```
Object subclass: #Settings
 instanceVariableNames: 'content'
 classVariableNames: ''
 poolDictionaries: ''
 category: 'PLP'!

!Settings methodsFor: 'as yet unclassified' stamp: 'JuanEdi 6/16/2015 10:06'!
doesNotUnderstand: aMessage

 (aMessage numArgs = 0 and: [content includesKey: aMessage selector]) ifTrue: [
  ^ content at: aMessage selector
  ].

 (aMessage numArgs = 1) ifTrue: [ |getter|
  getter := aMessage selector truncateTo: (aMessage selector size - 1).
  content at: getter put: (aMessage argument).
  ^ self.
   ].

 ^ super doesNotUnderstand: aMessage.! !

!Settings methodsFor: 'as yet unclassified' stamp: 'JuanEdi 6/16/2015 09:51'!
withContent: aDictionary

 content := aDictionary.! !

"-- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- "!

Settings class
 instanceVariableNames: ''!

!Settings class methodsFor: 'as yet unclassified' stamp: 'JuanEdi 6/16/2015 09:51'!
withContent: aDictionary

 ^ self new withContent: aDictionary! !


TestCase subclass: #SettingsTest
 instanceVariableNames: ''
 classVariableNames: ''
 poolDictionaries: ''
 category: 'PLP'!

!SettingsTest methodsFor: 'as yet unclassified' stamp: 'JuanEdi 6/16/2015 10:20'!
testGetProperties
 |c s|
 c := Dictionary newFromPairs: #(foo bar).
 s := Settings withContent: c.

 self assert: s foo equals: #bar.
! !
```

```
!SettingsTest methodsFor: 'as yet unclassified' stamp: 'JuanEdi 6/16/2015 10:31'!
testSetProperties
 |c s|
 c := Dictionary newFromPairs: #(foo bar).
 s := Settings withContent: c.

 s baz: 123.
 self assert: s baz equals: 123.
! !


Object subclass: #CodeGenerator
 instanceVariableNames: ''
 classVariableNames: ''
 poolDictionaries: ''
 category: 'PLP'!

!CodeGenerator methodsFor: 'as yet unclassified' stamp: 'JuanEdi 6/16/2015 11:16'!
doesNotUnderstand: aMessage

 |variableName getterTemplate setterTemplate|

 getterTemplate := '{1}
      ^ {1}.'.

 setterTemplate := '{1}: value
      {1} := value.'.

 (aMessage numArgs = 1) ifTrue: [
  variableName := (aMessage selector truncateTo: (aMessage selector size - 1)).
  self class addInstVarNamed: variableName.
  self class compile: (getterTemplate format: {variableName}).
  self class compile: (setterTemplate format: {variableName}).

  ^ aMessage sendTo: self.
  ].

 ^ super doesNotUnderstand: aMessage.! !

"-- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- "!

CodeGenerator class
 instanceVariableNames: ''!

!CodeGenerator class methodsFor: 'as yet unclassified' stamp: 'JuanEdi 6/16/2015 10:45'!
named: className

 |c|
 c := self subclass: className asSymbol
 instanceVariableNames: ''
 classVariableNames: ''
 category: 'PLP'.
 ^ c new.! !


TestCase subclass: #CodeGeneratorTest
 instanceVariableNames: ''
 classVariableNames: ''
 poolDictionaries: ''
 category: 'PLP'!

!CodeGeneratorTest methodsFor: 'as yet unclassified' stamp: 'JuanEdi 6/16/2015 10:40'!
clearSubclasses

 CodeGenerator subclasses do: [ :aSubclass | aSubclass removeFromSystem ].! !

!CodeGeneratorTest methodsFor: 'as yet unclassified' stamp: 'JuanEdi 6/16/2015 10:41'!
tearDown

 self clearSubclasses.! !

!CodeGeneratorTest methodsFor: 'as yet unclassified' stamp: 'JuanEdi 6/16/2015 11:12'!
test02AddsInstanceVariableWhenReceivingUnknownUnaryMessage

 | aCircle |

 aCircle := CodeGenerator named: #Circle.
 aCircle radius: 10.

 self assert: (aCircle class instanceVariables includes: #radius).! !

!CodeGeneratorTest methodsFor: 'as yet unclassified' stamp: 'JuanEdi 6/16/2015 11:12'!
test01GeneratesNewClass

 CodeGenerator named: #Circle.
 self assert: CodeGenerator subclasses size equals: 1.! !

!CodeGeneratorTest methodsFor: 'as yet unclassified' stamp: 'JuanEdi 6/16/2015 11:14'!
test03AddsGetterAndSetterWhenReceivingUnknownUnaryMessage

 | aCircle |

 aCircle := CodeGenerator named: #Circle.
 aCircle radius: 10.

 self assert: (aCircle respondsTo: #radius:).
 self assert: (aCircle respondsTo: #radius).

 self assert: aCircle radius equals: 10.

 aCircle radius: 20.
 self assert: aCircle radius equals: 20.! !
```