When you create a List<T> specifying Person objects, it is as if the List<T> type were defined as

follows:

namespace System.Collections.Generic

{

public class List<Person> :

IList<Person>, ICollection<Person>, IEnumerable<Person>, IReadOnlyList<Person>

IList, ICollection, IEnumerable

{

...

public void Add(Person item);

public ReadOnlyCollection<Person> AsReadOnly();

public int BinarySearch(Person item);

public bool Contains(Person item);

public void CopyTo(Person[] array);

public int FindIndex(System.Predicate<Person> match);

public Person FindLast(System.Predicate<Person> match);

public bool Remove(Person item);

public int RemoveAll(System.Predicate<Person> match);

public Person[] ToArray();

public bool TrueForAll(System.Predicate<Person> match);

public Person this[int index] { get; set; }

}

}

Of course, when you create a generic List<T> variable, the compiler does not literally create a brand

new implementation of the List<T> class. Rather, it will address only the members of the generic type

you actually invoke.