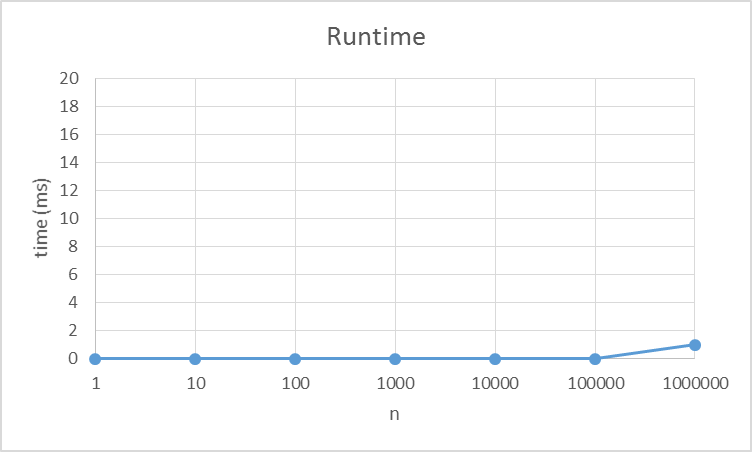
**Program 6**



*Comparison*

It’s one thing to be able to write a program. It’s another to comprehend its runtime. The runtime becomes an incredibly important attribute when it comes to writing programs. Being able to understand how the program functions lays leverage the basic concept of how it is structured. Which is why my program 6 uses a binary search to lookup a name or phone number. In using a binary search this allows the program to run its search in O (log (n)) time. Therefore it takes only a few comparisons to locate the target item. If a 1000 items or in this case person objects are searched. The search at most looks at 10 person objects. This allows the search to run in an amazingly short amount of time. The task happens so fast that all the data sets used up until 1000000 register as 0 milliseconds. A 1000000 person objects results in 20 examinations. When the data set increase by a 1000000 the time grows by a factor of 1 millisecond. Although there is some other draw backs on sorting and storage which aren’t being taken into consideration in this data.

*Big Bonus*

This concept is a highly sought after implementation and yet it has no correct answer. One person may devise a complete different structure then another, but generate the same results. The problem arise in this question all its own. It is not plausible to eliminate all the overhead. However you can reduce the storage and/or almost eliminate the sorting with parallel array lists. Starter with a sorted master list which can be used for either searching names or numbers. Then parallel lists may be kept to contain indices in there relevant order for the other fields of the data. This in concept lessens the strain on storage from holding duplicate data and the problematic resort. However this is probably only one of the numerals ways of theoretical solving this problem.