

Part 09 - Methods

Part 09 - Methods

**Definition: Method**

A function exclusively associated with a `class`

Defining a Method

Methods must be defined in `classes`. They are declared just like functions are.

arguments example

```
class Cat:
    def Roar():
        print "Meow!"

cat = Cat()
cat.Roar()
```

Output

Meow!

An object of `Cat` must be instanced, then its methods can be called.

**Recommendation**

Names of methods should always be verbs.
They should also be declared in `PascalCase`.

Class Constructor and Destructor

Constructors and Destructors are special methods that are called on when a `class` is being instanced or destroyed, respectively.

Both are optional.

arguments example

```
class Cat:
    def constructor():
        _name = 'Whiskers'

    def destructor():
        print "$_name is no more... RIP"

    [Getter(Name)]
    _name as string

cat = Cat()
print cat.Name
```

Output

```
Whiskers
Whiskers is no more... RIP
```

If a constructor has arguments, then they must be supplied when instanting. Destructors cannot have arguments.

arguments example

```
class Cat:
    def constructor(name as string):
        _name = name

    [Getter(Name)]
    _name as string

cat = Cat("Buttons")
print cat.Name
```

Output

```
Buttons
```



Be Careful

Do not depend on the destructor to always be called.

Method Modifiers

Modifier	Description
abstract	an abstract method has no implementation, which requires that an inheriting class implements it.

<code>static</code>	a <code>static</code> method is common to the entire <code>class</code> , which means that it can be called without ownership of a single instance of the <code>class</code>
<code>virtual</code>	See Part 10 - Polymorphism, or Inherited Methods
<code>override</code>	See Part 10 - Polymorphism, or Inherited Methods

All these modifiers also apply to properties (If they are explicitly declared).
`static` can also apply to fields.

static example

```
class Animal:
    def constructor():
        _currentId += 1
        _id = _currentId

    [Getter(Id)]
    _id as int

    static _currentId = 0
```

This will cause the `Id` to increase whenever an `Animal` is instanced, giving each `Animal` their own, unique `Id`.

All the methods defined in an interface are automatically declared abstract.
Abstract methods in a class must have a blank code block in its declaration.

abstract example

```
class Feline:
    abstract def Eat():
        pass

interface IFeline:
    def Eat()
```

Both declare roughly the same thing.

Member Visibility

Visibility Level	Description
<code>public</code>	Member is fully accessible to all types.
<code>protected</code>	Member is only visible to this class and inheriting classes.
<code>private</code>	Member is only visible to this class.



Important Information

All fields are by default `protected`. All methods, properties, and events are by default `public`.



Recommendation

Fields are typically either `protected` or `private`. Usually instead of making a public field, you might make a public property that wraps access to the field instead. This allows subclasses to possibly override behavior.
Methods can have any visibility.

Properties can have any visibility, and typically have both a getter and a setter, or only a getter. Instead of a set only property, consider using a method instead (like "SetSomeValue(val as int)").



Recommendation

It is recommended you prefix field names with an underscore if it is a private field.

Declaring Properties in the Constructor

One very nice feature that boo offers is being able to declare the values of properties while they are being instanced.

abstract example

```
class Box:
    def constructor():
        pass

    [Property(Value)]
    _value as object

box = Box(Value: 42)
print box.Value
```

Output

42

The constructor didn't take any arguments, yet the `Value: 42` bit declared `Value` to be 42, all in a tightly compact, but highly readable space.

Exercises

1. Create two classes, `Predator` and `Prey`. To the `Predator` class, add an `Eat` method that eats the `Prey`. Do not let the `Prey` be eaten twice.

Go on to [Part 10 - Polymorphism, or Inherited Methods](#)