Assignment 1

Description

Description provided by the course:

> In the first assignment of the MDSD course, you have to write an Internal DSL for state machines using a Fluent API. The assignment has two variants: "normal" and "advanced". You can pick any of these variants. The normal version is programmed using Java, and in the advanced version you can use any programming language you want.

This assignment uses the advanced version, with the programming language being PHP.

Solution

The solution builds on top of the provided source code, provided by the simple version of the assignment.
This has then been translated into PHP (mostly a one-to-one translation). This gives a baseline with three model classes: `Machine`, `State`, and `Transition`.
In order to run the state machine, a `MachineInterpreter` class is created. This handles the interpretation of the state machine, and the transitions between states. It therefore handles any conditions and operations that are defined.
In order to create the fluid API a `StateMachine` class is created. This handles the fluid API parsing to a state machine.

Tests for the code, which primarily builds on top of the provided tests, are located in the `tests` directory. These tests are run using PHPUnit.

Two variation of the fluid API is available.
First variation uses method chaining to create the state machine. Second variation uses litteral collection, or more specifically litteral map, as PHP supports this. However this implementation is a bit more complex and uses builder pattern with specific classes, which are located in `src/Builder` directory.

Both versions uses underlying the same process of building the state machine.
The building processes utilize command query access to data on the underlying model. Here, each element is added to the model, when it is specified in the fluid API.
Exceptions to this, in the target of transitions, which is connected when building the machine through the `build` method. This is due to the fact, that states may not be defined when added as targets to transitions. When setting states through the `states` method, they are added into a context, such that following transition additions can be build upon the context state. The same happens for transitions, when the `when` method is called.
It is important to highlight, that this method of gradual building the model, may lead to errors or partial builded elements. Therefore a validation check may

be added to the `build` method to ensure that the model is valid. However, this is not implemented in this solution.

Example usage

The following is example usages of the fluid API:

Method chaining (from `CDPLayerMethodChaining.php`):

```
$stateMachine = new StateMachine();
$NUMBER_TRACKS = 10;
$m = $stateMachine
    ->integer("track")
    ->state("STOP")->initial()
        ->when("PLAY")->to("PLAYING")->set("track", 1)->ifEquals("track", 0)
        ->when("PLAY")->to("PLAYING")
    ->state("PLAYING")
        ->when("STOP")->to("STOP")
        ->when("PAUSE")->to("PAUSED")
        ->when("TRACK_END")->to("STOP")->ifEquals("track", $NUMBER_TRACKS)
        ->when("TRACK_END")->to("PLAYING")->increment("track")
    ->state("PAUSED")
        ->when("STOP")->to("STOP")
        ->when("PLAY")->to("PLAYING")
        ->when("FORWARD")->to("PAUSED")
            ->increment("track")->ifLessThan("track", $NUMBER_TRACKS + 1)
        ->when("BACK")->to("PAUSED")
            ->decrement("track")->ifGreaterThan("track", 1)
    ->build();
```

Litteral map (from `CDPlayerLitteralMapTest.php`)::

```
$stateMachine = new StateMachine();
$NUMBER_TRACKS = 10;
$m = $stateMachine
    ->integer("track")
    ->state(
        name: "STOP",
        initial: true,
        transitions: [
            new Transition(
                when: "PLAY",
                to: "PLAYING",
                set: new Set("track", 1),
                ifEquals: new IfEquals("track", 0),
            ),
            new Transition(
                when: "PLAY",
                to: "PLAYING"
            )
        ],
    )
    ->state(
        name: "PLAYING",
        transitions: [
            new Transition(
                when: "STOP",
                to: "STOP"
            ),
            new Transition(
                when: "PAUSE",
                to: "PAUSED"
            ),
            new Transition(
                when: "TRACK_END",
                to: "STOP",
                ifEquals: new IfEquals("track", $NUMBER_TRACKS)
            ),
            new Transition(
                when: "TRACK_END",
                to: "PLAYING",
                Increment: new Increment("track")
            )
        ]
    )
    ->state("PAUSED", [
        new Transition("STOP", "STOP"),
        new Transition("PLAY", "PLAYING"),
        new Transition(
            "FORWARD",
            "PAUSED",
            new Increment("track"),
            new IfLessThan("track", $NUMBER_TRACKS + 1)
        ),
        new Transition(
            "BACK",
            "PAUSED",
            new Decrement("track"),
            new IfGreaterThan("track", 1)
        )
    ])
    ->build();
```

Litteral collection is build to support both litteral list and map. Litteral list is

shown in the last state ("PAUSED") in the example above.
The IDE, with the correct language support, would showcase keys for each element.
The implementation in the litteral list example, utalizie the class name, to determine which field is being filled out. This is implemented in `src/Builder/Transition.php`.

It is important to note, that the litteral collection example looks to fill more, but that is due to more new lines, compared to the method chaining example.

Source code

The source code for assignment 1 is available at github.com/The0mikkel/sdu-mbsd. A pull request is available at github.com/The0mikkel/sdu-mbsd/pull/1, where code can be commented on.

Included in the code, is a `docker-compose.yml` file, which can be used to run the tests in a docker container.

Please note, that the code utalize multiple PHP features, which is not available before PHP 8.2.

This document is available at github.com/The0mikkel/sdu-mbsd/assignment1/README.md.