An Elephant can use its trunk to grab the tail of another elephant (see image where Elephant e2 is holding onto the tail of e1). In this case e1 is e2’s leader, and e2 is e1’s trailer. Note that e2 grabs e1’s tail by grabTail(e1) and e2 might lose its grip on e1’s tail, by releaseTail();

There are several types of Visitors to Elephants. One Visitor is the Trainer, which first picks out an elephant to be the leader (by the selectLeader method), and then visits another elephant, which then is linked up so that it is trailing behind that leader. For example,

```java
Elephant e1 = new Elephant();
Elephant e2 = new Elephant();
Elephant e3 = new Elephant();
Trainer t = new Trainer();

t.selectLeader(e1); // e1 is chosen to be the front of the line
e2.acceptVisitor(t); // trainer t gets e2 to grab e1’s tail
t.selectLeader(e2); // next, t selects e2
e3.acceptVisitor(t); // and gets e3 to grab e2’s tail.
```

Another Visitor is in fact another Elephant, which if it visits any elephant, will go down the line until it finds the end (an elephant with no trailing elephant) and then grabs on to that last elephant’s tail.

The other visitor to an Elephant is a Mouse, which scares any Elephant that it visits. The visit makes the elephant release the tail it is holding onto (if any), but it also changes the state of the elephant to Scared. Note that an elephant is initially in a Calm state (and if scared by a mouse, a subsequent visit from the trainer puts it back into the Calm state). The elephant has one state-dependent method, namely grabTail, which only works correctly in the Calm state. Presume grabTail just returns immediately in the Scared state.

Finish writing Elephant, Visitor, Trainer, Mouse, State, Scared, and Calm. Remember an Elephant is also a Visitor.

**IMPORTANT:** You must make Visitor an interface. Do not make Visitor an abstract base class.
public class Elephant implements Visitor {
    private Elephant leader;
    private Elephant trailer;
    private State currentState;
    private Scared scared; // please assign these instance variables
    private Calm calm; // please assign these instance variables

    public Elephant() {
        scared = new Scared(this);
        calm = new Calm(this);
        setCalm();
    }

    public void setCalm() {
        currentState = calm;
    }

    public void setScared() {
        currentState = scared;
    }

    public void grabTail(Elephant e) {
        currentState.grabTail();
    }

    public void releaseTail() {
        if (leader == null) return;
        leader.setTrailer(null); // remember the elephant in front!
        leader = null;
    }

    public Elephant getLeader() { return leader; }
    public void setLeader(Elephant e) { leader = e; }

    public Elephant getTrailer() { return trailer; }
    public void setTrailer(Elephant e) { trailer = e; }

    public boolean hasLeader() { return leader != null; }
    public boolean hasTrailer() { return trailer != null; }

    public void acceptVisitor(Visitor v) { v.visit(this); }

    // go down to the end of the line and grab the last elephant’s tail.
    // be careful to not make a null pointer exception!
    public void visit(Elephant e) {
        Elephant trailer = e.hasTrailer() ? e.getTrailer() : e;
        while (trailer.hasTrailer())
            trailer = trailer.getTrailer();
        grabTail(trailer);
    }
}
abstract public class State {
    protected Elephant parent;     // use this instance variable!!

    public State(Elephant parent) { this.parent = parent; }

    abstract public void grabTail(Elephant e);
}

public class Calm extends State {
    public Calm(Elephant e) { super(e); }
    public void grabTail(Elephant e) {
        parent.setLeader(e);
        e.setTrailer(parent);
    }
}

public class Scared extends State {
    public Scared(Elephant e) { super(e); }

    public void grabTail(Elephant e) {}
// remember you need only deal with one kind of visitee

public interface Visitor {
    public void visit(Elephant e);
}

/* One Visitor is the Trainer, which first picks out an elephant to be the leader (by the selectLeader method), and then visits another elephant. The visit first makes the elephant calm, and then the elephant is linked up so that it is trailing behind that leader. */

public class Trainer implements Visitor() {
    private Elephant leader;

    public void selectLeader(Elephant e) {
        leader = e;
    }

    public void visit(Elephant e) {
        e.setCalm();
        e.grabTail(leader);
    }
}

/* Another visitor to an Elephant is a Mouse, which scares any Elephant that it visits. The visit makes the elephant release the tail it is holding onto (if any), but it also changes the state of the elephant to Scared. */

public class Mouse implements Visitor() {
    public void visit(Elephant e) {
        e.releaseTail();
        e.setScared();
    }
}