```
// Wrapper class - uses composition in place of inheritance
public class InstrumentedSet implements Set {
    private final Set s;
    private int addCount = 0;

    public InstrumentedSet(Set s) {
        this.s = s;
    }

    public boolean add(Object o) {
        addCount++;
        return s.add(o);
    }

    public boolean addAll(Collection c) {
        addCount += c.size();
        return s.addAll(c);
    }

    public int getAddCount() {
        return addCount;
    }

    // Forwarding methods
    public void clear()                { s.clear();              }
    public boolean contains(Object o) { return s.contains(o); }
    public boolean isEmpty()          { return s.isEmpty();    }
    public int size()                 { return s.size();       }
    public Iterator iterator()        { return s.iterator();   }
    public boolean remove(Object o)   { return s.remove(o);    }
    public boolean containsAll(Collection c)
                                   { return s.containsAll(c); }
    public boolean removeAll(Collection c)
                                   { return s.removeAll(c);   }
    public boolean retainAll(Collection c)
                                   { return s.retainAll(c);   }
    public Object[] toArray()           { return s.toArray();  }
    public Object[] toArray(Object[] a) { return s.toArray(a); }
    public boolean equals(Object o)     { return s.equals(o);  }
    public int hashCode()               { return s.hashCode(); }
    public String toString()            { return s.toString(); }
}
```