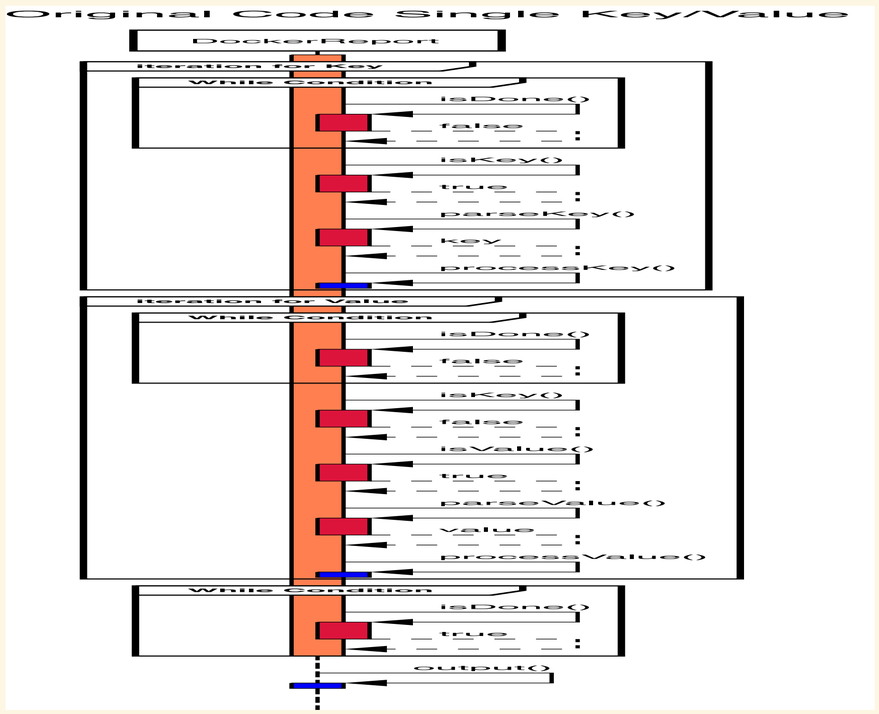
Object-Oriented Programming

IoC

Michael L. Collard, Ph.D.

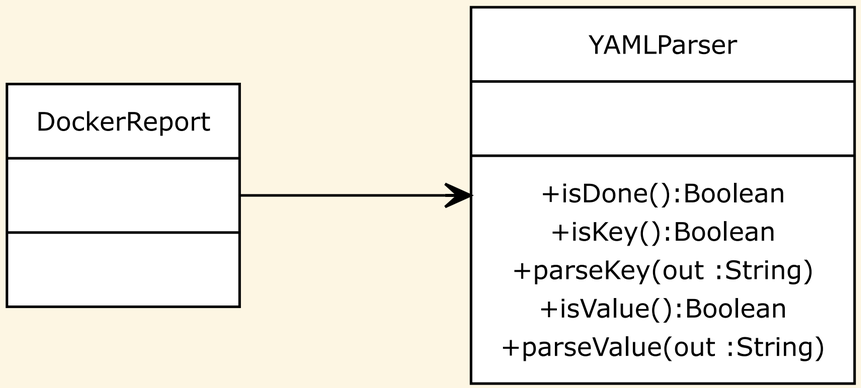
Department of Computer Science, The University of Akron

Sequence Diagram: Original



- Scenario: Single key/value
- Docker Report code
- Low-level YAML parsing code
- YAML parsing loop code
- Full Diagram

UML Class Diagram: Current YAMLParser



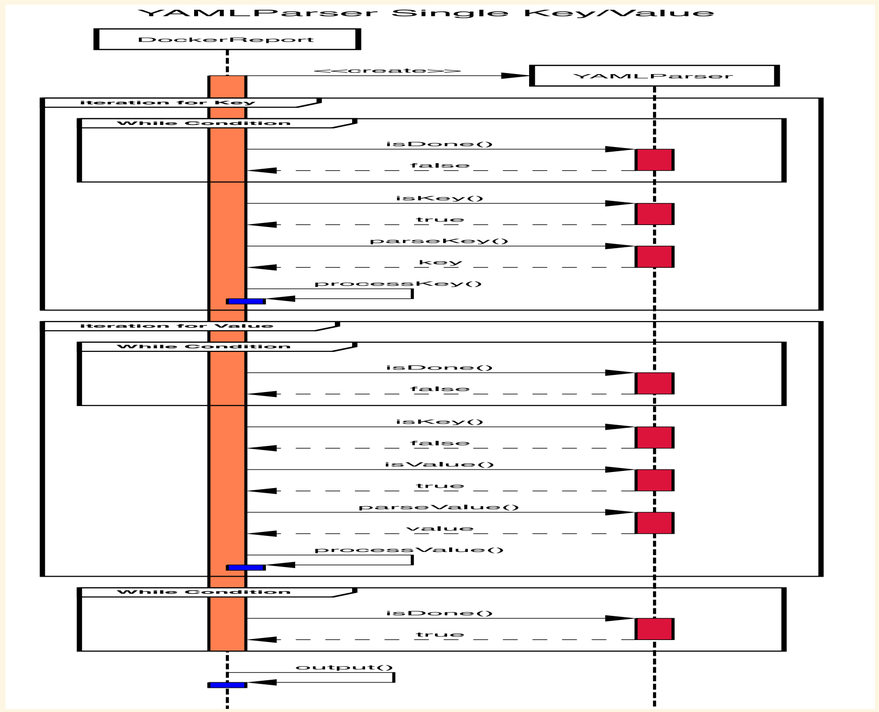
- The multiple public methods correspond to specific predicates and parsing

- Requires complex use by the client (DockerReport)

- Low-level parsing is encapsulated and follows information hiding

- High-level code for parsing is still in the application

Sequence Diagram: YAMLParser



- Scenario: Single key/value
- Docker Report
- Low-level YAML parsing
- YAML parsing loop
- Many YAML concerns still exist in DockerReport
- Cannot move loop concern to YAMLParser because of repeated calls to code with DockerReport concerns
- Full Diagram
- Compare to Original

Inversion of Control (IoC)

- Inverts part of the control flow of the program
- Instead of your code calling a function/method, you *register* parts of your code, and the function/method calls those parts of your code at the appropriate time
- Similar to much *event-driven programming*
- For low-level functionality with a function, typically called a *callback*
- May be a single method or an entire object
- Essential for *frameworks*

Extension Points

- What points in the program can you register a handler for

- Every extension point has a single handler

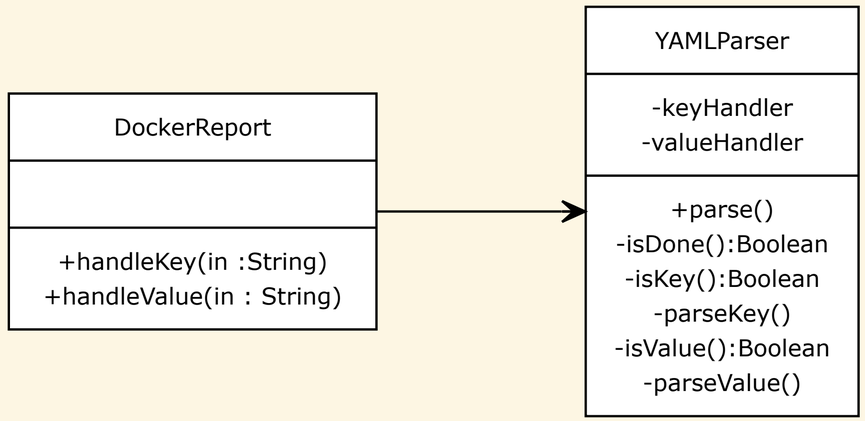
- The handler is passed all data as parameters

- In all our examples, the parameters are IN, but with some IoC, the parameters can be OUT or IN/OUT

YAMLParser Extension Points

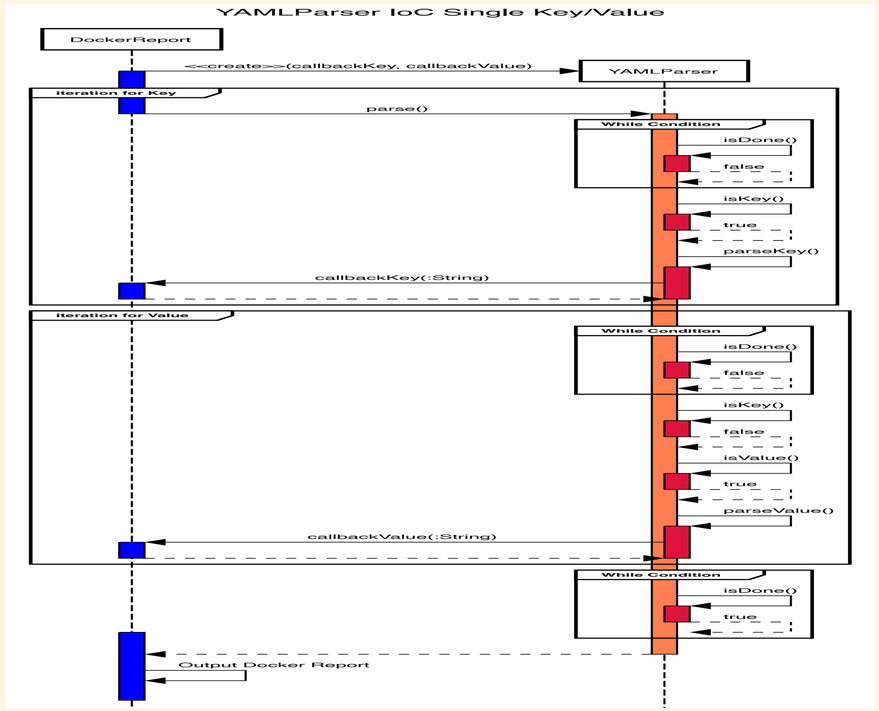
- Start Document
- Key: *name*
- Value: *value*
- End Document

UML Class Diagram: YAMLParser IoC



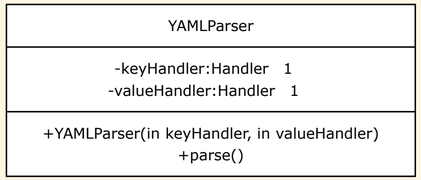
- Single public method, `parse()` is the only one the client uses to parse the YAML
- All predicates and parsing methods are now private
- Straightforward use by the client (DockerReport)
- Low-level parsing is encapsulated and follows information hiding
- High-level code for parsing is in the `parse()` method

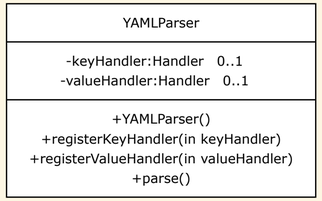
IoC YAMLParser



- Scenario: Single key/value
- Docker Report
- Low-level YAML parsing
- YAML parsing loop
- Full Diagram
- Compare to non-IoC

IoC YAMLParser: Registration





Adding Handlers

Type	Example	Notes

function pointers	`void process(void(*output)(int));`
	
	`void (*output)(int) = [](int n) { std::cout << n; };`
		Limited to free functions (empty lambda capture)
		Very fast

`std::function`	`void f(std::function<void(int)>);`
	
	`std::function<void(int)> output = [&counter](int n) {`
	`  std::cout << n; ++counter;`
	`};`
	
	`output = [&counter](int n) { std::cout << n; ++counter; };`
		No limitations
		Very slow

C++ templates	`template <typename Output>` `void f(Output output);`	No limitations Extremely fast