Classes and Objects

Classes are code templates providing a common place to store values and behaviors (functions).
Objects are variables that are created from a class.

Differently to many other languages, Lua does not have built-in classes, but you can easily define classes by putting variables and functions in lists.

Creating a point class

```
local Point = {}
function Point:create()
  local point = {}

  point.x = 0

  function point:action()
    -- code with the action
  end

  return point
end
```

Creating an object from a class

```
vertex= Point:create()
vertex:action()
```


Classes and Objects

Classes are code templates providing a common place to store values and behaviors (functions).
Objects are variables that are created from a class.

Differently to many other languages, Lua does not have built-in classes, but you can easily define classes by putting variables and functions in lists.

Creating a point class

```
local Point = {}
function Point:create()
  local point = {}

  point.x = 0

  function point:action()
    -- code with the action
  end

  return point
end
```

Creating an object from a class

```
vertex= Point:create()
vertex:action()
```

Classes and Objects

Classes are code templates providing a common place to store values and behaviors (functions).
Objects are variables that are created from a class.

Differently to many other languages, Lua does not have built-in classes, but you can easily define classes by putting variables and functions in lists.

Creating a point class

```
local Point = {}
function Point:create()
  local point = {}

  point.x = 0

  function point:action()
    -- code with the action
  end

  return point
end
```

Creating an object from a class

```
vertex= Point:create()
vertex:action()
```

Classes and Objects

Classes are code templates providing a common place to store values and behaviors (functions).
Objects are variables that are created from a class.

Differently to many other languages, Lua does not have built-in classes, but you can easily define classes by putting variables and functions in lists.

Creating a point class

```
local Point = {}
function Point:create()
  local point = {}

  point.x = 0

  function point:action()
    -- code with the action
  end

  return point
end
```

Creating an object from a class

```
vertex= Point:create()
vertex:action()
```

The anatomy of a class

- The class is "contained" in a list (called here Point, with a capital "P"; in your code you will use a specific name for your class, instead of Point)
- The list only contains a constructor called create.
- The constructor first defines a new list (point), then adds the internal variables (x) and functions (action) to the list. Finally the constructor returns the local variable. It's this variable that will become the object in the "main" code.

The object

- The object is created by calling the create() function of Point.

Class modules

Most of the time you will put the class in a separate file: create a file containing a local definition of a class and returning the local variable at the end of the file.

To be noted

- The name of the class starts with a capital letter;
- Per convention, we always call the "constructor" create().
- The name of the variable containing the class implementation starts with a lowercase letter (point).
- the function in the class are defined and called by using a semi colon and not a dot.

The anatomy of a class

- The class is "contained" in a list (called here Point, with a capital "P"; in your code you will use a specific name for your class, instead of Point)
- The list only contains a constructor called create.
- The constructor first defines a new list (point), then adds the internal variables (x) and functions (action) to the list. Finally the constructor returns the local variable. It's this variable that will become the object in the "main" code.

The object

- The object is created by calling the create() function of Point.

Class modules

Most of the time you will put the class in a separate file: create a file containing a local definition of a class and returning the local variable at the end of the file.

To be noted

- The name of the class starts with a capital letter;
- Per convention, we always call the "constructor" create().
- The name of the variable containing the class implementation starts with a lowercase letter (point).
- the function in the class are defined and called by using a semi colon and not a dot.

The anatomy of a class

- The class is "contained" in a list (called here Point, with a capital "P"; in your code you will use a specific name for your class, instead of Point)
- The list only contains a constructor called create.
- The constructor first defines a new list (point), then adds the internal variables (x) and functions (action) to the list. Finally the constructor returns the local variable. It's this variable that will become the object in the "main" code.

The object

- The object is created by calling the create() function of Point.

Class modules

Most of the time you will put the class in a separate file: create a file containing a local definition of a class and returning the local variable at the end of the file.

To be noted

- The name of the class starts with a capital letter;
- Per convention, we always call the "constructor" create().
- The name of the variable containing the class implementation starts with a lowercase letter (point).
- the function in the class are defined and called by using a semi colon and not a dot.

The anatomy of a class

- The class is "contained" in a list (called here Point, with a capital "P"; in your code you will use a specific name for your class, instead of Point)
- The list only contains a constructor called create.
- The constructor first defines a new list (point), then adds the internal variables (x) and functions (action) to the list. Finally the constructor returns the local variable. It's this variable that will become the object in the "main" code.

The object

- The object is created by calling the create() function of Point.

Class modules

Most of the time you will put the class in a separate file: create a file containing a local definition of a class and returning the local variable at the end of the file.

To be noted

- The name of the class starts with a capital letter;
- Per convention, we always call the "constructor" create().
- The name of the variable containing the class implementation starts with a lowercase letter (point).
- the function in the class are defined and called by using a semi colon and not a dot.