C# Generics and Collections (Data Structures)

INSTRUCTOR: ADEEN BIN WAHEED

LECTURE 10

Parameterized Classes and Generics

The classes that have type parameters are called *parameterized class or* simply *generics.*

For Example:

The class LinkedList is a *parameterized class.*

It has a parameter, denoted by Base_Type, that can be replaced by any reference type to obtain a class for LinkedList with the specified base type as shown below.

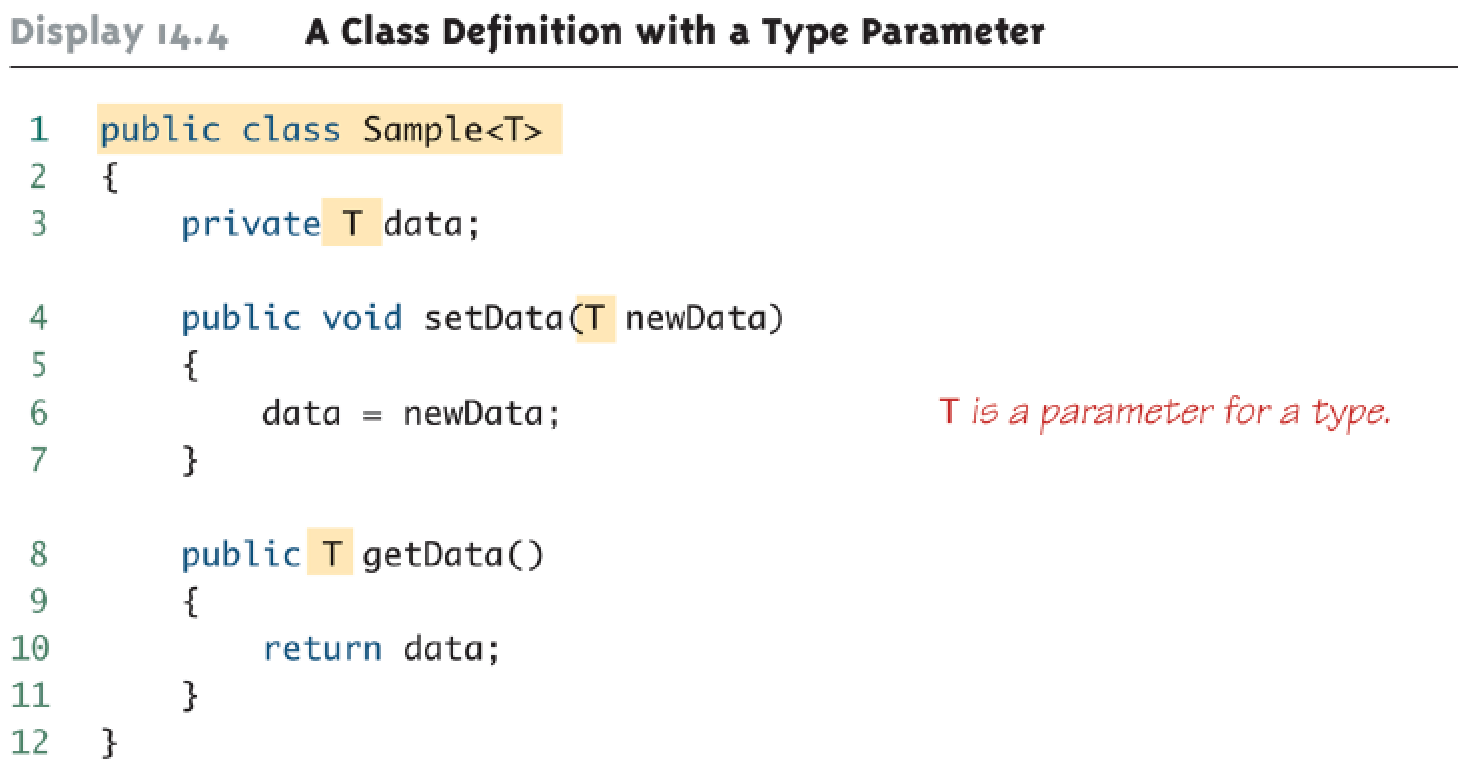
```
LinkedList <String> l=new LinkedList<String>();
```

A Class Definition with a Type Parameter

Some Key Points:

- The type parameter is included in angular brackets after the class name in the class definition heading.
- Any non-keyword identifier can be used for the type parameter, but by convention, the parameter starts with an uppercase letter.
- The type parameter can be used like other types used in the definition of a class.

Similarly we can make our own parameterized or generic classes



Display 14.4 A Class Definition with a Type Parameter

```java
public class Sample<T>
{
    private T data;

    public void setData(T newData)
    {
        data = newData;
    }

    public T getData()
    {
        return data;
    }
}
```

T is a parameter for a type.

Generic Interfaces

An interface can have one or more type parameters.

The details and notation are the same as they are for classes with type parameters.

Generic Methods

Syntax:

The type parameter must be placed (in angular brackets) after the method name, and before the parameters:

public T genMethod <T>(T[] a)

When one of these generic methods is invoked, the method name is prefaced with the type to be plugged in, enclosed in angular brackets

obj.<String>genMethod(c);

Parameterized or Generic Example

Running Demo

Data Structures

Data Structures or Collections are our data Storage Elements

- Linked List
- Stack
- Queue
- Trees
- Graphs

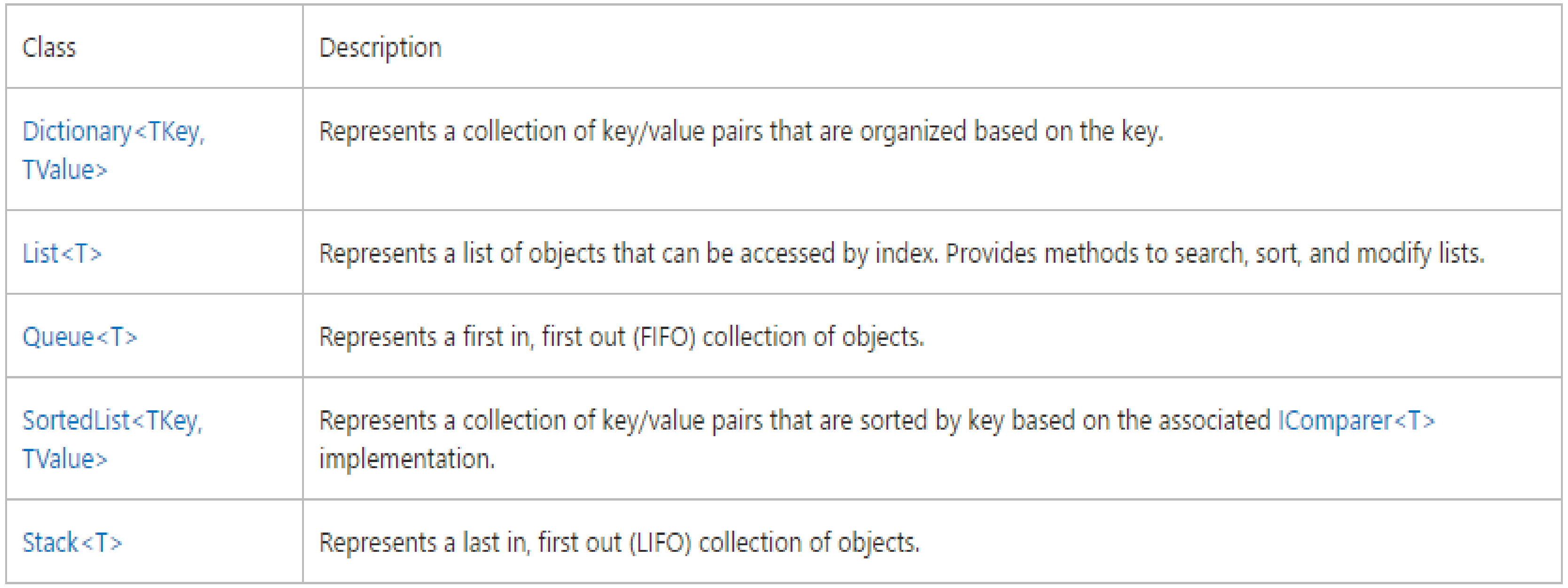
Collections in C#

Collection classes are specialized classes for data storage and retrieval. These classes provide support for stacks, queues, lists, and hash tables. Most collection classes implement the same interfaces.

The following are the various commonly used classes of the**System.Collection** namespace.

System.Collections.Generic Classes

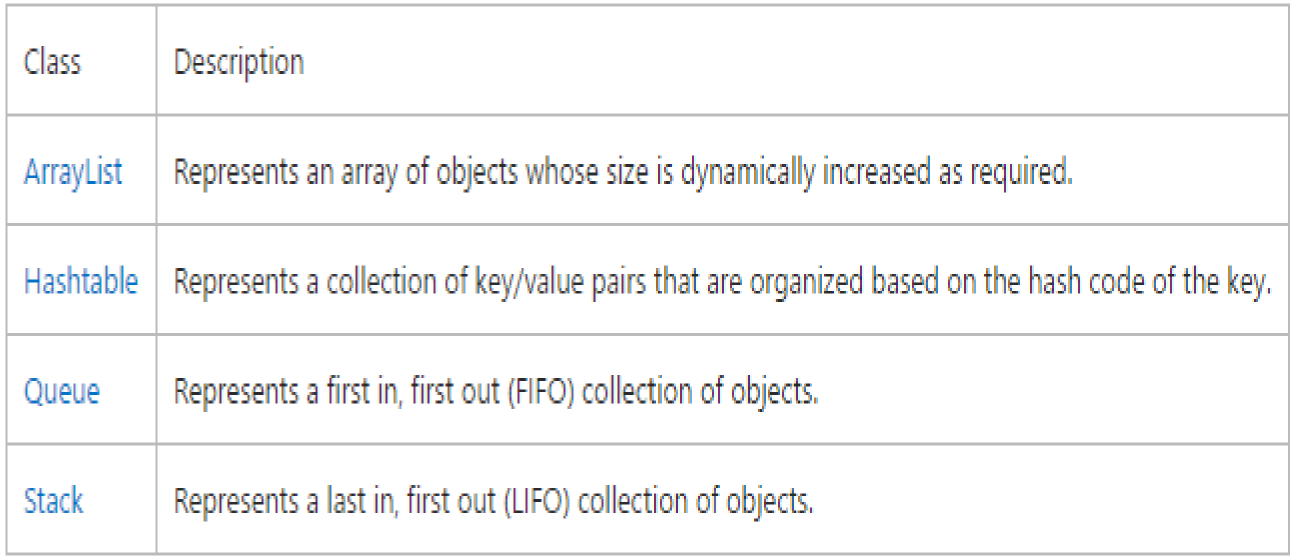
You can create a generic collection by using one of the classes in the System Collections Generic namespace. A generic collection is useful when every item in the collection has the same data type.



Class	Description
Dictionary<TKey, TValue>	Represents a collection of key/value pairs that are organized based on the key.
List<T>	Represents a list of objects that can be accessed by index. Provides methods to search, sort, and modify lists.
Queue<T>	Represents a first in, first out (FIFO) collection of objects.
SortedList<TKey, TValue>	Represents a collection of key/value pairs that are sorted by key based on the associated IComparer<T> implementation.
Stack<T>	Represents a last in, first out (LIFO) collection of objects.

System.Collections Classes

The classes in the System.Collections namespace do not store elements as specifically typed, but as objects of type **Object**.



Class	Description
ArrayList	Represents an array of objects whose size is dynamically increased as required.
Hashtable	Represents a collection of key/value pairs that are organized based on the hash code of the key.
Queue	Represents a first in, first out (FIFO) collection of objects.
Stack	Represents a last in, first out (LIFO) collection of objects.

Collections

Running Demos