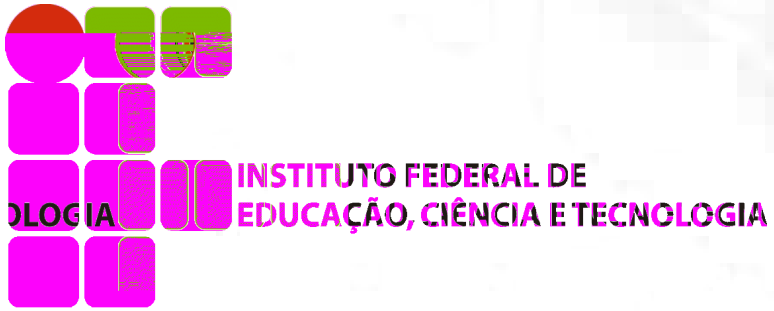
Visitor

Sandro Santos Andrade

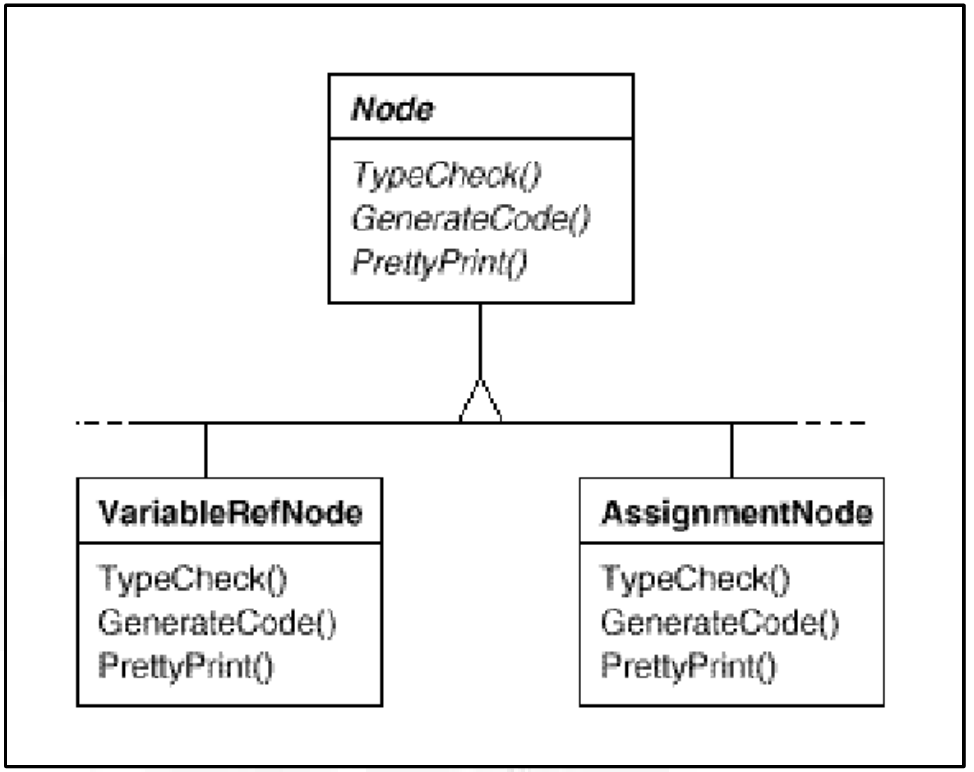
Instituto Federal de Educação, Ciência e Tecnologia da Bahia
Departamento de Tecnologia Eletro-Eletrônica
Graduação Tecnológica em Análise e Desenvolvimento de Sistemas



Visitor

-
  -
  - *Visitor*
-
  -
  -
    -
    -

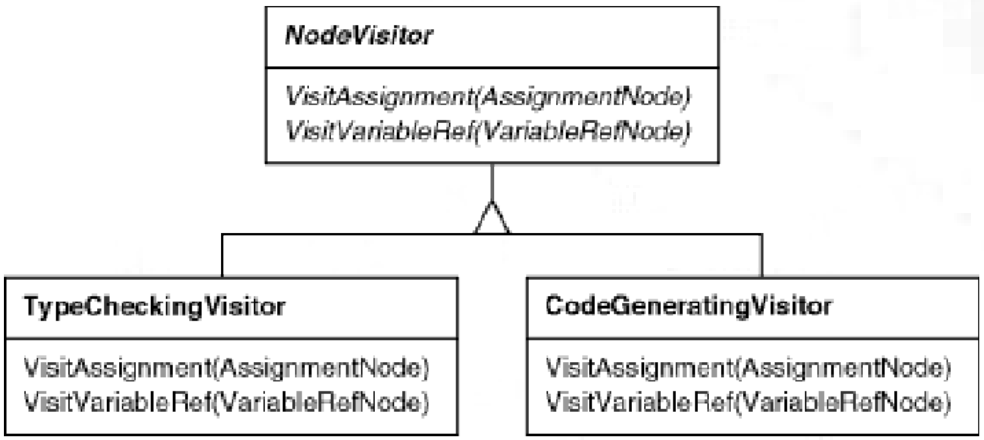
Visitor

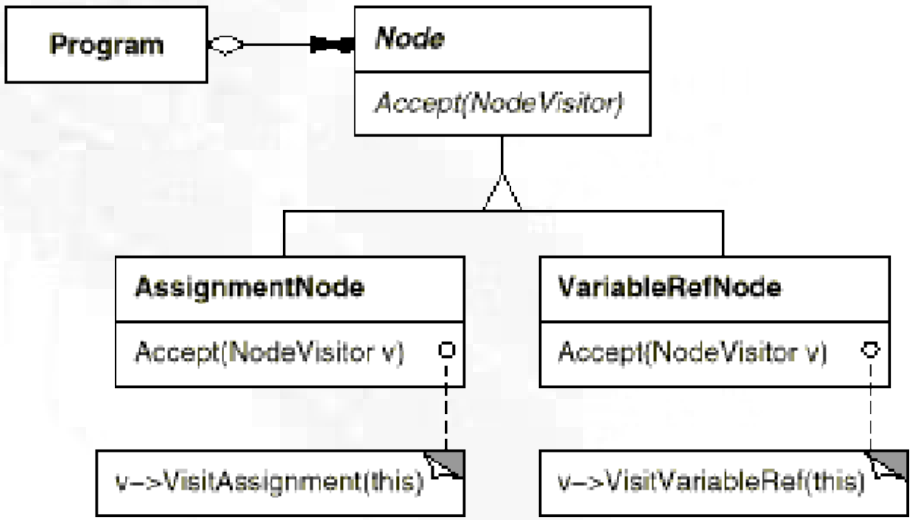

Node
TypeCheck()
GenerateCode()
PrettyPrint()
VariableRefNode
TypeCheck()
GenerateCode()
PrettyPrint()
AssignmentNode
TypeCheck()
GenerateCode()
PrettyPrint()

Visitor

-
  -
  -

Visitor





Visitor

Visitor


Client
Visitor
ConcreteVisitor1
VisitConcreteElementA(Conc
VisitConcreteElementB(Conc

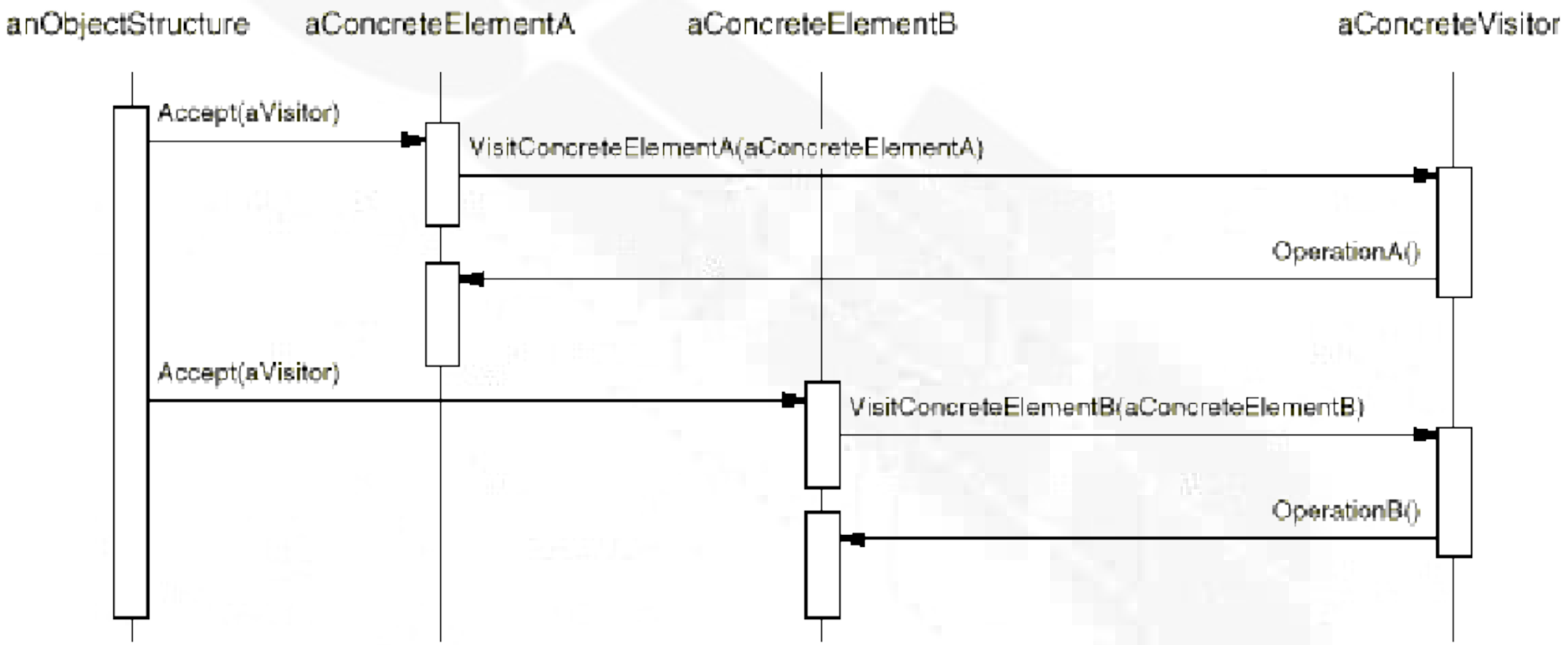
Visitor

- *Visitor*
  - *visit()*
    ConcreteElemente
  - *visit()*
    visit() *Visitor*
  - *Visitor*
- *ConcreteVisitor*
  - *Visitor*
  - *ConcreteVisitor*

Visitor

-
  - *Element*
    - *accept()* *visitor*
  - *ConcreteElement*
    - *accept()*
  - *Agregado*
    -
    -
      Visitor
    - *Composite* *collection*

Visitor


anObjectStructure
aConcreteElementA
aConcreteElementB
aConcreteVisitor
Accept(aVisitor)
VisitConcreteElementA(aConcreteElementA)
OperationA()
Accept(aVisitor)
VisitConcreteElementB(aConcreteElementB)
OperationB()

Visitor

-
  - *Visitor*
  - *Visitor*
  - *Visitor* *ConcreteElements*
  -
    - *Iterator*

```
template <class Item>
class Iterator {
        // ...
        Item CurrentItem() const;
};
```

Visitor

-
  - *Visitor*

```
class Visitor {
public:
        // ...
        void VisitMyType(MyType*);
        void VisitYourType(YourType*);
};
```

Visitor

-
  -
    - *Visitors*

      Visitor
  -
    - *Visitor*

      ConcreteElement

      Visitor

Visitor

-
  - *Double Dispatch*
    -
      -
      - *Double-Dispatch*
      - *Visitor*
        e *Visitor*

Visitor

-
  -
    - *Visitor*
    -
      - *(collection composite)*
      - *Iterator*
      - *Visitor*

ConcreteVisitor

-

Visitor

```
class Equipment {
public:
        virtual ~Equipment();

        const char* Name() { return _name; }

        virtual Watt Power();
        virtual Currency NetPrice();
        virtual Currency DiscountPrice();

        virtual void Accept(EquipmentVisitor&);
protected:
        Equipment(const char*);
private:
        const char* _name;
};
```

```
void FloppyDisk::Accept (EquipmentVisitor& visitor) {
        visitor.VisitFloppyDisk(this);
}
```

Visitor

```
void Chassis::Accept (EquipmentVisitor& visitor) {
        for (
                ListIterator i(_parts);
                !i.IsDone();
                i.Next()
        ) {
                i.CurrentItem()->Accept(visitor);
        }
        visitor.VisitChassis(this);
}
```

Visitor

```
class EquipmentVisitor {
public:
        virtual ~EquipmentVisitor();

        virtual void VisitFloppyDisk(FloppyDisk*);
        virtual void VisitCard(Card*);
        virtual void VisitChassis(Chassis*);
        virtual void VisitBus(Bus*);

        // and so on for other concrete subclasses of Equipment
protected:
        EquipmentVisitor();
};
```

Visitor

```
class PricingVisitor : public EquipmentVisitor {
public:
        PricingVisitor();

        Currency& GetTotalPrice();

        virtual void VisitFloppyDisk(FloppyDisk*);
        virtual void VisitCard(Card*);
        virtual void VisitChassis(Chassis*);
        virtual void VisitBus(Bus*);
        // ...
private:
        Currency _total;
};
```

```
void PricingVisitor::VisitFloppyDisk (FloppyDisk* e) {
        _total += e->NetPrice();
}

void PricingVisitor::VisitChassis (Chassis* e) {
        _total += e->DiscountPrice();
}
```

Visitor

```
class InventoryVisitor : public EquipmentVisitor {
public:
        InventoryVisitor();

        Inventory& GetInventory();

        virtual void VisitFloppyDisk(FloppyDisk*);
        virtual void VisitCard(Card*);
        virtual void VisitChassis(Chassis*);
        virtual void VisitBus(Bus*);
        // ...
private:
        Inventory _inventory;
};
```

```
void InventoryVisitor::VisitFloppyDisk (FloppyDisk* e) {
        _inventory.Accumulate(e);
}

void InventoryVisitor::VisitChassis (Chassis* e) {
        _inventory.Accumulate(e);
}
```

Visitor

```
Equipment* component;
InventoryVisitor visitor;

component->Accept(visitor);
cout << "Inventory "
          << component->Name()
          << visitor.GetInventory();
```

Visitor

-
  - *Smalltalk*
  - *IRIS Inventor*

Visitor

- *Visitors*

  Composite

- *Visitor*

  Interpreter

Visitor

Sandro Santos Andrade

Instituto Federal de Educação, Ciência e Tecnologia da Bahia
Departamento de Tecnologia Eletro-Eletrônica
Graduação Tecnológica em Análise e Desenvolvimento de Sistemas