Object Creation

In this lesson, we will create a rectangle object and study its properties.

WE'LL COVER THE FOLLOWING

The Structure

The structure of an object is very similar to that of a record. A typical object contains three components:

Values

These are the properties of the object. They are defined in the object body using the `val` keyword. The values of an object cannot be directly accessed from the outside.

Depending on our program, the values of an object can be mutable or immutable.

Public Functions

Public functions are methods which can manipulate and use the object's values. They can be defined using the `pub` keyword.

These functions can be accessed from the outside. They allow an interface between the object and the other parts of the program between the object and the other parts of the program.

In that sense, public functions can be considered open to the public.

Private Functions

Like their public counterparts, private functions also have access to the object's values. However, they are hidden from the outside world and cannot be called directly. Instead, only public functions can call the private functions of an object.

Private functions can be defined using the `pri` keyword.

Type Definition

So, let's get started and build a `rectangle` object. A good practice is to define the object's type first, just like we do for records.

For object types, the `.` operator has to be used right after the scope of the definition begins. In the type definition, we only mention the names of the public methods and their return types.

```
type rectangle = {.
  getLength: float,
  getWidth: float,
  getColor: string,
  getArea: float
};
```

This part is pretty simple. Our rectangle type is comprised of four public methods: `getLength`, `getWidth`, `getColor`, and `getArea`.

Now, we can create an object of the rectangle type. The body of the object will contain the values, along with the public and private method definitions. For now, we'll omit private methods.

The Object

```
type rectangle = {
  .
  getColor: string,
  getLength: float,
  getWidth: float,
  getArea: float
};

let obj: rectangle = {
  /* Values */
  val l = 10.5;
  val w = 5.5;

  val c = "Blue";

  /* Function definitions */
  pub getColor = c;
  pub getWidth = w;
  pub getLength = l;
  pub getArea = this#getWidth *. this#getLength
};
```

`this` Keyword

The `this` keyword is used to refer to the object itself. Whenever an object's method is used inside its body, the `this` keyword must be used to tell the compiler that the function being called is indeed, a component of the object.

The methods of an object can be accessed using the `#` operator. This is why we used the `this#getWidth` and `this#getLength` commands to access the width and length respectively.

Outside the object's body, we can access public methods using the object's name with the `#` operator:

```
type rectangle = {
  .
  getColor: string,
  getLength: float,
  getWidth: float,
  getArea: float
};

let rect: rectangle = {
  /* Values */
  val l = 10.5;
  val w = 5.5;
  val c = "Blue";

  /* Function definitions */
  pub getColor = c;
  pub getWidth = w;
  pub getLength = l;
  pub getArea = this#getWidth *. this#getLength
};

Js.log(rect#getColor);
Js.log(rect#getArea);
```

From the implementation above, we can see one big difference between objects and records.

> In an object's type definition, we do not need to define all of its properties. A record's structure must be defined completely in its type definition.

In the next lesson, we'll learn about the use of private methods in an object.