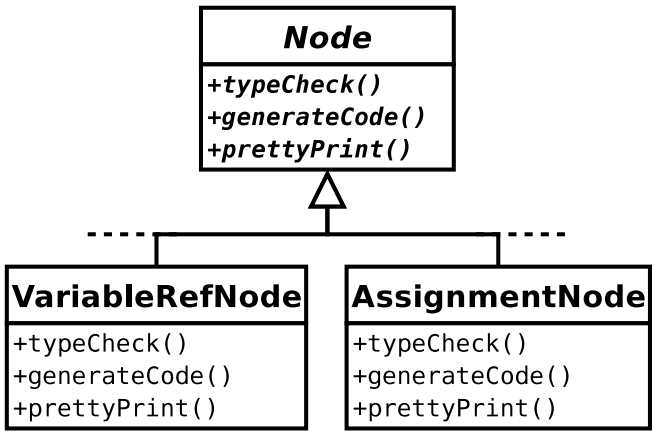
Visitante (Visitor)

- *Patrón de Comportamiento*
- *Propósito*

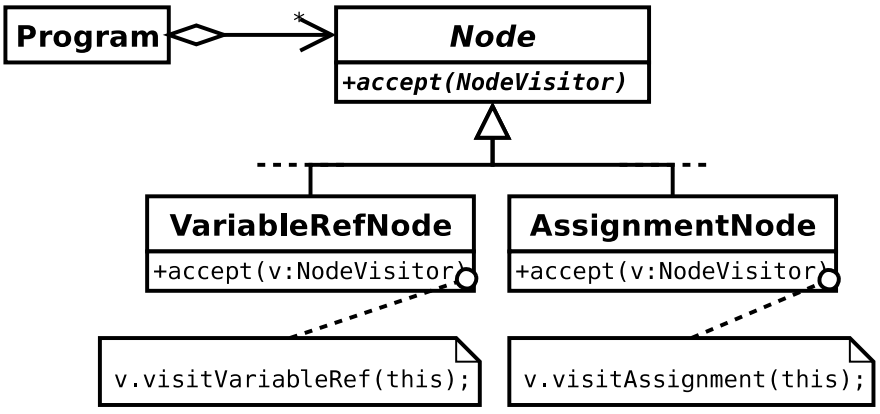
  Permite definir una operación sobre objetos de una jerarquía de clases sin modificar las clases sobre las que opera.

- *Motivación*
  - Compilador que representa programas como árboles con su sintaxis abstracta
  - Necesarias diversas operaciones sobre el árbol sintáctico: Comprobación de tipos, optimización de código, reestructuración de código, generación de código, instrumentación...
  - Cada nodo en el árbol sintáctico necesita un tratamiento diferente: clases

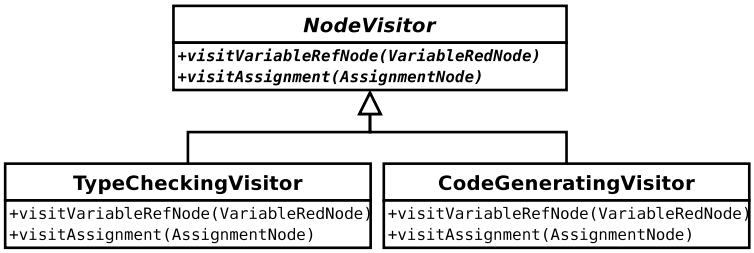


- Motivación (cont.)
  - Problema: Difícil de entender, mantener, modificar y extender puesto que cada clase (variable, asignación,...) tiene su parte correspondiente de cada una de las operaciones
  - Solución: Agrupar las operaciones relacionadas de cada clase en un objeto (*visitante*) y pasarlo como argumento a los elementos del árbol sintáctico.

    El elemento "acepta" al visitante envíandole una petición específica para su clase con él mismo como argumento
    - VariableRef → VisitVariableRef
    - Assignment → VisitAssignment
    - ...

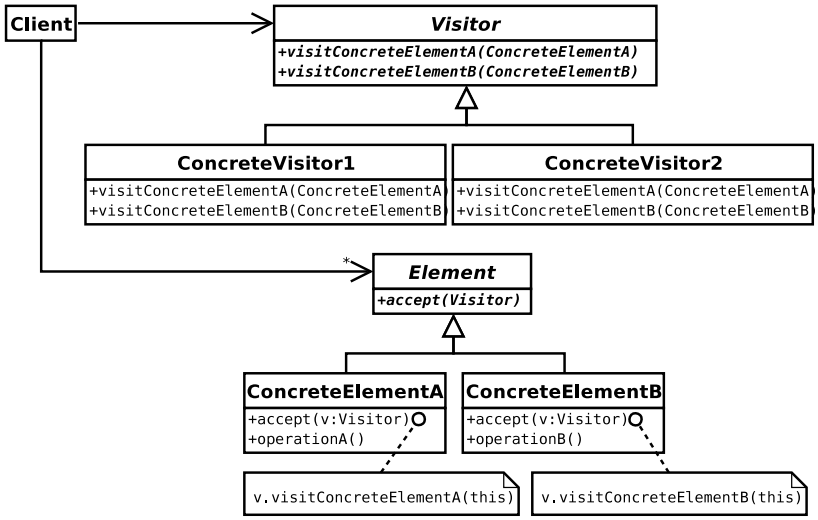


- Motivación (cont.)
  - Para permitir más de un visitante, se define una superclase abstracta con una operación por cada tipo de nodo (2 jerarquías!)



- *Aplicabilidad*
  - Varias clases de objetos con interfaces diferentes y se desean realizar operaciones que dependen de sus clases concretas
  - Se necesitan diversas operaciones (no siempre relacionadas) sobre objetos de una jerarquía y no se desea recargar las clases con estas operaciones
    - Agrupa operaciones relacionadas en una clase
    - Si la jerarquía se usa en diversas aplicaciones, sólo se incluyen las operaciones relevantes
  - Las clases de la jerarquía no cambian, pero se añaden con frecuencia operaciones a la estructura. Si la jerarquía cambia, *NO* es aplicable

- *Estructura*



- *Participantes*

  - Visitante (Visitor)

    Declara una operación de visita para cada elemento concreto en la estructura de objetos, que incluye el propio objeto visitado

  - Visitante Concreto (ConcreteVisitor$_i$)

    Implementa las operaciones del visitante y acumula resultados como estado local

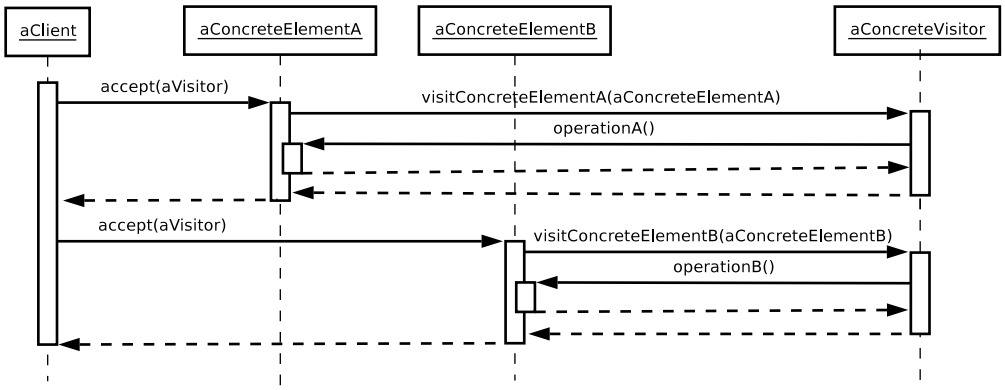
  - Elemento (Element)

    Define una operación "Accept" que toma un visitante como argumento

  - Elemento Concreto (ConcreteElement$_j$)

    Implementa la operación "Accept"

- *Colaboraciones*
  - El cliente debe crear un visitante concreto y luego visitar cada elemento de la estructura de objetos con dicho visitante
  - Cuando se visita un elemento, éste llama a la operación del visitante correspondiente a su clase. Normalmente, el objeto se pasa como argumento para permitir al visitante el acceso a su estado



- *Consecuencias*
  - Añadir operaciones nuevas es sencillo
  - Agrupa operaciones relacionadas y separa las no relacionadas
  - Difícil añadir nuevas clases de elementos
  - No tiene que limitarse a una única jerarquía de elementos
  - Facilita la acumulación de estado
  - Problemas con la encapsulación