Delegates And Events 2

```
/*
Learning C#
by Jesse Liberty

Publisher: O'Reilly
ISBN: 0596003765
*/
using System;

namespace DelegatesAndEvents
{
public enum comparison
{
theFirstComesFirst = 1,
theSecondComesFirst = 2
}

// A simple collection to hold two items.
class Pair
{
// Private array to hold the two objects.
private object[] thePair = new object[2];

// The delegate declaration.
public delegate comparison
WhichIsFirst(object obj1, object obj2);

// Passed in constructor takes two objects,
// added in order received.
public Pair(
object firstObject,
object secondObject)
{
thePair[0] = firstObject;
thePair[1] = secondObject;
```

```
}

// Public method that orders
// the two objects by whatever criteria the objects like!
public void Sort(
WhichIsFirst theDelegatedFunc)
{
if (theDelegatedFunc(thePair[0],thePair[1])
== comparison.theSecondComesFirst)
{
object temp = thePair[0];
thePair[0] = thePair[1];
thePair[1] = temp;
}
}

// Public method that orders
// the two objects by the reverse of whatever criteria
// the objects like!
public void ReverseSort(
WhichIsFirst theDelegatedFunc)
{
if (theDelegatedFunc(thePair[0],thePair[1]) ==
comparison.theFirstComesFirst)
{
object temp = thePair[0];
thePair[0] = thePair[1];
thePair[1] = temp;
}
}

// Ask the two objects to give their string value.
public override string ToString()
{
return thePair[0].ToString() + “, ”
+ thePair[1].ToString();
}
}
```

```
class Dog
{
private int weight;

// A static delegate.
public static readonly Pair.WhichIsFirst OrderDogs =
new Pair.WhichIsFirst(Dog. WhichDogComesFirst);

public Dog(int weight)
{
this.weight=weight;
}

// Dogs are sorted by weight.
public static comparison WhichDogComesFirst(
Object o1, Object o2)
{
Dog d1 = (Dog) o1;
Dog d2 = (Dog) o2;
return d1.weight > d2.weight ?
comparison.theSecondComesFirst :
comparison.theFirstComesFirst;
}
public override string ToString()
{
return weight.ToString();
}
}

class Student
{
private string name;

// A static delegate.
public static readonly Pair.WhichIsFirst OrderStudents =
new Pair.WhichIsFirst(Student.WhichStudentComesFirst);

public Student(string name)
{
```

```
this.name = name;
}

// Students are sorted alphabetically.
public static comparison
WhichStudentComesFirst(Object o1, Object o2)
{
Student s1 = (Student) o1;
Student s2 = (Student) o2;
return (String.Compare(s1.name, s2.name) < 0 ? comparison.theFirstComesFirst : comparison.theSecondComesFirst); } public override string ToString() { return name; } } public class TesterDelegatesAndEvents1 { public void Run() { // Create two students and two dogs // and add them to Pair objects. Student Jesse = new Student("Jesse"); Student Stacey = new Student ("Stacey"); Dog Milo = new Dog(65); Dog Fred = new Dog(12); // Create the Pair object. Pair studentPair = new Pair(Jesse,Stacey); Pair dogPair = new Pair(Milo, Fred); Console.WriteLine("studentPair : {0}", studentPair.ToString()); Console.WriteLine("dogPair : {0}", dogPair.ToString()); // Tell the student Pair to sort itself, // passing in the Student delegate. studentPair.Sort(Student.OrderStudents); Console.WriteLine("After Sort studentPair : {0}", studentPair.ToString()); studentPair.ReverseSort(Student.OrderStudents); Console.WriteLine("After ReverseSort studentPair : {0}", studentPair.ToString()); // Tell the Dog pair to sort itself, // passing in the Dog delegate. dogPair.Sort(Dog.OrderDogs); Console.WriteLine("After Sort dogPair : {0}", dogPair.ToString()); dogPair.ReverseSort(Dog.OrderDogs); Console.WriteLine("After ReverseSort dogPair : {0}", dogPair.ToString()); } [STAThread] static void Main() { TesterDelegatesAndEvents1 t = new TesterDelegatesAndEvents1(); t.Run(); } } } [/csharp]
```