Visitor

Návštevník

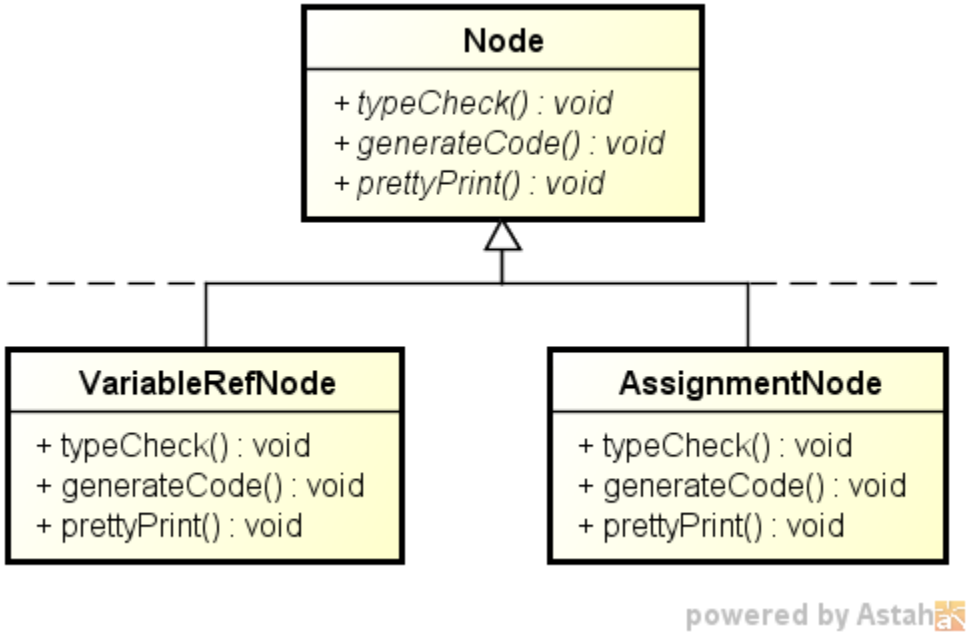
Prestavuje operáciu, ktorú je možné vykonať na prvkoch nejakej objektovej štruktúry. Návštevník umožňuje definovať novú operáciu bez toho, aby boli zmenené triedy prvkov, nad ktorými sa operácia vykonáva.

Represent an operation to be performed on the elements of an object structure. Visitor lets you define a new operation without changing the classes of the elements on which it operates.

Motivácia

- Majme kompilátor reprezentujúci programy ako syntaktické stromy, ktorý potrebuje pre svoju činnosť rôzne operácie, napr.:
  - typové kontroly,
  - analýza a optimalizácia kódu,
  - kontrola inicializácie premenných pred ich prvým použitím,
  - generovanie čitateľného kódu.
- Jednotlivé operácie sa inak správajú k rôznym uzlom syntaktického stromu.

Motivácia

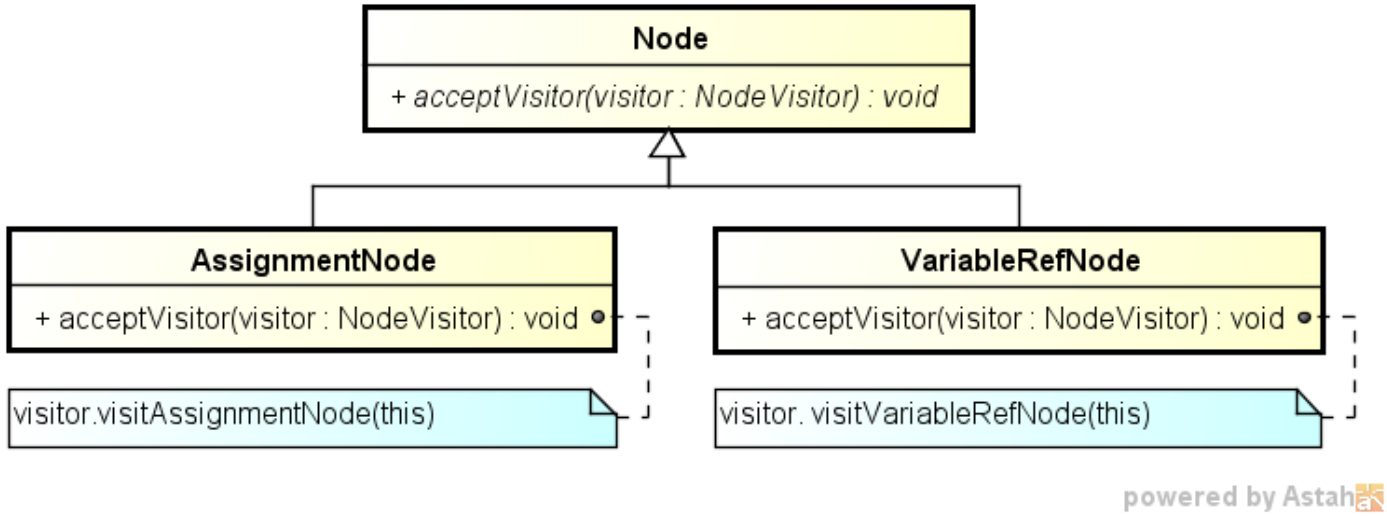


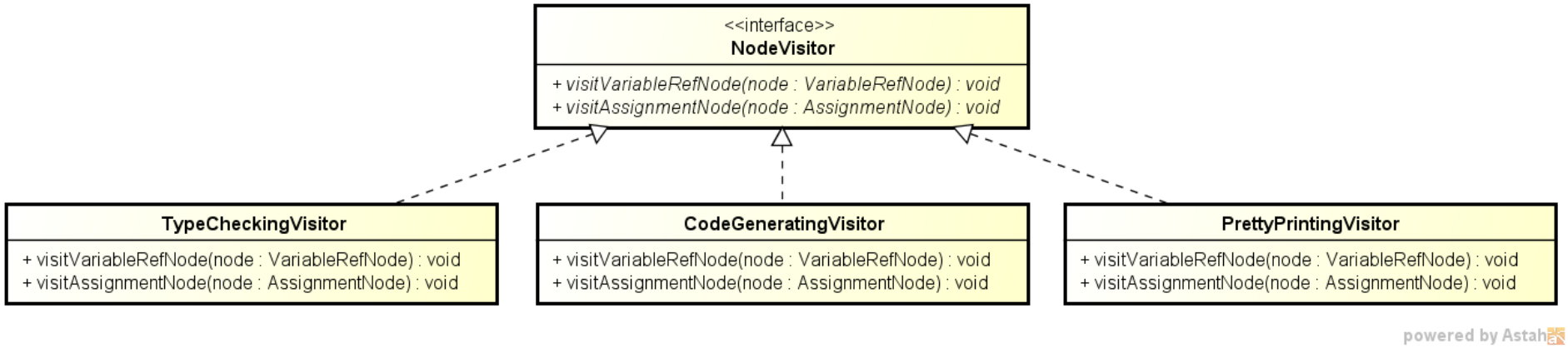
- Problém s takýmto návrhom je, že kódy pre „všetko“ (aj nesúvisiace) sú definované v rámci jednej triedy, čo môže spraviť systém neprehľadný a ťažko spravovateľný.

Motivácia

- Rozdelenie na dve hierarchie:
  1. jedna pre zapúzdrenie funkčnosti – Návštevník (NodeVisitor),
  2. jedna pre triedy, ktorých funkčnosť rozširujú (Node).
- Odpadá nutnosť použitia kontroly typov.
- NodeVisitor **musí** deklarovať metódu pre **každú** Node.

Motivácia

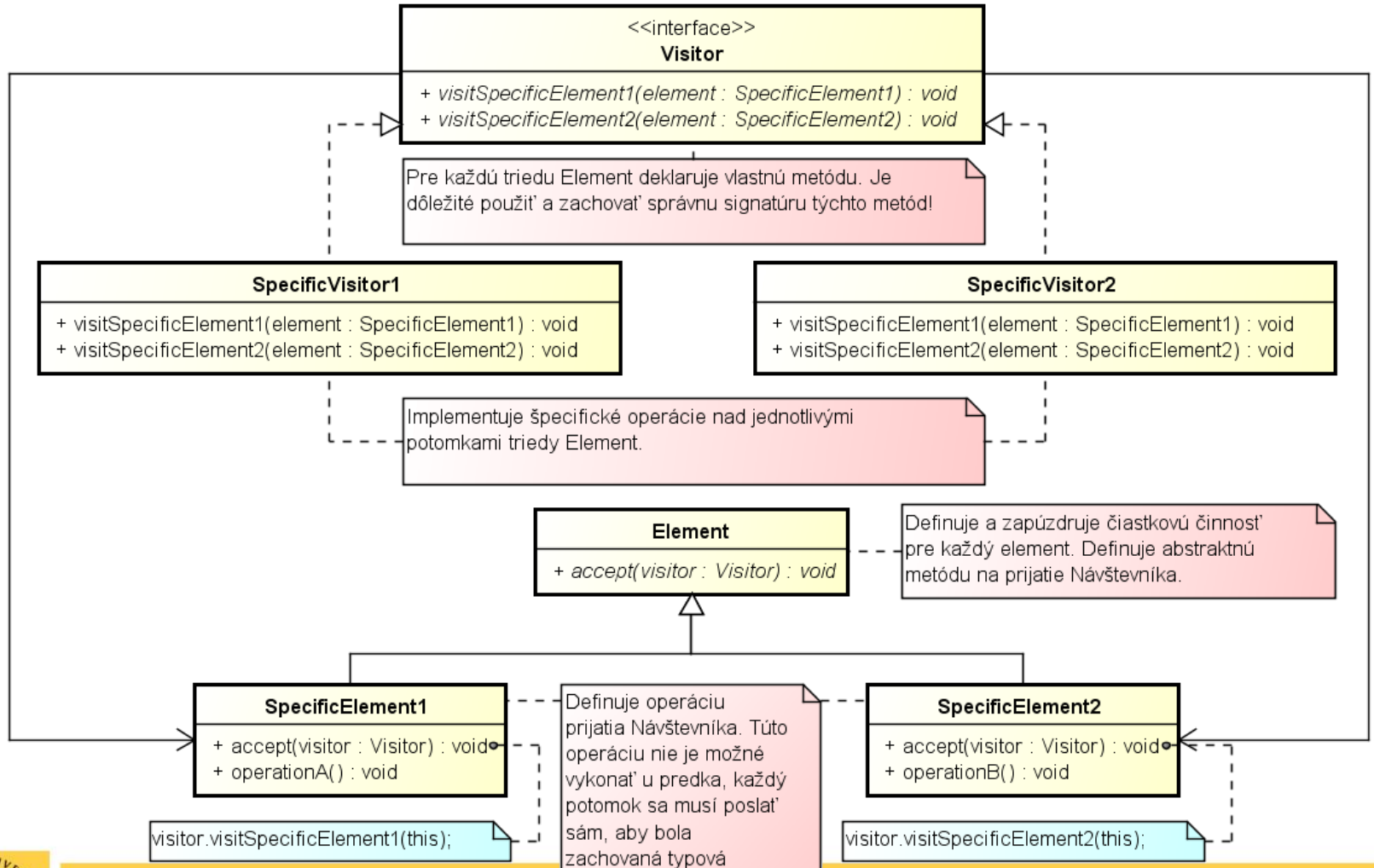

Node
+ acceptVisitor(visitor : NodeVisitor) : void
AssignmentNode
+ acceptVisitor(visitor : NodeVisitor) : void
visitor.visitAssignmentNode(this)
VariableRefNode
+ acceptVisitor(visitor : NodeVisitor) : void
visitor. visitVariableRefNode(this)
powered by Astah


<<interface>>
NodeVisitor
+ visitVariableRefNode(node : VariableRefNode) : void
+ visitAssignmentNode(node : AssignmentNode) : void
TypeCheckingVisitor
+ visitVariableRefNode(node : VariableRefNode) : void
+ visitAssignmentNode(node : AssignmentNode) : void
CodeGeneratingVisitor
+ visitVariableRefNode(node : VariableRefNode) : void
+ visitAssignmentNode(node : AssignmentNode) : void
PrettyPrintingVisitor
+ visitVariableRefNode(node : VariableRefNode) : void
+ visitAssignmentNode(node : AssignmentNode) : void
powered by Astah

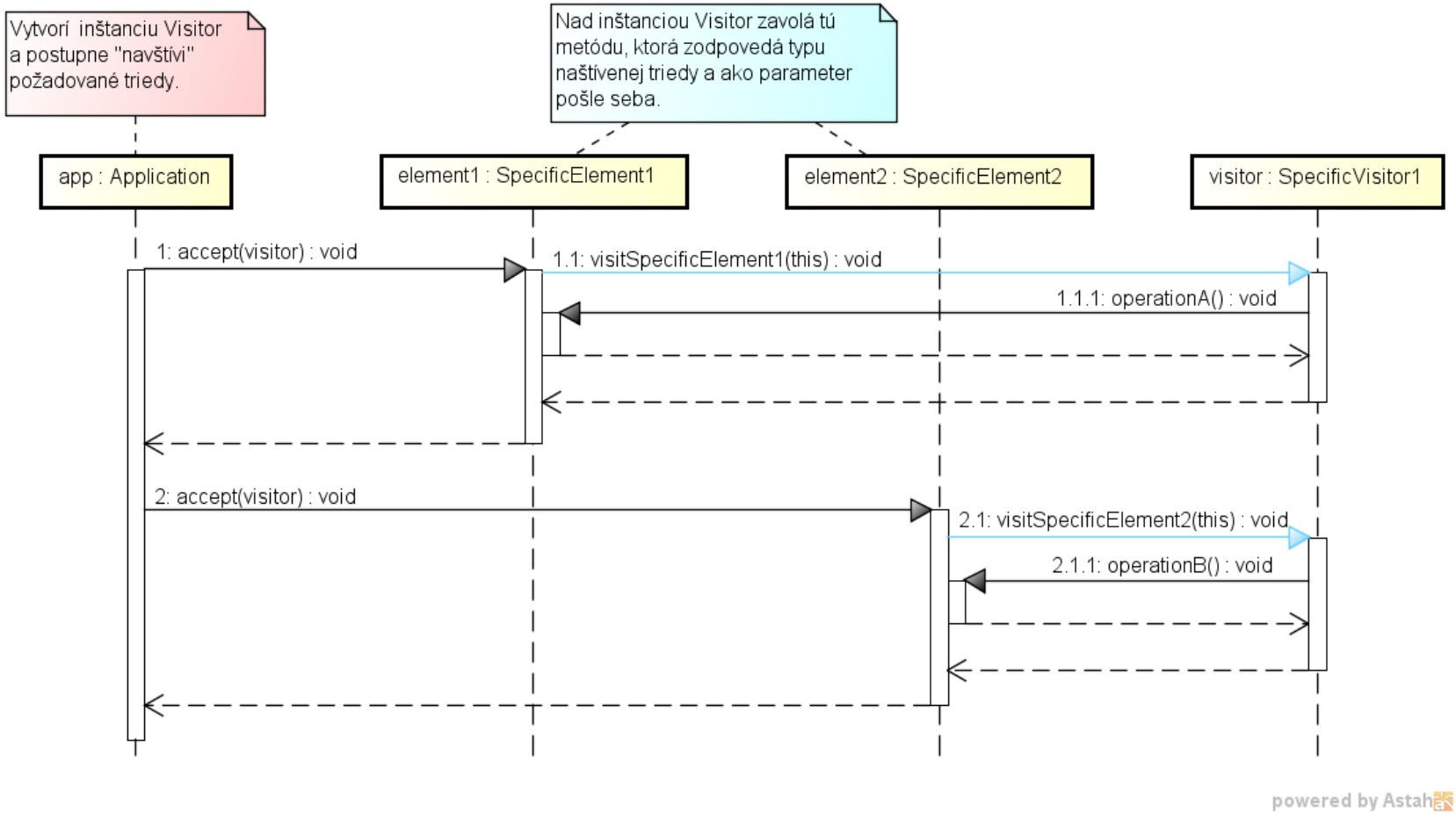
Aplikovateľnosť

- Návštevníka je vhodné použiť, keď:
  - objektová štruktúra obsahuje mnoho tried objektov s rôznymi rozhraniami a chcete na týchto objektoch vykonávať operácie, ktoré závisia na konkrétnej triede.
  - má byť na objektoch v objektovej štruktúre vykonaných mnoho rozličných alebo nesúvisiacich operácií a chcete, aby tieto operácie nemuseli byť definované priamo v triede. Návštevník umožňuje spojiť súvisiace operácie do jednej triedy. Keď je objektová štruktúra zdieľaná viacerými aplikáciami, tak jednotlivé aplikácie definujú iba takých Návštevníkov, ktorých potrebujú.
  - sa triedy definujúce štruktúru objektov menia iba sporadicky, ale často chcete definovať nové operácie nad štruktúrou. Naopak, častá zmena objektovej štruktúry môže spôsobiť opätovné definovanie rozhraní všetkých Návštevníkov, čo môže byť potenciálne zložité. V takomto prípade je lepšie definovať operácie priamo v triedach.

Implementácia



Spolupráca


Vytvorí inštanciu Visitor a postupne "navštívi" požadované triedy.
Nad inštanciou Visitor zavolá tú metódu, ktorá zodpovedá typu naštívenej triedy a ako parameter pošle seba.
app : Application
element1 : SpecificElement1
element2 : SpecificElement2
visitor : SpecificVisitor1
1: accept(visitor) : void
1.1: visitSpecificElement1(this) : void
1.1.1: operationA() : void
2: accept(visitor) : void
2.1: visitSpecificElement2(this) : void
2.1.1: operationB() : void
powered by Astah

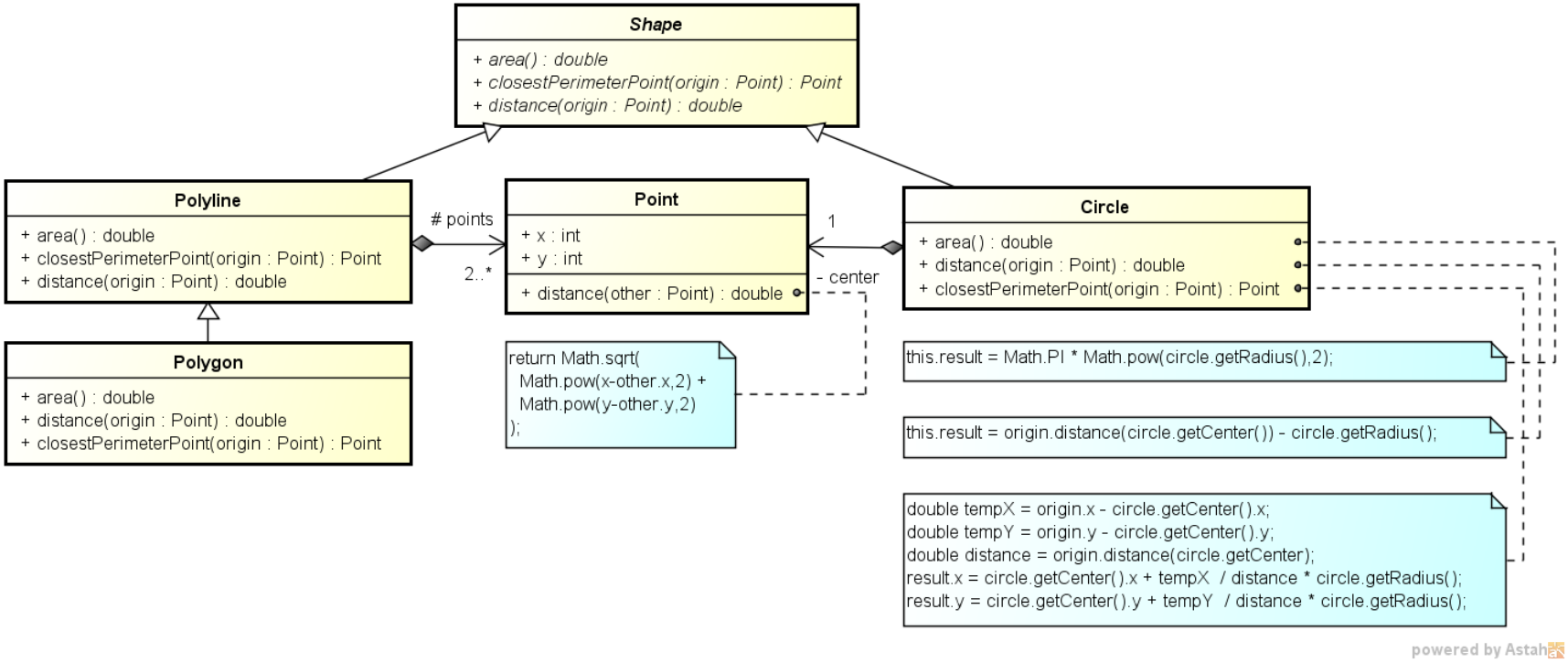
Dôsledky

- Zavedením ďalšieho Návštevníka umožňuje jednoducho pridávať nové operácie. Ak je však funkčnosť rozdelená medzi viacero tried, je pre pridanie operácie potrebné upraviť každú z nich.
- Dokáže spojiť súvisiace operácie a oddeliť tie nesúvisiace, nie je teda nutné, aby každá trieda v štruktúre poznala správanie – to je centralizované v Návštevníkovi, čím sa triedy sprehľadnia a vyjmú sa z nich štruktúry, od ktorých závisia špecifické algoritmy.
- Je komplikované pridať novú triedu typu SpecificElement. Nový potomok spôsobí pridanie novej abstraktnej metódy do Návštevníka, čo spôsobí jej pridanie a definovanie do každého jeho potomka.

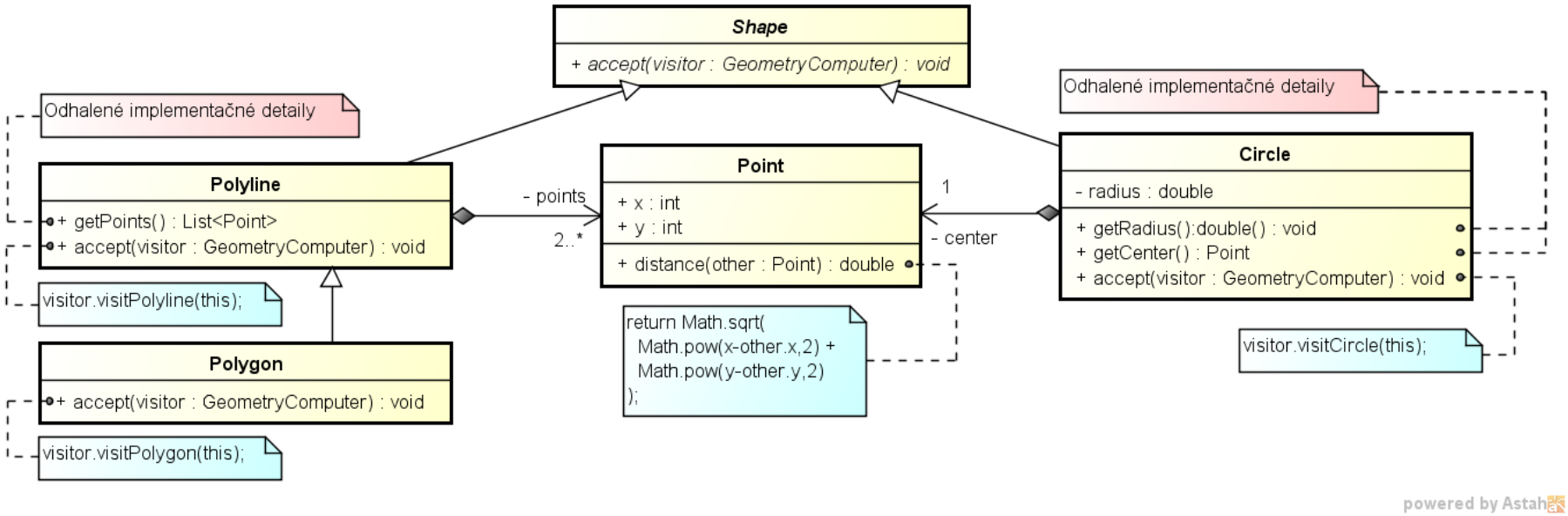
Dôsledky

- Nie je viazaný na jednu konkrétnu objektovú hierarchiu. Ak sa má použiť v inej objektovej hierarchii, Návštevník jednoducho pridá novú abstraktnú metódu do svojho rozhrania.
- Môže porušiť zapúzdrenie! Aby Návštevník mohol vykonať svoju činnosť, tak potenciálne potrebuje mať prístup k prvkom objektu, ktoré by inak mali ostať skryté.

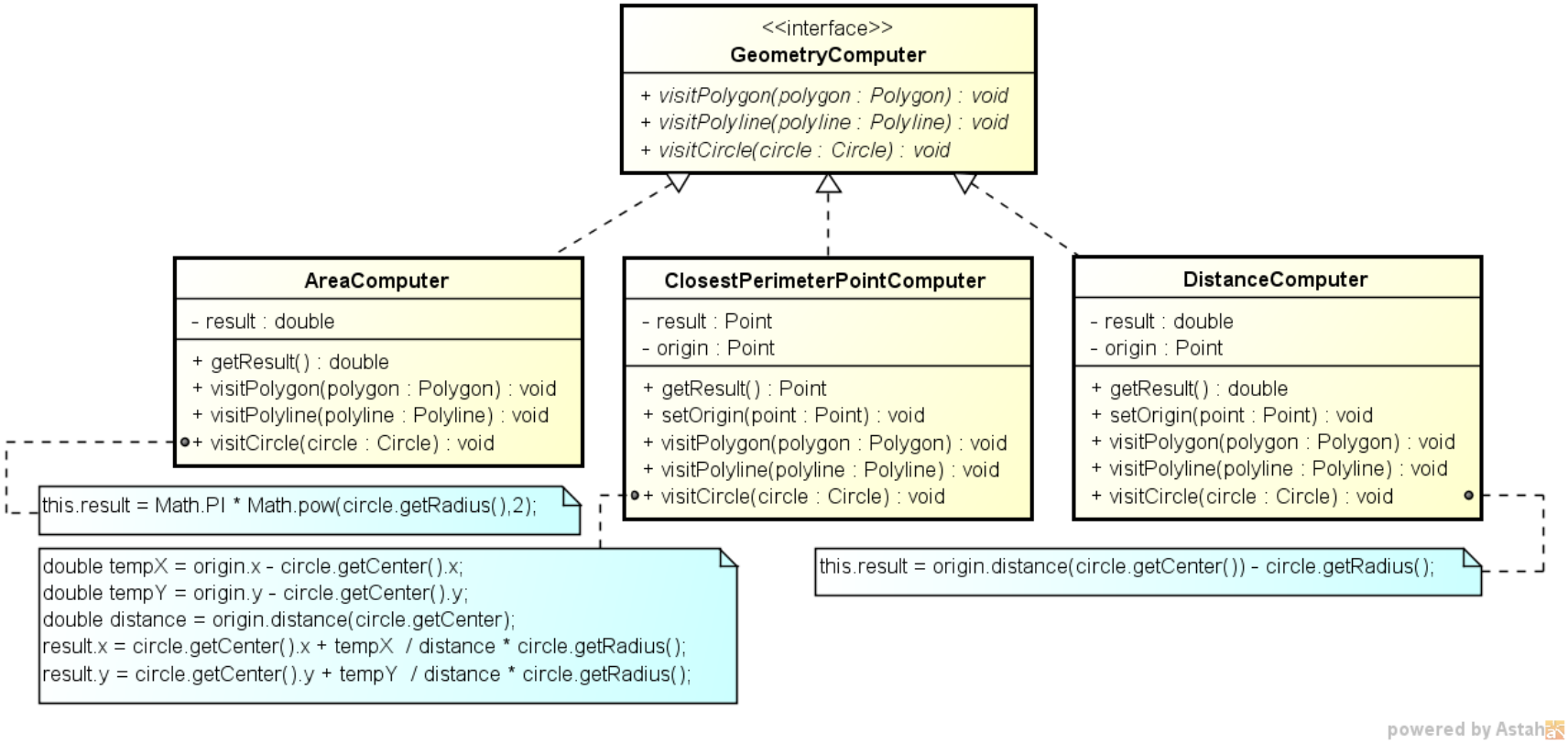
Príklad – bez návštevníka



Príklad – hierarchia tvarov



Príklad – hierarchia návštevníkov



Príbuzné vzory

- *Composite* – Návštevníci môžu byť využití na aplikáciu operácií nad štruktúrou objektov vytvorených ako Kompozity.
- *Interpreter* – Návštevník môže vykonávať interpretáciu.

Upozornenie

- Tieto študijné materiály sú určené výhradne pre študentov predmetu 5II132 Návrhové vzory (Design Patterns) na Fakulte riadenia a informatiky Žilinskej univerzity v Žiline.
- Reprodukovanie, šírenie (i častí) materiálov bez písomného súhlasu autora nie je dovolené.

Ing. Michal Varga, PhD.
Katedra informatiky
Fakulta riadenia a informatiky
Žilinská univerzita v Žiline
Michal.Varga@fri.uniza.sk