```java
public class C1 {
    int counter = 0;
    // a state modifying constructor
    C1() {
        counter++;
    }
    /* the author of this class assumes extensions but no overriding and therefore uses
     * cross-invocations among class methods */
    void A() {
        if (counter%2 != 0) {
            /* the danger lurks in this cross-call because in case of overriding "this"
             * will redispatch to the context of the call!
             *
             * (the condition for the cross-call may obviously be much less obvious and deterministic!) */
            this.B();
        }
    }
    void B() {
        System.out.println("Counter evaluates to: " + counter + "\n");
    }
}
```

```java
public class C2 extends C1 {
    // class state
    int increment = 0;

    C2(int step) {
        super();
        increment = increment + step;
    }

    /* the author of this class is unaware (perhaps guiltily) of the superclass design
     * assumption that disallows overriding */
    void B() {
        counter = counter + increment;
        /* the public documentation on super.A() may (innocently) omit implementation details
         * so that the author of C2 may like what (s)he reads about super.A() without getting to know
         * about its dangerous cross-call */
        this.A();
    }
}
```

```
class EP {
    public static void main(String[] args) {
        int i;
        int step = 0; // default initialisation
        try {
            step = args[0].length();
        } catch (java.lang.ArrayIndexOutOfBoundsException e) {
            System.out.println("Usage: java EP input_string");
        }
        C2 instance = new C2(step);
        for (i = 1; i <= step ; i++) {
            instance.B();
            System.out.println
                ("Iteration #" + i + " when counter evaluates to: " + instance.counter);
        }
    }
}
```