```
Object subclass: #FormulaDisplayer
	instanceVariableNames: ''
	classVariableNames: ''
	poolDictionaries: ''
	category: 'PLP Solución'!

FormulaDisplayer class
	instanceVariableNames: ''!

printFormula: aFormula
	"Returns the string of a formula, with parenthesis if needed"

	^ (aFormula isKindOf: BinaryFormula) ifTrue: [ '( ' , (aFormula asString) , ' )' ] ifFalse: [ aFormula
asString ].! !

Object subclass: #PropositionalFormula
	instanceVariableNames: ''
	classVariableNames: ''
	poolDictionaries: ''
	category: 'PLP Solución'!

==> aFormula
	"Creates a new Implication formula"

	^ Implication of: self and: aFormula.
	! !

not
	"Creates a new Negation formula"

	^ Negation of: self .
	! !

| aFormula
	"Creates a new Disjunction formula"

	^ Disjunction of: self and: aFormula.
	! !

hash
	"Returns the hash of the formula"

	^ self asString hash.! !

= aFormula
	"Compares 2 formulas"

	^ (self asString) = (aFormula asString)! !

printString
	"Prints the formula as a string"
	^ self asString.! !

& aFormula
	"Creates a new Conjunction formula"

	^ Conjunction of: self and: aFormula.
	! !

PropositionalFormula subclass: #BinaryFormula
	instanceVariableNames: 'form1 form2'
	classVariableNames: ''
	poolDictionaries: ''
	category: 'PLP Solución'!

allPropVars
	"Return the name of all the vars in the formula"

	| vars1 vars2 |

	vars1 := form1 allPropVars.
	vars2 := form2 allPropVars.

	vars2 do: [:each | vars1 add: each].
	^ vars1.

	! !

setForm1: aFormula1 setForm2: aFormula2

	form1 := aFormula1.
	form2 := aFormula2.
	^ self.! !

value: aSet
	"Return the evaluation of the formula"

	| val1 val2 |

	val1 := form1 value: aSet.
	val2 := form2 value: aSet.

	^ (Message selector: (self operator) argument: val2) sendTo: val1.! !

asString
	"Prints the formula as a string"

	| theOperator string1 string2 |

	theOperator := self operator.
	string1 := FormulaDisplayer printFormula: form1.
	string2 := FormulaDisplayer printFormula: form2.

	^ string1, ' ', theOperator, ' ', string2.

! !

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

BinaryFormula class
	instanceVariableNames: ''!

of: f1 and: f2.
	"Create a new binary formula with the given formulas"

	^ self new setForm1: f1 setForm2: f2.! !

BinaryFormula subclass: #Conjunction
	instanceVariableNames: ''
	classVariableNames: ''
	poolDictionaries: ''
	category: 'PLP Solución'!

operator
	"Returns the conjunction operator"

	^ #&.! !

negate
	"Negates the formula"

	^ Disjunction of: (form1 negate) and: (form2 negate)! !

toNNF
	"Transforms the formula to its NNF version"

	^ Conjunction of: (form1 toNNF) and: (form2 toNNF).! !

BinaryFormula subclass: #Disjunction
	instanceVariableNames: ''
	classVariableNames: ''
	poolDictionaries: ''
	category: 'PLP Solución'!

operator
	"Returns the disjunction operator"

	^ #|.! !

negate
	"Negates the formula"
	^ Conjunction of: (form1 negate) and: (form2 negate).! !

toNNF
	"Transforms the formula to its NNF version"

	^ Disjunction of: (form1 toNNF) and: (form2 toNNF).! !

BinaryFormula subclass: #Implication
	instanceVariableNames: ''
	classVariableNames: ''
	poolDictionaries: ''
	category: 'PLP Solución'!

operator
	"Returns the implication operator"

	^ #==>.! !

negate
	"Negates the formula"

	^ Conjunction of: form1 and: (form2 negate).! !

toNNF
	"Transforms the formula to its NNF version"

	^ Disjunction of: (form1 negate toNNF) and: (form2 toNNF).! !

PropositionalFormula subclass: #PropositionalVariable
	instanceVariableNames: 'name'
	classVariableNames: ''
	poolDictionaries: ''
	category: 'PLP Solución'!

value: aSet
	"Returns true iif the name of the var is contained in the set"

	^ aSet includes: name.! !

toNNF
	"Transforms the formula to its NNF version"

	^ self.! !

setName: aString
	"Used to set the name of the variable"

	name := aString.
	^ self.! !
```

```
negate
	"Negates the formula"

	^ Negation of: self.! !

allPropVars
	"Returns the name of the var"

	^ Set newFrom: {name}.! !

asString
	"Prints the formula as a string"

	^ name.

! !

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

PropositionalVariable class
	instanceVariableNames: ''!

named: aString
	"Creates a Propositional Variable with a name"

	^ self new setName: aString! !

PropositionalFormula subclass: #UnaryFormula
	instanceVariableNames: 'form'
	classVariableNames: ''
	poolDictionaries: ''
	category: 'PLP Solución'!

setFormula: aFormula
	"Sets the formula for the negation"

	form := aFormula.
	^ self.! !

allPropVars
	"Returns the name of all the vars in the formula"

	^ form allPropVars.

	! !

asString
	"Prints the formula as a string"

	^ '¬', (FormulaDisplayer printFormula: form).
! !
```

```
"-- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- "!

UnaryFormula class
	instanceVariableNames: ''!

of: aFormula
	"Given a formula, creates a new one negating it"

	^ self new setFormula: aFormula.! !

UnaryFormula subclass: #Negation
	instanceVariableNames: ''
	classVariableNames: ''
	poolDictionaries: ''
	category: 'PLP Solución'!

operator
	"Returns the conjuntion operator"

	^ Message selector: #not.! !

value: aSet
	"Returns the evaluation of the formula"

	^ (form value: aSet) not.! !

negate
	"Negates the formula"

	^ form.! !

toNNF
	"Negates the formula"

	^ form toNNF negate.! !
```