Today

- Quiz 3
- Abstract Syntax Trees (cont)

Announcements

- HW-3 due fri
- Exercises ...
The **Visitor Design Pattern**

The Visitor pattern allows:

1. functions over an object structure (like an AST) to be decoupled from the object structure itself

2. this means you can have many different functions, without having to change the object structure

- an object (node) in the structure “accepts” a visitor
- which means the node simply passes itself to the visitor
- the visitor then “visits” the node (e.g., prints, evaluates, etc.)
- and then “navigates” to child nodes (repeating the process)

For those interested in more details, see “double dispatch” ...
A simple/hypothetical example

In $\text{PrintVisitor}$:

```java
public class PrintVisitor {

    public void visit(ValueNode v) {
        System.out.print(v.value);
    }

    public void visit(PlusNode v) {
        v.leftExpr.accept(this);
        System.out.print(" + ");
        v.rightExpr.accept(this);
    }

    public void visit(TimesNode v) {
        v.leftExpr.accept(this);
        System.out.print(" * ");
        v.rightExpr.accept(this);
    }

}
```
How we’ll use the Visitor Pattern:

- In HW-4 to “pretty” print the program from the AST
- In HW-5 for “type checking” (semantically analyzing) the AST
- As part of implementing the interpreter (translation to VM code)

All without modifying the AST classes (just via different Visitor implementations)