```
            if(numHeats >= 2 ) {
               if(numHeats >= 3)
                    circle = 3;
             else
                    circle = 2;
             int i= 0;
             for(int j = 0; j < numLanes; j++) {
                    for(int k = 0; k < circle; k++) {
                           swmrs[i].setLane(lanes[j]);
                           swmrs[i++].setHeat(numHeats - k);
                    }
             }
            }
        }
}
```

Since the constructor calls the parent class constructor, it copies the swimmer Vector and lane values. Then a call to *super.seed()* does the straight seeding. This simplifies things because we will always need to seed the remaining heats by straight seeding. Then we seed the last 2 or 3 heats as shown previously, and we are done with that type of seeding as well.

The Seeding Program

In this example, we took from the Web a list of swimmers who had competed in the 500-yard freestyle and the 100-yard freestyle and used them to build our TimedFinalEvent and PrelimEvent classes. You can see the results of these two seedings in Figure 4.2.

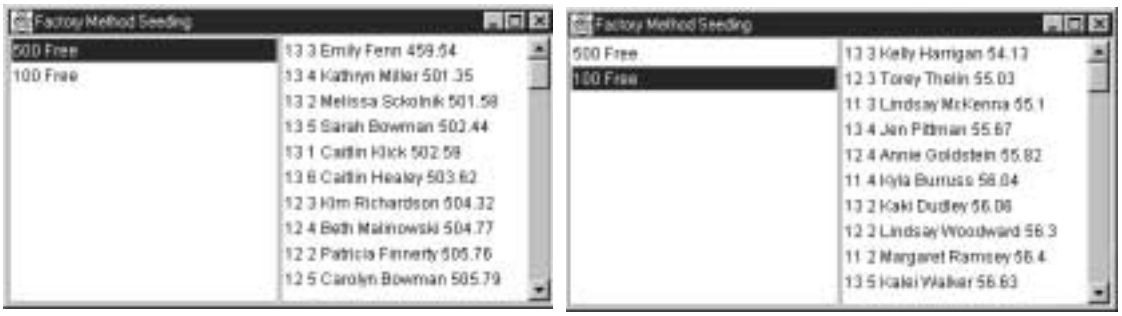

Figure 4.2 Straight seeding of the 500-yard and circle seeding of the 100-yard freestyles.

Other Factories

Now one issue that we have skipped over is how the program that reads in the swimmer data decides which kind of event to generate. We finesse this here by calling the two constructors directly: