/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*

\* Interfaces: Part 2

\*

\* Core Topics:

\* 1. Implement implicitly multiple interfaces.

\* 2. Two ways for executing an interface method:

\* a. Using object reference.

\* b. Using interface reference.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

usingSystem**;**

usingSystem.Collections.Generic**;**

usingSystem.Text**;**

namespaceInterfaceDemo

**{**

**interface ISortable**

**{**

**void Sort();**

**}**

interfaceISearchable

**{**

intIndexOf**(**objecto**);**

boolFound**(**objecto**);**

**}**

internalclassCollection **:** ISearchable**, ISortable**

**{**

privateint **[]** \_internalList**;**

publicCollection**(**intsize**)**

**{**

\_internalList=newint**[**size**];**

GenerateNumbers**();**

**}**

publicintIndexOf**(**objecto**)**

**{**

intindex=-1**;**

inttargetData=0**;**

try

**{**

targetData= **(**int**)**o**;**

for **(**inti=0**;** i<\_internalList.Length**;** i++**)**

**{**

if **(**\_internalList**[**i**]** ==targetData**)**

**{**

index=i**;**

break**;**

**}**

**}**

**}**

catch **(**Exceptione**)**

**{**

index=-1**;**

Console.WriteLine**(**"\nEXCEPTION: {0}"**,** e.Message**);**

Console.WriteLine**(**" {1}"**,** e.StackTrace**);**

**}**

returnindex**;**

**}**

publicboolFound**(**objecto**)**

**{**

boolfound=false**;**

intindex=0**;**

inttargetData=0**;**

try

**{**

targetData= **(**int**)**o**;**

while **(**!found&&index<\_internalList.Length**)**

**{**

if **(**\_internalList**[**index**]** ==targetData**)**

**{**

found=true**;**

**}**

else

**{**

index++**;**

**}**

**}**

**}**

catch **(**Exceptione**)**

**{**

index=-1**;**

Console.WriteLine**(**"\nEXCEPTION: {0}"**,** e.Message**);**

Console.WriteLine**(**" {1}"**,** e.StackTrace**);**

**}**

returnfound**;**

**}**

**public void Sort ()**

**{**

// Because \_internalList is an array, the Array type has a

// sort method, and the Array type knows how to sort ints,

// we can simply rely on the Array.Sort() method to do the

// sorting for us. If we were using our own custom type

// rather than int, we would have to implement IComparable

// to implement how to compare two objects.

Array.Sort**(**\_internalList**);**

**}**

publicoverridestringToString**()**

**{**

StringBuildersb=newStringBuilder**();**

for **(**inti=0**;** i<\_internalList.Length**;** i++**)**

**{**

sb.AppendFormat**(**"{0} "**,** \_internalList**[**i**]);**

**}**

returnsb.ToString**();**

**}**

privatevoidGenerateNumbers**()**

**{**

Randomr=newRandom**(**DateTime.Now.Millisecond**);**

for **(**inti=0**;** i<\_internalList.Length**;** i++**)**

**{**

\_internalList**[**i**]** =r.Next**(**1**,** 10000**);**

System.Threading.Thread.Sleep**(**71**);**

**}**

**}**

**}**

classProgram

**{**

privatestaticintGetTargetNumber**()**

**{**

inttargetNumber=-1**;**

Console.Write**(**"\nWhat number are you looking for: "**);**

stringuserInput=Console.ReadLine**();**

if **(**!int.TryParse**(**userInput**,** outtargetNumber**))**

**{**

targetNumber=-1**;**

**}**

returntargetNumber**;**

**}**

staticvoidMain**(**string**[]** args**)**

**{**

boolkeepGoing=true**;**

intlistSize=0**;**

CollectionlistOfNumbers=null**;**

Console.Clear**();**

Console.Write**(**"How many numbers do you want to generate: "**);**

stringuserInput=Console.ReadLine**();**

if **(**!int.TryParse**(**userInput**,** outlistSize**))**

**{**

listSize=0**;**

**}**

if **(**listSize<=0**)**

**{**

keepGoing=false**;**

**}**

else

**{**

listOfNumbers=newCollection**(**listSize**);**

**}**

while **(**keepGoing**)**

**{**

Console.WriteLine**(**"\nChoose from the following options:\n"**);**

Console.WriteLine

**(**"\t1. Get the index of an item in the list."**);**

Console.WriteLine**(**"\t2. See if an item is in the list."**);**

Console.WriteLine**(**"\t3. Display the entire list."**);**

Console.WriteLine**(**"\t4. Sort the entire list."**);**

Console.WriteLine**(**"\tE. Exit."**);**

Console.Write**(**"\nYour choice: "**);**

stringuserOption=Console.ReadLine**();**

userOption=userOption.ToUpper**();**

inttargetNumber=-1**;**

switch **(**userOption**)**

**{**

case"1"**:**

targetNumber=GetTargetNumber**();**

if **(**targetNumber>0**)**

**{**

intindex=listOfNumbers.IndexOf**(**targetNumber**);**

if **(**index>=0**)**

**{**

Console.WriteLine

**(**"\n{0} is found at index {1}.\n"**,**

targetNumber**,** index**);**

**}**

else

**{**

Console.WriteLine

**(**"\n{0} is not in the list.\n"**,**

targetNumber**);**

**}**

**}**

break**;**

case"2"**:**

targetNumber=GetTargetNumber**();**

if **(**targetNumber>0**)**

**{**

if **(**listOfNumbers.Found**(**targetNumber**))**

**{**

Console.WriteLine**(**"\n{0} is in the list.\n"**,**

targetNumber**);**

**}**

else

**{**

Console.WriteLine

**(**"\n{0} is NOT in the list.\n"**,**

targetNumber**);**

**}**

**}**

break**;**

case"3"**:**

Console.Write**(**"\nThe list: {0}\n"**,**

listOfNumbers.ToString**());**

break**;**

case"4"**:**

// You can access the implicit implementation of an interface

// method through the reference to the object.

**listOfNumbers.Sort();**

// OR

// You can access the implicit implementation of an interface

// method by implicitly converting the object reference to

// an interface reference and use the interface reference.

// You will see how this will becomeadvantageous later on

// using polymorphism.

**//ISortable isort = listOfNumbers;**

**//isort.Sort();**

Console.WriteLine**(**"\nThe list has been sorted.\n"**);**

break**;**

case"E"**:**

keepGoing=false**;**

break**;**

default**:**

Console.WriteLine

**(**"\nYour option {0} is not valid. Try again!\n"**,**

userOption**);**

break**;**

**}**

**}**

Console.Write**(**"\n\nPress <ENTER> to end: "**);**

Console.ReadLine**();**

**}**

**}**

**}**