio Code



Go for Visual Studio Code



Build Go - Windows

```
go build hello-world.go
```



Atlanta, USA (October 3rd - 4th, 2023) and Paris, France (October 17th - 18th, 2023)

Build Go - macOS

```
GOOS=darwin GOARCH=amd64 go build -o  
hello-world_amd64 hello-world.go
```

```
GOOS=darwin GOARCH=arm64 go build -o  
hello-world_arm64 hello-world.go
```

```
lipo -create -output hello-world  
hello-world_amd64 hello-world_arm64
```



Demonstration

System Worker is ready to Go

The Go Programming Language

<https://golang.org/>

- read/write **Microsoft Excel** spreadsheets

<https://github.com/qax-os/excelize>

- generate **responsive HTML e-mail**

<https://github.com/matcornic/hermes>

- generate **PDF**

<https://github.com/signintech/gopdf>



<https://github.com/avelino/awesome-go>

Question Time

System Workers, OOP, Go

Go places with
System Workers!

4D Summit 2023 presentation