Visiting the Classes

Now all that we must do to compute the total vacation days taken is to go through a list of the employees, visit each of them, and then ask the Visitor for the total.

```
        VacationVisitor vac = new VacationVisitor();
        for (int i = 0; i < employees.length; i++) {
            employees[i].accept(vac);
    }
    total.setText(new Integer(vac.getTotalDays()).toString());
```

Here's what happens for each visit:

1. We move through a loop of all Employees.
2. The Visitor calls each Employee's *accept* method.
3. That instance of Employee calls the Visitor's *visit* method.
4. The Visitor fetches the value for the number of vacation days and adds that number to the total.
5. The main program prints out the total number of days when the loop is complete.

Visiting Several Classes

The Visitor becomes more useful when we have several different classes with different interfaces and we want to encapsulate how we get data from these classes. Let's extend our vacation days model by introducing a new Employee type called Boss. We further suppose that at this company, Bosses are rewarded with bonus vacation days (instead of money). So the Boss class has a couple of extra methods to set and obtain the bonus vacation days information.

```
public class Boss extends Employee {
    private int bonusDays;

    public Boss(String name, float salary, int vacdays,
            int sickdays) {
        super(name, salary, vacdays, sickdays);
    }
    public void setBonusDays (int bonus) {
        bonusDays = bonus;
    }
    public int getBonusDays() {
        return bonusDays;
    }
    public void accept(Visitor v) {
```