sion point as we had in our simple Namer example. Instead, the decision as to which Event class to instantiate determines which Seeding class will be instantiated.

While it looks like there is a one-to-one correspondence between the Seeding and Event class hierarchies, there needn't be. There could be many kinds of Events and only a few kinds of Seeding that they use.

The Swimmer Class

We haven't said much about the Swimmer class, except that it contains a name, club, age, seed time, and place to put the heat and lane after seeding. The Event class reads in the Swimmers to a Vector from some database (a file, in this example) and then passes that Vector to the Seeding class when we call the *getSeeding* method for that event.

The Event Classes

We have seen the abstract base Event class earlier. We actually use it to read in the swimmer data (here from a file) and pass it on to instances of the Swimmer class to parse.

```
public abstract class Event {
    protected int    numLanes;
                                        //number of lanes
    protected Vector swimmers;          //list of swimmers

    public Event(String filename, int lanes) {
        numLanes = lanes;
        swimmers = new Vector();
        //read in swimmers from file
        InputFile f = new InputFile(filename);
        String s = f.readLine();
        while(s != null) {
            Swimmer sw = new Swimmer(s);
            swimmers.addElement(sw);
            s = f.readLine();
        }
        f.close();
    }
    public abstract Seeding getSeeding();
    public abstract boolean isPrelim();
    public abstract boolean isFinal();
    public abstract boolean isTimedFinal();
}
```