Score with classic



The Score class provides you a structure for keeping track of the score.

Download the classic.lua module from https://github.com/rxi/classic

Put it next to your main.lua file.

Now, create a score.lua file, next to your main.lua file:

```
local Class = require "classic"
local Score = Class:extend()
```

```
function Score:new()
    self.value = 0
end
```

function Score:show()

```
print("Score: " .. tostring(self.score))
```

end

```
function Score:increment(value)
   self.value = self.value + 1
```

end

end

Return Score

function Score:set(value) self.value = value

Score with classic



The Score class provides you a structure for keeping track of the score.

Download the classic.lua module from https://github.com/rxi/classic

Put it next to your main. lua file.

Now, create a score.lua file, next to your main.lua file:

```
local Class = require "classic"
local Score = Class:extend()
```

```
function Score:new()
    self.value = 0
end
```

function Score:show()

```
print("Score: " .. tostring(self.score))
```

```
function Score:increment(value)
   self.value = self.value + 1
end
```

function Score:set(value) self.value = value

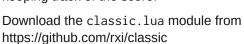
end

Return Score

Score with classic



The Score class provides you a structure for keeping track of the score.



Put it next to your main.lua file.

Now, create a score.lua file, next to your main.lua file:

```
local Class = require "classic"
local Score = Class:extend()
```

```
function Score:new()
    self.value = 0
end
```

function Score:show()

```
print("Score: " .. tostring(self.score))
```

end

function Score:increment(value) self.value = self.value + 1 end

function Score:set(value) self.value = value end

Return Score

Score with classic



The Score class provides you a structure for keeping track of the score.

Download the classic.lua module from https://github.com/rxi/classic

Put it next to your main. lua file.

Now, create a score. lua file, next to your main. lua file:

```
local Class = require "classic"
local Score = Class:extend()
```

```
function Score:new()
    self.value = 0
end
```

```
function Score:show()
    print("Score: " .. tostring(self.score))
```

```
function Score:increment(value)
   self.value = self.value + 1
end
```

```
function Score:set(value)
    self.value = value
end
```

Return Score

end

Using the Score class

score:show()

You can now use the Score class in your main.lua:

```
local Score = require "score"

playerA = Score()
playerA:show()

playerB = Score2()
playerB:show()
playerB:increment()
player:show()
```

Before using it, we have to first include the Score class from the score.lua file: most of the time you will name the class with the same name as defined in the module, which is probably also the name of the file, capitalized, without the .lua extension.. But it's not mandatory.

When you run your program you will get the output:

```
1 -- the score for player A
```

- 1 -- the initial score for player B
- 2 -- the incremented score for player B
- 1 -- the unmodified score for player A

PlayerA and PlayerB have different scores but share the implementation of the show() and increment() functions.

Using the Score class

You can now use the Score class in your main.lua:

```
local Score = require "score"

playerA = Score()
playerA:show()

playerB = Score2()
playerB:show()
playerB:increment()
player:show()

score:show()
```

Before using it, we have to first include the Score class from the score.lua file: most of the time you will name the class with the same name as defined in the module, which is probably also the name of the file, capitalized, without the .lua extension.. But it's not mandatory.

When you run your program you will get the output:

```
1 -- the score for player A
```

- 1 -- the initial score for player B
- 2 -- the incremented score for player B
- 1 -- the unmodified score for player A

PlayerA and PlayerB have different scores but share the implementation of the show() and increment() functions.

Using the Score class

You can now use the Score class in your main.lua:

```
local Score = require "score"

playerA = Score()
playerA:show()

playerB = Score2()
playerB:show()
playerB:increment()
player:show()

score:show()
```

Before using it, we have to first include the Score class from the score.lua file: most of the time you will name the class with the same name as defined in the module, which is probably also the name of the file, capitalized, without the .lua extension.. But it's not mandatory.

When you run your program you will get the output:

```
1 -- the score for player A
```

- 1 -- the initial score for player B
- 2 -- the incremented score for player B
- 1 -- the unmodified score for player A

PlayerA and PlayerB have different scores but share the implementation of the show() and increment() functions.

Using the Score class

You can now use the Score class in your main.lua:

```
local Score = require "score"

playerA = Score()
playerA:show()

playerB = Score2()
playerB:show()
playerB:increment()
player:show()

score:show()
```

Before using it, we have to first include the Score class from the score.lua file: most of the time you will name the class with the same name as defined in the module, which is probably also the name of the file, capitalized, without the .lua extension.. But it's not mandatory.

When you run your program you will get the output:

- 1 -- the score for player A
- 1 -- the initial score for player B
- 2 -- the incremented score for player B
- 1 -- the unmodified score for player A

PlayerA and PlayerB have different scores but share the implementation of the show() and increment() functions.