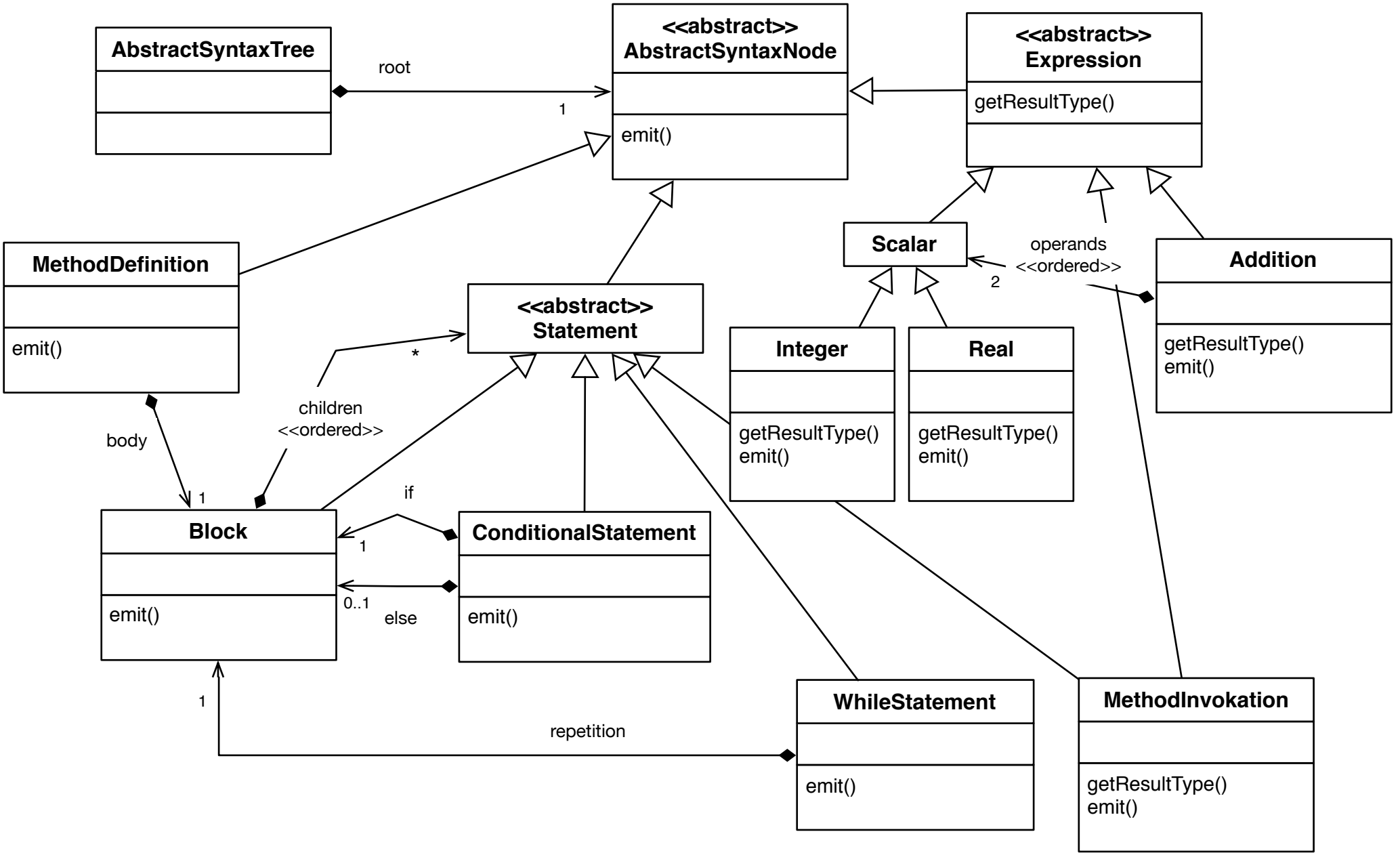
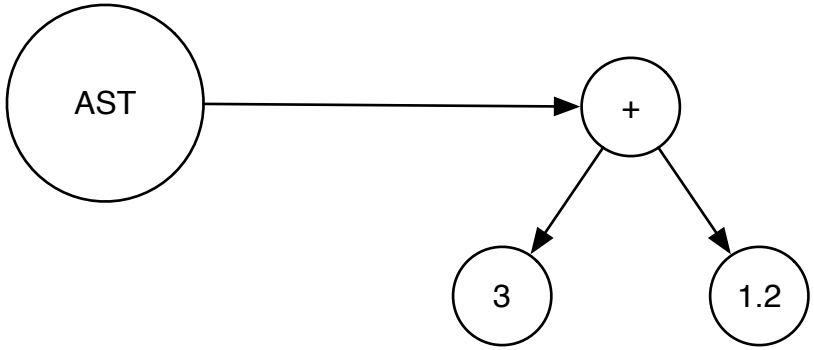
Pretty Print an Abstract Syntax Tree

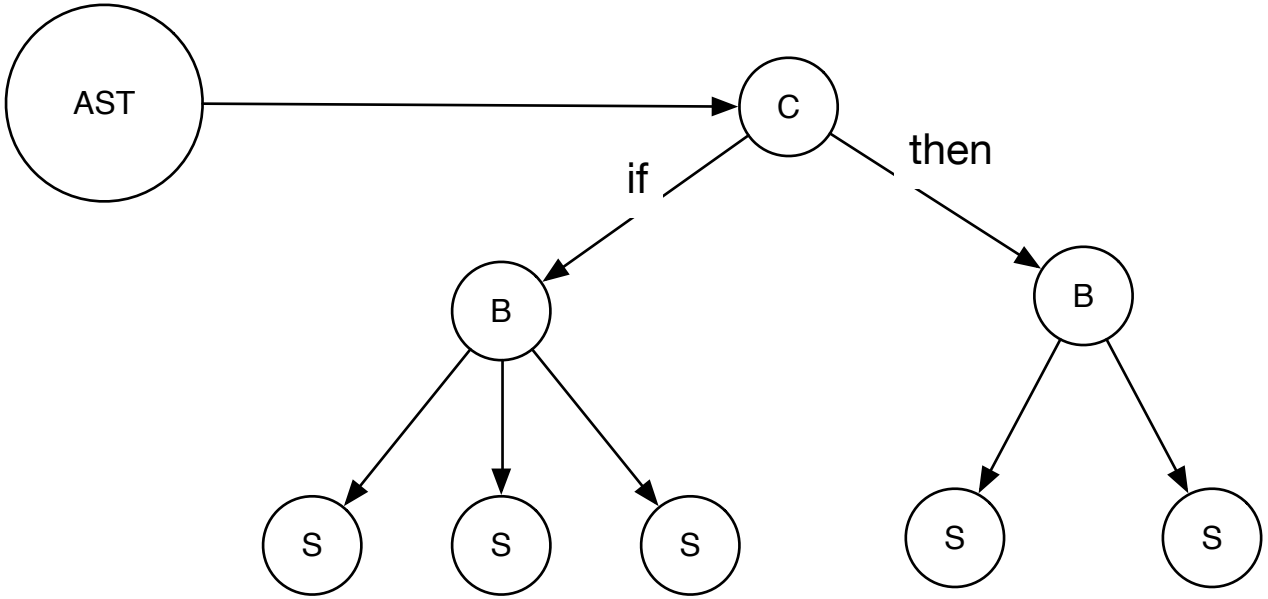


Pretty Print an Abstract Syntax Tree



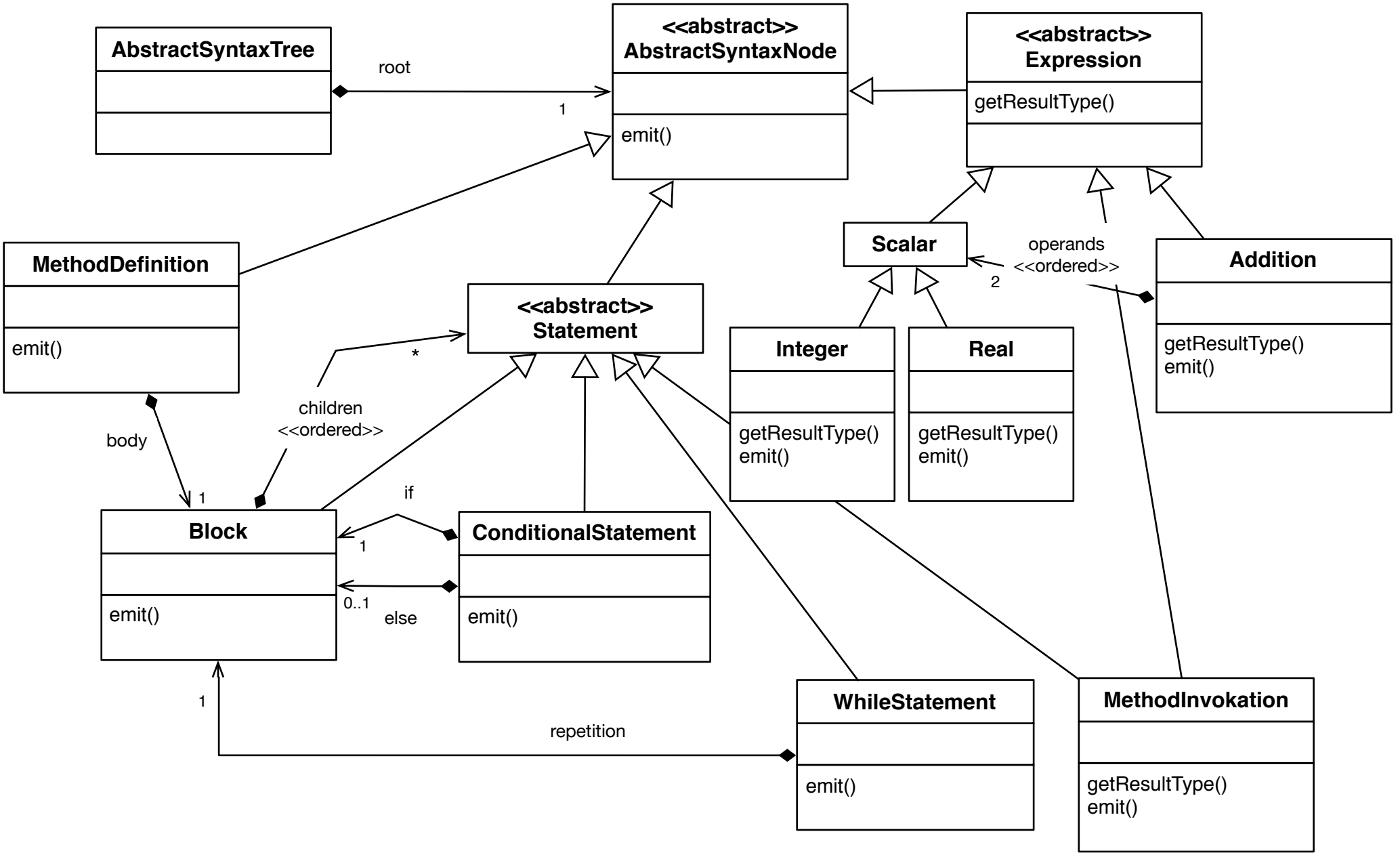
3 + 1.2

Abstract Syntax Tree Examples

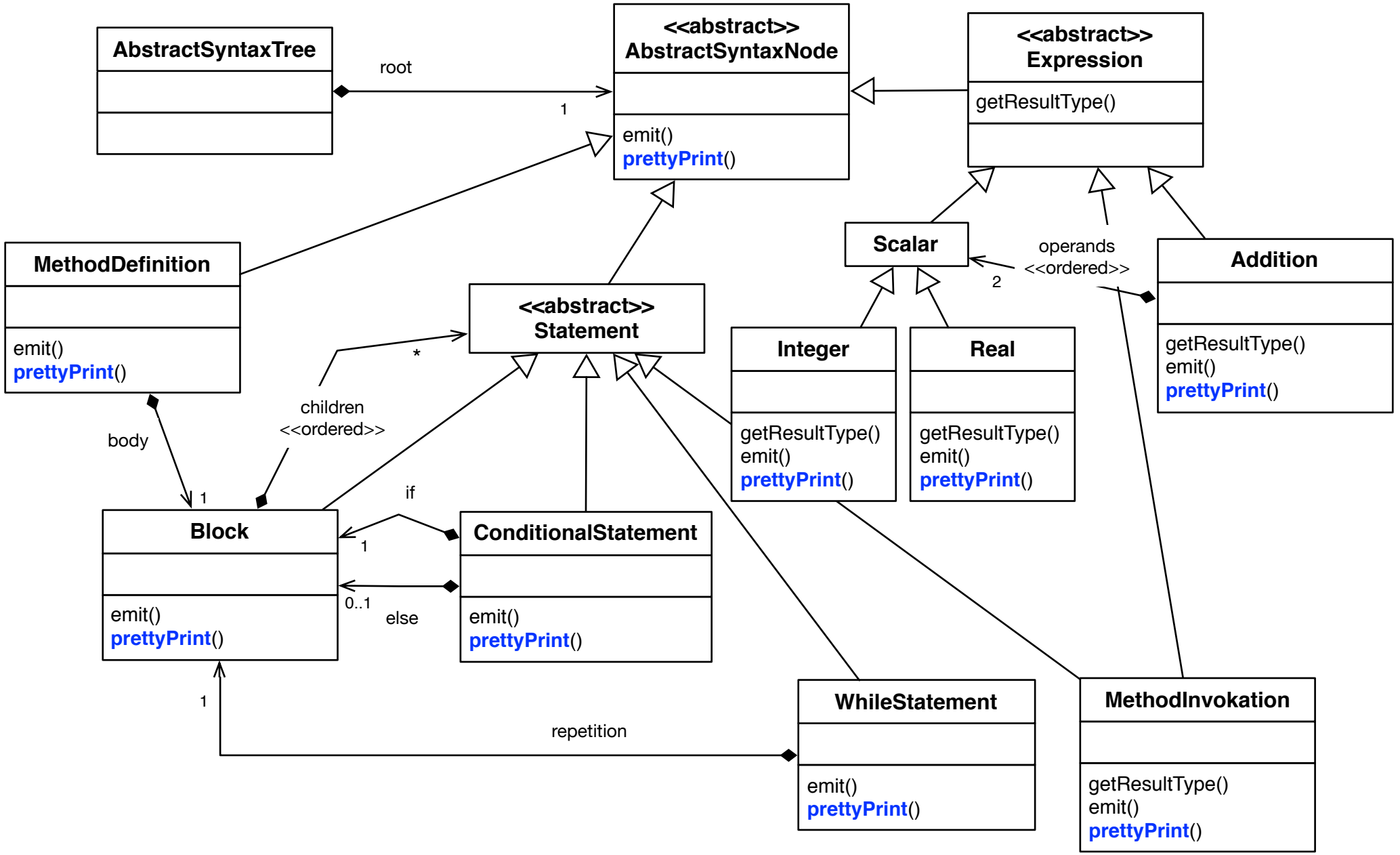



```
if () then
    5.6 + 1.2
    method()
    while()
else
    method()
    while()
end
```

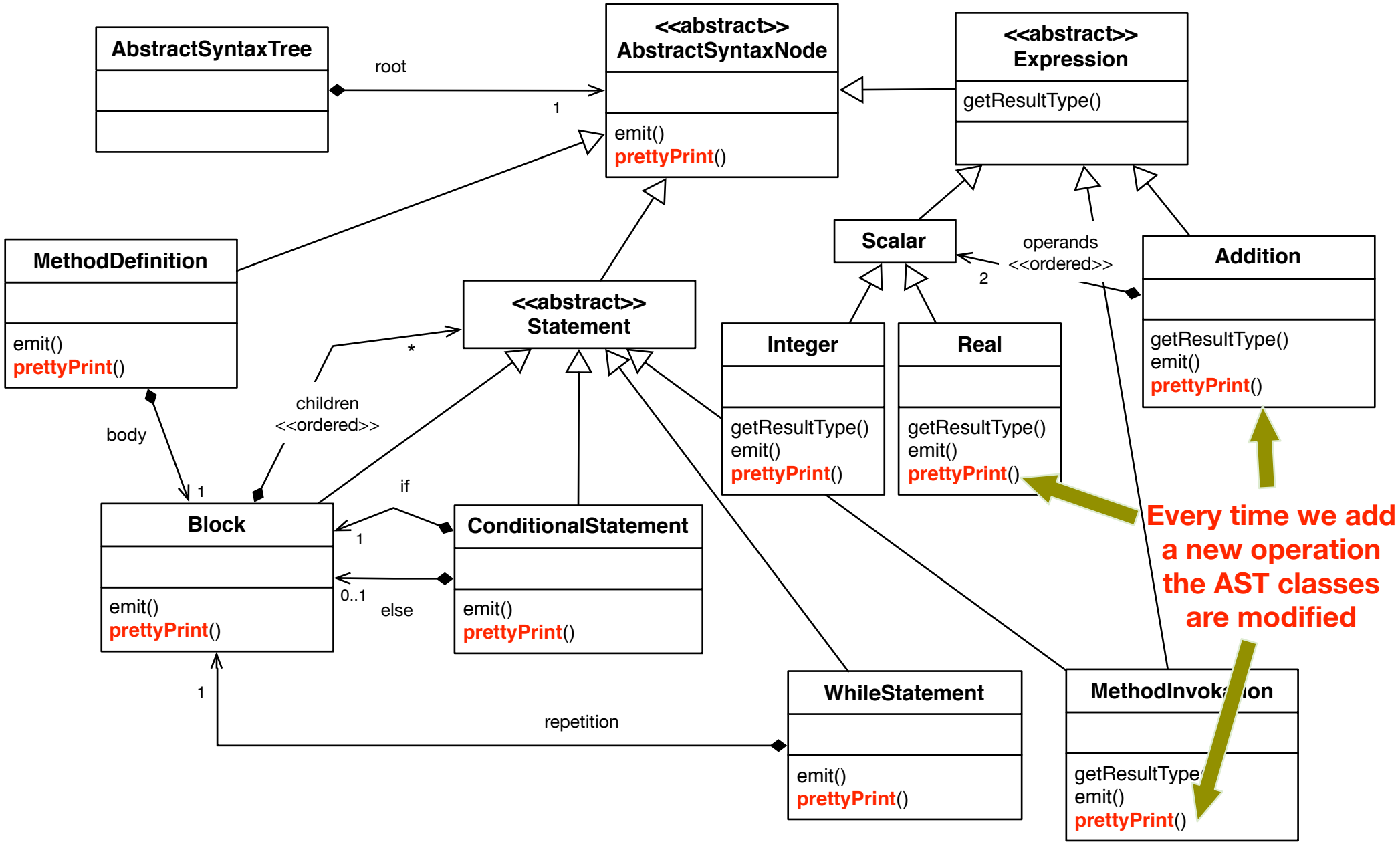
Pretty Print an Abstract Syntax Tree



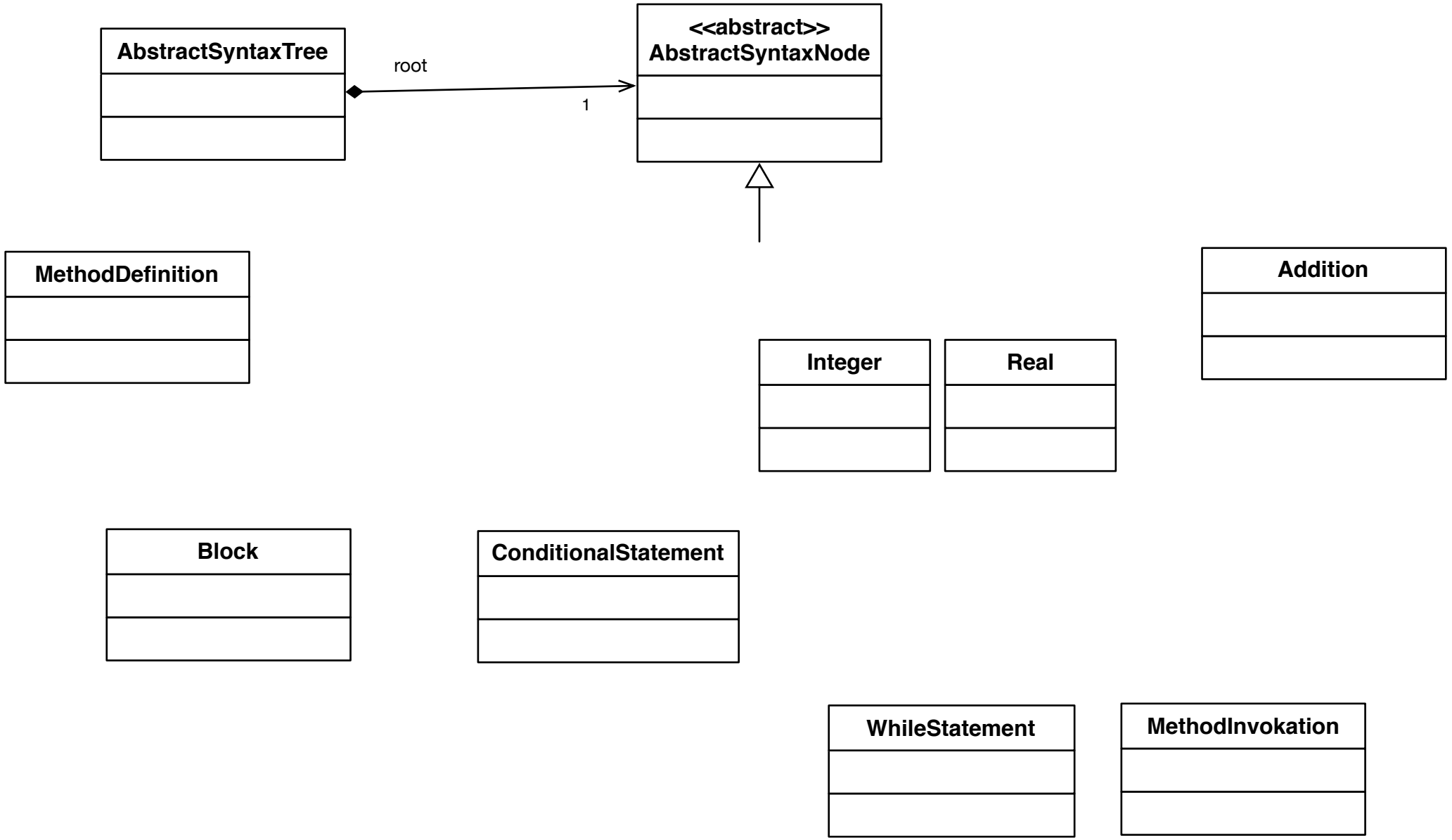
Pretty Print an Abstract Syntax Tree



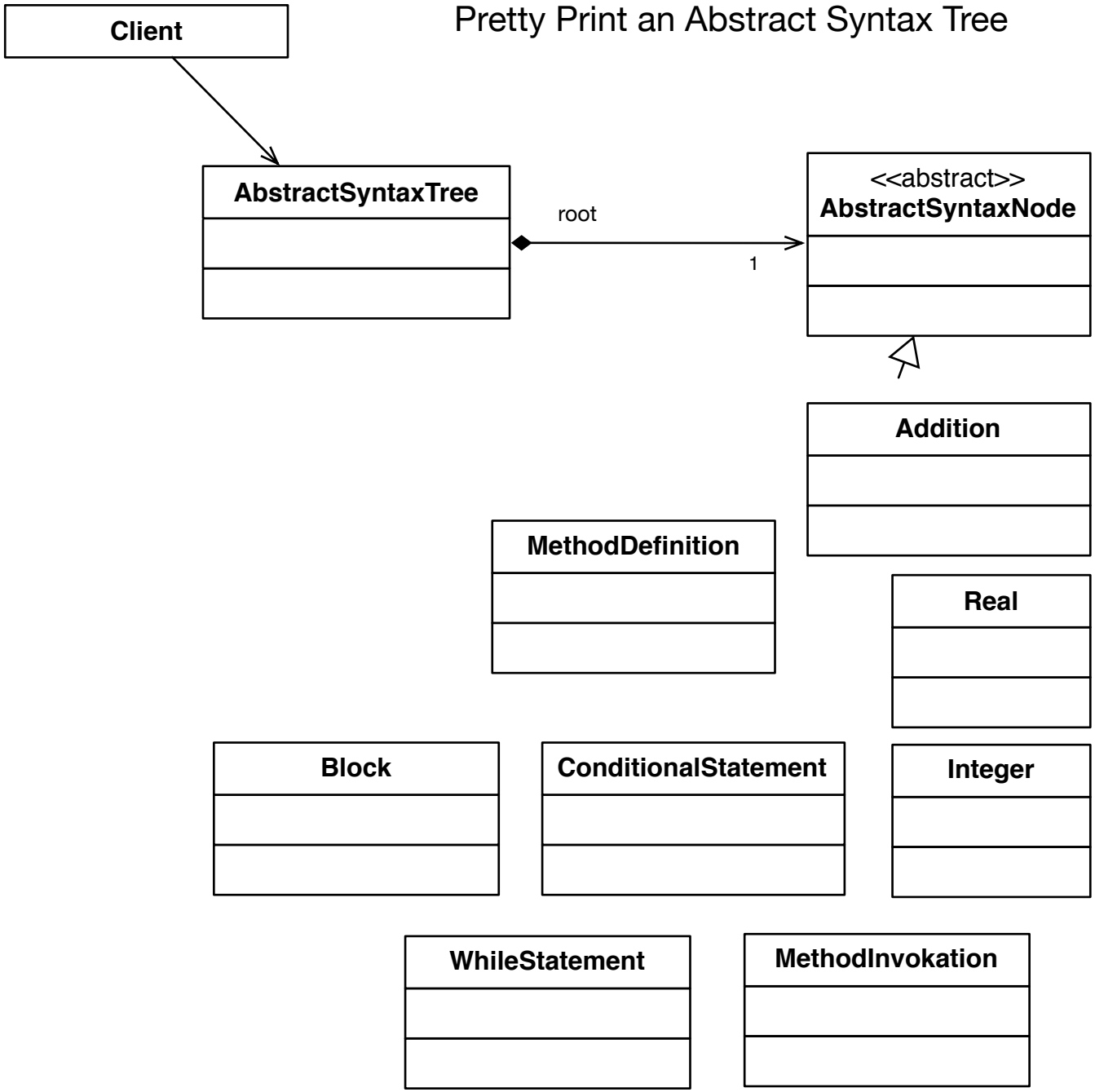
Pretty Print an Abstract Syntax Tree

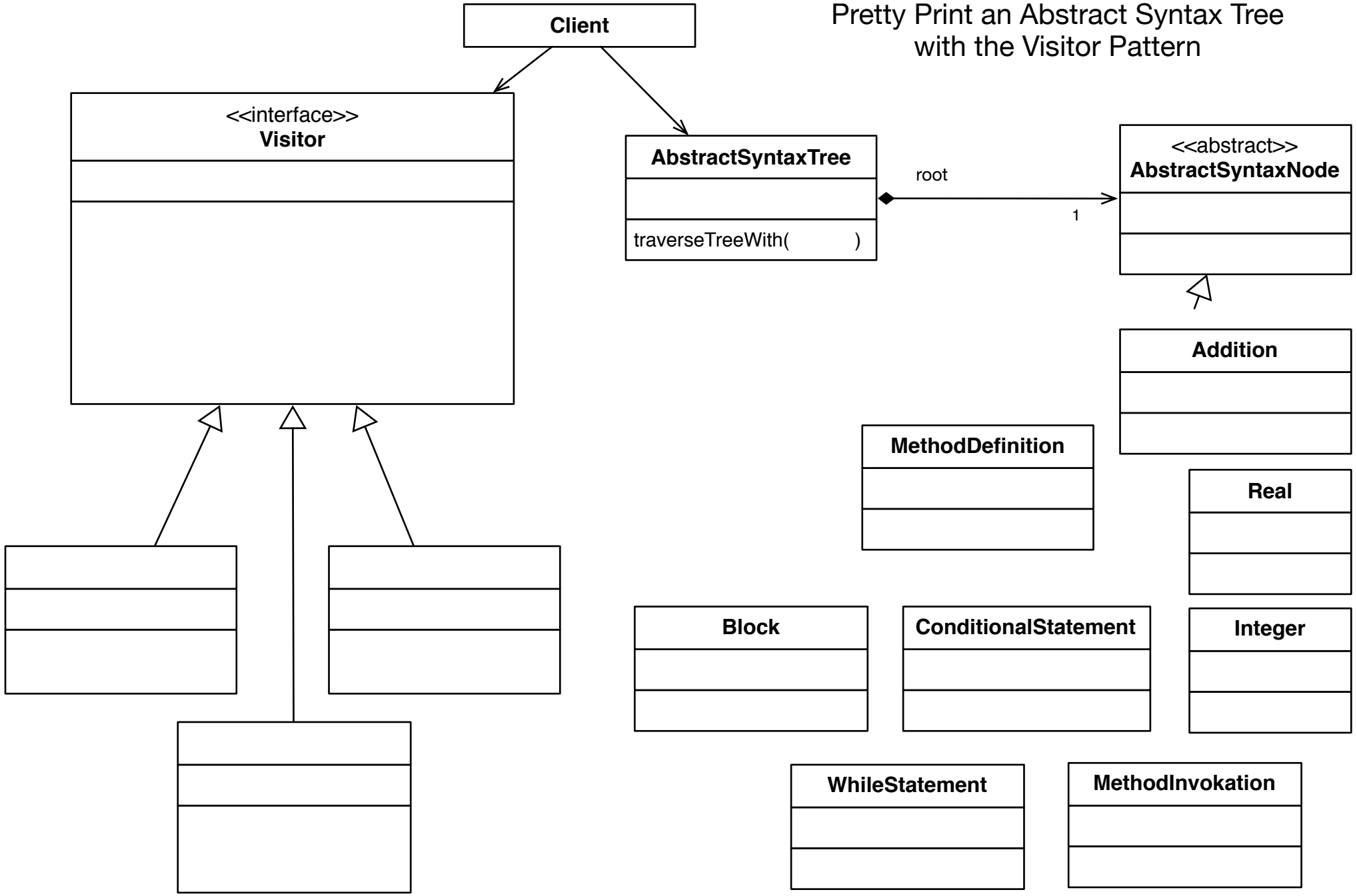


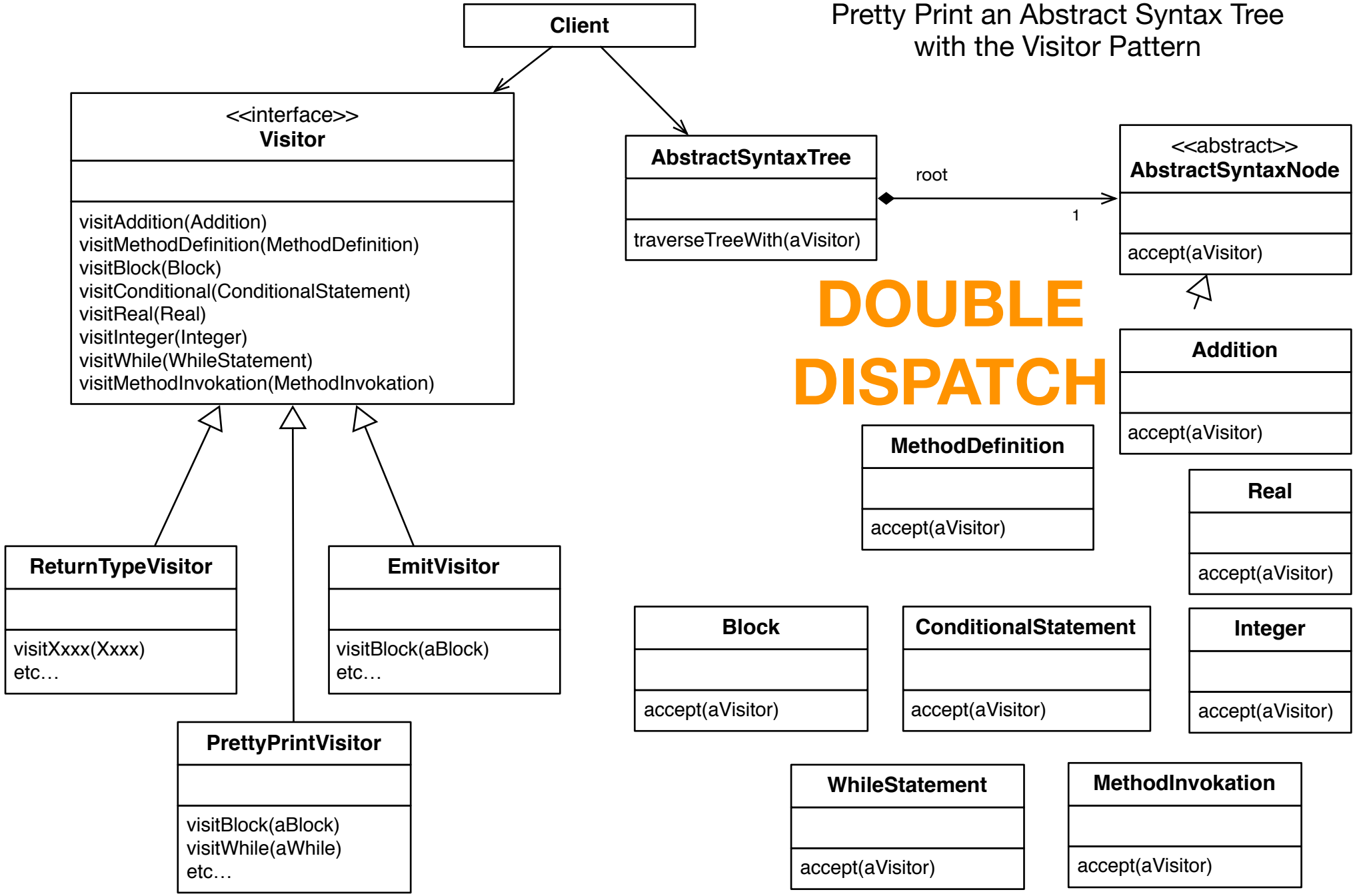
Pretty Print an Abstract Syntax Tree

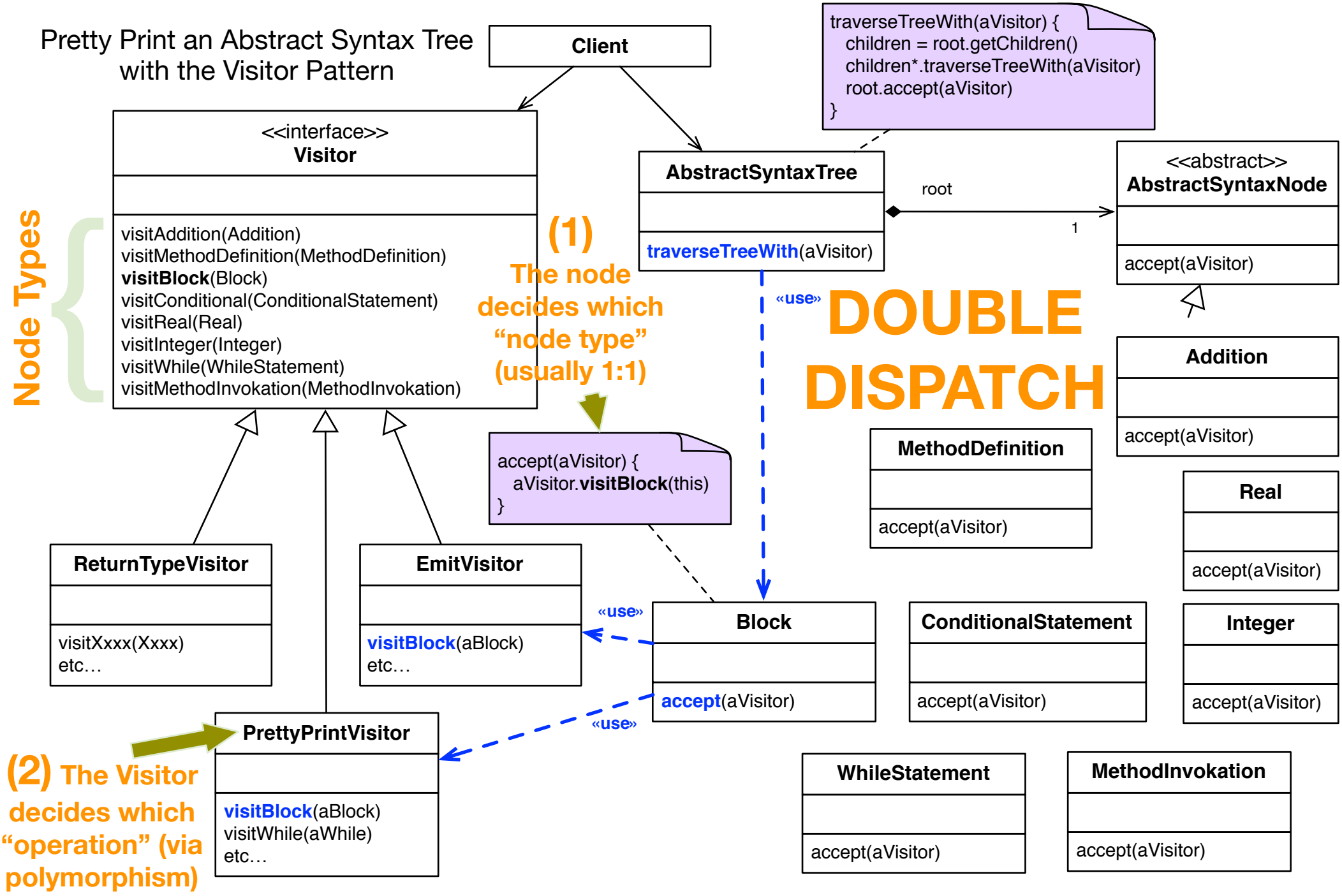


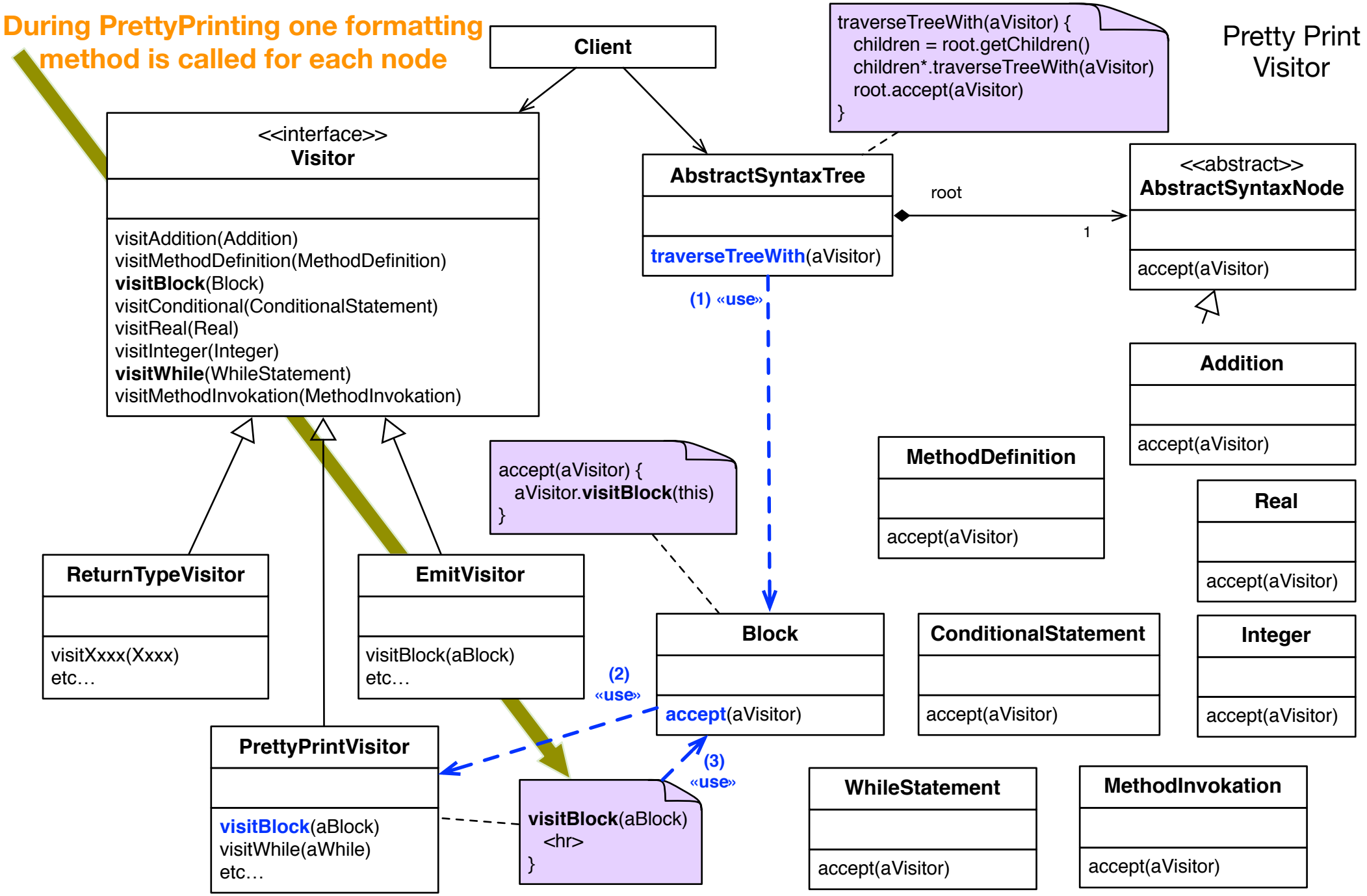
Pretty Print an Abstract Syntax Tree

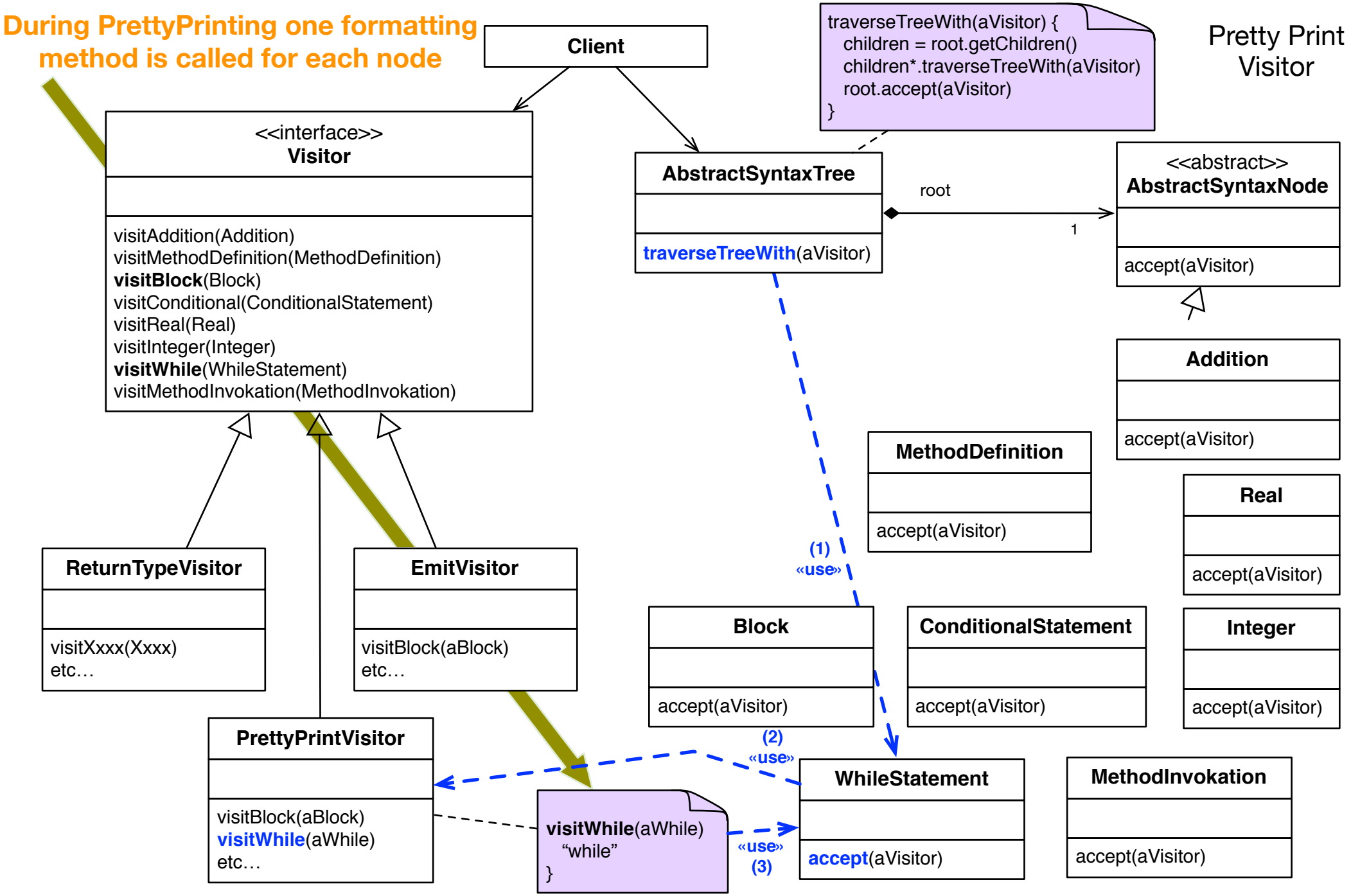


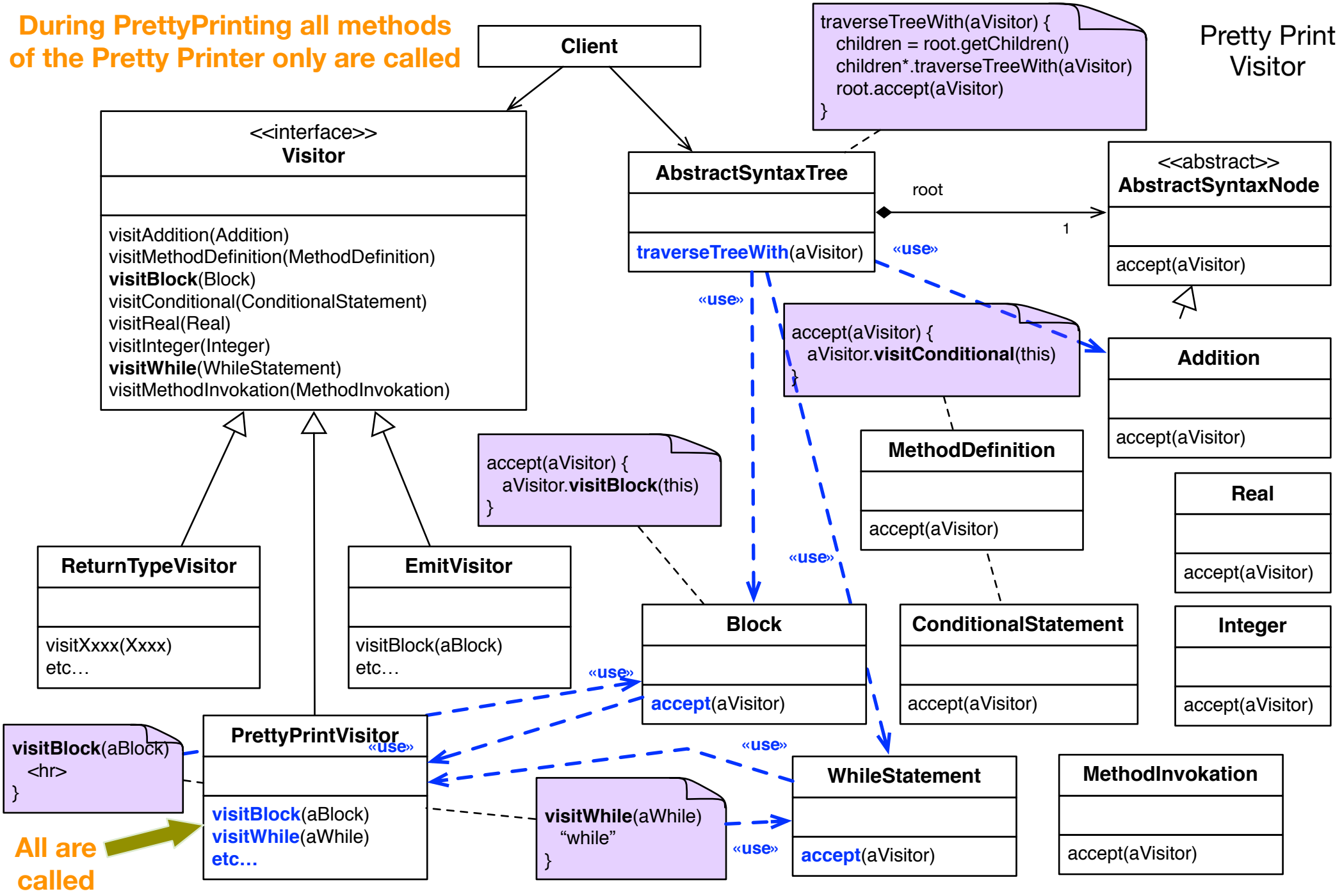

Pretty Print an Abstract Syntax Tree
with the Visitor Pattern
Client
<<interface>>
Visitor
AbstractSyntaxTree
traverseTreeWith( )
root
1
<<abstract>>
AbstractSyntaxNode
Addition
MethodDefinition
Real
Block
ConditionalStatement
Integer
WhileStatement
MethodInvokation

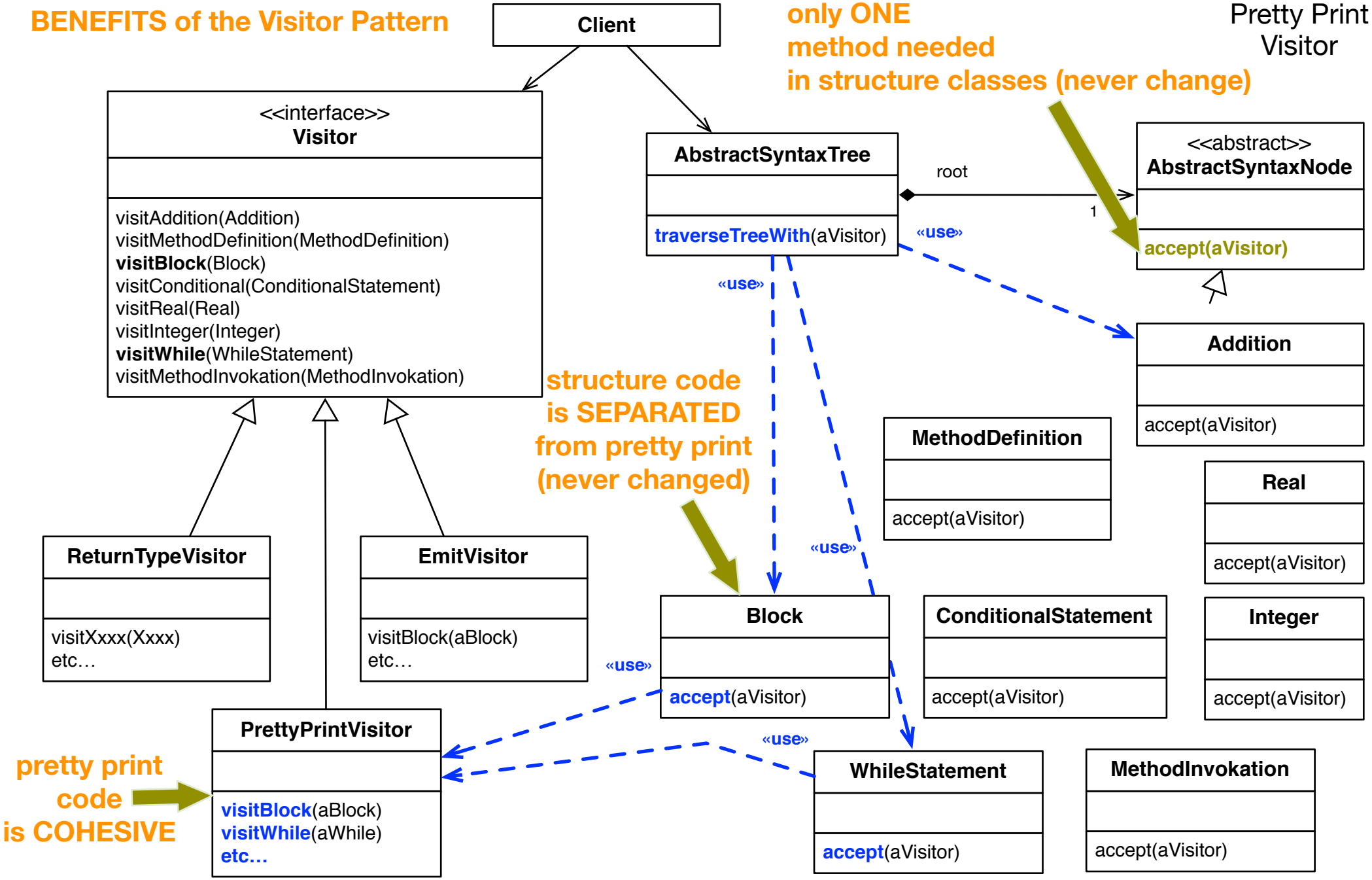

Pretty Print an Abstract Syntax Tree
with the Visitor Pattern
Client
<<interface>>
Visitor
visitAddition(Addition)
visitMethodDefinition(MethodDefinition)
visitBlock(Block)
visitConditional(ConditionalStatement)
visitReal(Real)
visitInteger(Integer)
visitWhile(WhileStatement)
visitMethodInvokation(MethodInvokation)
AbstractSyntaxTree
traverseTreeWith(aVisitor)
root
1
<<abstract>>
AbstractSyntaxNode
accept(aVisitor)
DOUBLE
DISPATCH
Addition
accept(aVisitor)
MethodDefinition
accept(aVisitor)
Real
accept(aVisitor)
ReturnTypeVisitor
visitXxxx(Xxxx)
etc…
EmitVisitor
visitBlock(aBlock)
etc…
Block
accept(aVisitor)
ConditionalStatement
accept(aVisitor)
Integer
accept(aVisitor)
PrettyPrintVisitor
visitBlock(aBlock)
visitWhile(aWhile)
etc…
WhileStatement
accept(aVisitor)
MethodInvokation
accept(aVisitor)


Pretty Print an Abstract Syntax Tree with the Visitor Pattern
Client
traverseTreeWith(aVisitor) {
children = root.getChildren()
children*.traverseTreeWith(aVisitor)
root.accept(aVisitor)
}
<<interface>>
Visitor
visitAddition(Addition)
visitMethodDefinition(MethodDefinition)
visitBlock(Block)
visitConditional(ConditionalStatement)
visitReal(Real)
visitInteger(Integer)
visitWhile(WhileStatement)
visitMethodInvokation(MethodInvokation)
Node Types
AbstractSyntaxTree
traverseTreeWith(aVisitor)
root
1
<<abstract>>
AbstractSyntaxNode
accept(aVisitor)
(1)
The node decides which "node type" (usually 1:1)
«use»
DOUBLE DISPATCH
Addition
accept(aVisitor)
accept(aVisitor) {
aVisitor.visitBlock(this)
}
MethodDefinition
accept(aVisitor)
Real
accept(aVisitor)
ReturnTypeVisitor
visitXxxx(Xxxx)
etc…
EmitVisitor
visitBlock(aBlock)
etc…
«use»
Block
accept(aVisitor)
ConditionalStatement
accept(aVisitor)
Integer
accept(aVisitor)
PrettyPrintVisitor
visitBlock(aBlock)
visitWhile(aWhile)
etc…
«use»
(2) The Visitor decides which "operation" (via polymorphism)
WhileStatement
accept(aVisitor)
MethodInvokation
accept(aVisitor)


During PrettyPrinting one formatting method is called for each node
Pretty Print Visitor
Client
traverseTreeWith(aVisitor) {
children = root.getChildren()
children*.traverseTreeWith(aVisitor)
root.accept(aVisitor)
}
<<interface>>
Visitor
visitAddition(Addition)
visitMethodDefinition(MethodDefinition)
visitBlock(Block)
visitConditional(ConditionalStatement)
visitReal(Real)
visitInteger(Integer)
visitWhile(WhileStatement)
visitMethodInvokation(MethodInvokation)
AbstractSyntaxTree
traverseTreeWith(aVisitor)
root
1
<<abstract>>
AbstractSyntaxNode
accept(aVisitor)
(1) «use»
Addition
accept(aVisitor)
accept(aVisitor) {
aVisitor.visitBlock(this)
}
MethodDefinition
accept(aVisitor)
Real
accept(aVisitor)
ReturnTypeVisitor
visitXxxx(Xxxx)
etc…
EmitVisitor
visitBlock(aBlock)
etc…
Block
accept(aVisitor)
ConditionalStatement
accept(aVisitor)
Integer
accept(aVisitor)
(2)
«use»
PrettyPrintVisitor
visitBlock(aBlock)
visitWhile(aWhile)
etc…
(3)
«use»
visitBlock(aBlock)
<hr>
}
WhileStatement
accept(aVisitor)
MethodInvokation
accept(aVisitor)


During PrettyPrinting one formatting method is called for each node
Client
traverseTreeWith(aVisitor) {
children = root.getChildren()
children*.traverseTreeWith(aVisitor)
root.accept(aVisitor)
}
Pretty Print Visitor
<<interface>>
Visitor
visitAddition(Addition)
visitMethodDefinition(MethodDefinition)
visitBlock(Block)
visitConditional(ConditionalStatement)
visitReal(Real)
visitInteger(Integer)
visitWhile(WhileStatement)
visitMethodInvokation(MethodInvokation)
AbstractSyntaxTree
traverseTreeWith(aVisitor)
root
1
<<abstract>>
AbstractSyntaxNode
accept(aVisitor)
Addition
accept(aVisitor)
MethodDefinition
accept(aVisitor)
Real
accept(aVisitor)
ReturnTypeVisitor
visitXxxx(Xxxx)
etc…
EmitVisitor
visitBlock(aBlock)
etc…
(1)
«use»
Block
accept(aVisitor)
ConditionalStatement
accept(aVisitor)
Integer
accept(aVisitor)
PrettyPrintVisitor
visitBlock(aBlock)
visitWhile(aWhile)
etc…
(2)
«use»
visitWhile(aWhile)
"while"
}
«use»
(3)
WhileStatement
accept(aVisitor)
MethodInvokation
accept(aVisitor)


During PrettyPrinting all methods of the Pretty Printer only are called
Client
traverseTreeWith(aVisitor) {
children = root.getChildren()
children*.traverseTreeWith(aVisitor)
root.accept(aVisitor)
}
Pretty Print Visitor
<<interface>>
Visitor
visitAddition(Addition)
visitMethodDefinition(MethodDefinition)
visitBlock(Block)
visitConditional(ConditionalStatement)
visitReal(Real)
visitInteger(Integer)
visitWhile(WhileStatement)
visitMethodInvokation(MethodInvokation)
AbstractSyntaxTree
traverseTreeWith(aVisitor)
root
1
<<abstract>>
AbstractSyntaxNode
accept(aVisitor)
«use»
accept(aVisitor) {
aVisitor.visitConditional(this)
}
Addition
accept(aVisitor)
MethodDefinition
accept(aVisitor)
Real
accept(aVisitor)
accept(aVisitor) {
aVisitor.visitBlock(this)
}
ReturnTypeVisitor
visitXxxx(Xxxx)
etc…
EmitVisitor
visitBlock(aBlock)
etc…
Block
accept(aVisitor)
ConditionalStatement
accept(aVisitor)
Integer
accept(aVisitor)
visitBlock(aBlock)
<hr>
}
PrettyPrintVisitor
visitBlock(aBlock)
visitWhile(aWhile)
etc…
All are called
visitWhile(aWhile)
“while”
}
WhileStatement
accept(aVisitor)
MethodInvokation
accept(aVisitor)


BENEFITS of the Visitor Pattern
Client
only ONE
method needed
in structure classes (never change)
Pretty Print
Visitor
<<interface>>
Visitor
visitAddition(Addition)
visitMethodDefinition(MethodDefinition)
visitBlock(Block)
visitConditional(ConditionalStatement)
visitReal(Real)
visitInteger(Integer)
visitWhile(WhileStatement)
visitMethodInvokation(MethodInvokation)
AbstractSyntaxTree
traverseTreeWith(aVisitor)
root
1
<<abstract>>
AbstractSyntaxNode
accept(aVisitor)
«use»
Addition
accept(aVisitor)
structure code
is SEPARATED
from pretty print
(never changed)
MethodDefinition
accept(aVisitor)
Real
accept(aVisitor)
ReturnTypeVisitor
visitXxxx(Xxxx)
etc…
EmitVisitor
visitBlock(aBlock)
etc…
Block
accept(aVisitor)
ConditionalStatement
accept(aVisitor)
Integer
accept(aVisitor)
PrettyPrintVisitor
visitBlock(aBlock)
visitWhile(aWhile)
etc…
pretty print
code
is COHESIVE
WhileStatement
accept(aVisitor)
MethodInvokation
accept(aVisitor)