***Группа ИТ-2707. Леженко С.А.***

1. Создать строго типизированную коллекцию, не принимающую значения никаких других типов, кроме исходного (наваше усмотрение).

***IntList.cs***

using System;

using System.Collections;

namespace DZ\_6

{

public class IntList : ICollection

{

private int[] contents = new int[10];

private int count;

public IntList()

{ count = 0; }

public int Count

{ get { return count; } }

public int Length

{ get { return contents.Length; } }

private void UpSize()

{

int[] tmp = new int[contents.Length \* 2];

Array.Copy(contents, tmp, contents.Length);

contents = tmp;

}

#region IList members

public int Add(int value)

{

if (count == contents.Length)

UpSize();

contents[count] = value;

count++;

return (count - 1);

}

public void Clear()

{ count = 0; }

public bool Contains(int value)

{

bool inList = false;

for (int i = 0; i < Count; i++)

{

if (contents[i] == value)

{

inList = true;

break;

}

}

return inList;

}

public int IndexOf(int value)

{

int itemIndex = -1;

for (int i = 0; i < Count; i++)

{

if (contents[i] == value)

{

itemIndex = i;

break;

}

}

return itemIndex;

}

public void Insert(int index, int value)

{

if ((index < Count) && (index >= 0))

{

if (count == contents.Length)

UpSize();

count++;

for (int i = Count - 1; i > index; i--)

contents[i] = contents[i - 1];

contents[index] = value;

}

}

public void Remove(int value)

{ RemoveAt(IndexOf(value)); }

public void RemoveAt(int index)

{

if ((index >= 0) && (index < Count))

{

for (int i = index; i < Count - 1; i++)

contents[i] = contents[i + 1];

count--;

}

}

public int this[int index]

{

get { return contents[index]; }

set { contents[index] = value; }

}

public void CopyTo(Array array, int index)

{

int j = index;

for (int i = 0; i < Count; i++)

array.SetValue(contents[i], j++);

}

public object SyncRoot { get { return this; } }

#endregion

IEnumerator IEnumerable.GetEnumerator()

{ return GetEnumerator(); }

public IntListEnum GetEnumerator()

{

int[] tmp = new int[Count];

Array.Copy(contents, tmp, Count);

return new IntListEnum(tmp);

}

public bool IsSynchronized

{ get { return false; } }

}

}

***IntListEnum.cs***

using System;

using System.Collections;

namespace DZ\_6

{

public class IntListEnum : IEnumerator

{

public int[] contents;

int pos = -1;

public IntListEnum(int[] contents)

{

this.contents = contents;

}

object IEnumerator.Current

{ get { return Current; } }

public int Current

{

get

{

try

{

return contents[pos];

}

catch (IndexOutOfRangeException)

{

throw new InvalidOperationException();

}

}

}

public bool MoveNext()

{

++pos;

return (pos < contents.Length);

}

public void Reset()

{

pos = -1;

}

}

}

***Program.cs***

using System;

namespace DZ\_6

{

public class Program

{

public static void Main(string[] args)

{

IntList il = new IntList();

Console.WriteLine("il.Length = {0}", il.Length);

Console.WriteLine("il.Count = {0}", il.Count);

for (int i = 0; i < 15; ++i)

il.Add(i \* 3);

foreach (int i in il)

Console.Write("{0} ", i);

Console.WriteLine("\nil.Contains(9) = {0}", il.Contains(9));

Console.WriteLine("il.IndexOf(9) = {0}", il.IndexOf(9));

il.RemoveAt(3);

Console.WriteLine("il.RemoveAt(3);");

foreach (int i in il)

Console.Write("{0} ", i);

il.Remove(12);

Console.WriteLine("\nil.Remove(12);");

foreach (int i in il)

Console.Write("{0} ", i);

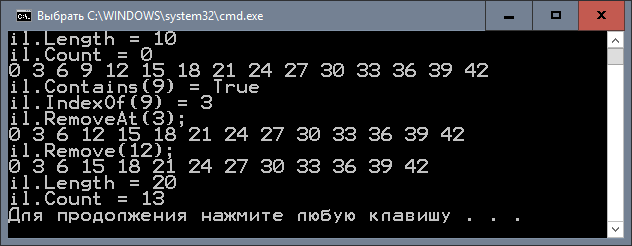
Console.WriteLine("\nil.Length = {0}", il.Length);

Console.WriteLine("il.Count = {0}", il.Count);

}

}

}



2. Разработать собственный класс, имитирующий работу стека.

***MyStack.cs***

using System;

using System.Collections;

namespace DZ\_6\_2

{

class MyStack : ICollection, IEnumerable

{

object[] contents;

int count = 0;

public int Count

{

get { return count; }

}

public MyStack(int count= 10)

{

if (count <= 0) count = 10;

contents = new object[count];

}

public void Push(object value)

{

if (count < contents.Length)

contents[count++] = value;

else

throw new StackOverflowException();

}

public object Peek()

{

if (count > 0)

return contents[count - 1];

else

throw new InvalidOperationException();

}

public void Pop()

{

if (count > 0)

--count;

else

throw new InvalidOperationException();

}

public int Length

{

get { return contents.Length; }

}

public bool IsSynchronized

{

get { return false; }

}

public object SyncRoot

{

get { return this; }

}

public void CopyTo(Array array, int index)

{

int j = index;

for (int i = 0; i < Count; i++)

array.SetValue(contents[i], j++);

}

IEnumerator IEnumerable.GetEnumerator()

{

return new MyStackEnumerator(contents, count);

}

}

}

***MyStackEnumerator.cs***

using System;

using System.Collections;

namespace DZ\_6\_2

{

class MyStackEnumerator : IEnumerator

{

object[] contents;

int pos = -1;

int last;

public MyStackEnumerator(object[] contents, int last)

{

this.contents = contents;

if (last > 0)

this.last = last;

else

this.last = contents.Length;

}

object IEnumerator.Current

{

get

{

try

{

return contents[pos];

}

catch (IndexOutOfRangeException)

{

throw new InvalidOperationException();

}

}

}

bool IEnumerator.MoveNext()

{

++pos;

return (pos < last);

}

void IEnumerator.Reset()

{

pos = -1;

}

}

}

***Program.cs***

using System;

namespace DZ\_6\_2

{

class Program

{

static void Main(string[] args)

{

MyStack St = new MyStack(15);

Console.WriteLine("St.Length = {0}", St.Length);

Console.WriteLine("St.Count = {0}", St.Count);

for (int i = 0; i < 15; ++i)

St.Push(i);

foreach (int i in St)

Console.Write("{0} ", i);

Console.WriteLine("\nSt.Peek() = {0}", St.Peek());

Console.WriteLine("St.Pop()");

St.Pop();

foreach (int i in St)

Console.Write("{0} ", i);

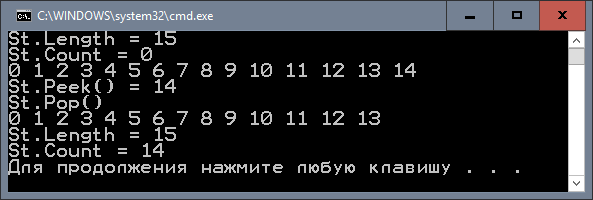
Console.WriteLine("\nSt.Length = {0}", St.Length);

Console.WriteLine("St.Count = {0}", St.Count);

}

}

}



3. Создать примитивный англо-русский и русско-английский словарь, содержащий пары слов –названий стран на русском и английском языках. Пользователь должен иметь возможность выбирать направление перевода и запрашивать перевод.

***CountryDictionary.cs***

using System;

using System.IO;

using System.Collections.Generic;

namespace DZ\_6\_3

{

/// <summary>

/// Description of CountryDictionary.

/// </summary>

public class CountryDictionary

{

const string path = @".\dict.txt";

SortedList<string, string> ruToEnDict = new SortedList<string, string>(200);

SortedList<string, string> enToRuDict = new SortedList<string, string>(200);

public CountryDictionary()

{

using (var sr = new StreamReader(path, System.Text.Encoding.Default))

{

string line;

var buf = new string[2];

while ((line = sr.ReadLine()) != null)

{

buf = line.Split('\t');

try

{

ruToEnDict.Add(buf[0], buf[1]);

enToRuDict.Add(buf[1], buf[0]);

}

catch (ArgumentException)

{

Console.WriteLine("An element {0} - {1} already exists.", buf[0], buf[1]);

}

}

}

}

public string TranslateRuToEn(string ru)

{

string en;

try

{

en = ruToEnDict[ru];

}

catch (KeyNotFoundException)

{

en = ru + " is not found.";

}

return en;

}

public string TranslateEnToRu(string en)

{

string ru;

try

{

ru = enToRuDict[en];

}

catch (KeyNotFoundException)

{

ru = en + " is not found.";

}

return ru;

}

}

}

***Program.cs***

using System;

namespace DZ\_6\_3

{

class Program

{

private delegate string transDelegate(string country);

public static void Main(string[] args)

{

CountryDictionary cd;

try

{

cd = new CountryDictionary();

}

catch (System.IO.FileNotFoundException)

{

Console.WriteLine("Error: Not found dict.txt");

return;

}

ConsoleKeyInfo key;

transDelegate trans;

Console.WriteLine("Translator of the names of the countries of the world");

do

{

Console.WriteLine("\n1 - Translate from Russian to English");

Console.WriteLine("2 - Translate from English to Russian");

Console.WriteLine("0 - Exit");

Console.Write("Select translation direction: ");

key = Console.ReadKey();

switch (key.KeyChar)

{

case '1':

{

trans = cd.TranslateRuToEn;

break;

}

case '2':

{

trans = cd.TranslateEnToRu;

break;

}

case '0':

{

continue;

}

default:

{

Console.WriteLine("\nInvalid choice. Try again");

continue;

}

}

do

{

Console.Write("\nEnter name of country to translate: ");

Console.WriteLine("Translate: " + trans(Console.ReadLine()));

Console.Write("next country? (y/n): ");

} while (Console.ReadKey().KeyChar == 'y');

trans = null;

} while (key.KeyChar != '0');

Console.Write("\nPress any key to continue . . . ");

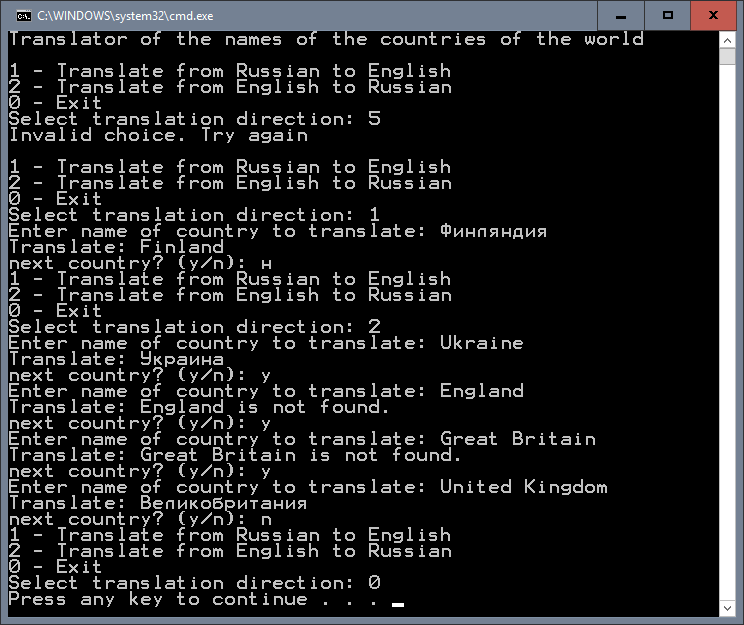
Console.ReadKey(true);

}

}

}

***dict.txt содержит пары названий на русском и английском языках***



4. Создать не обобщенный класс точки в 3-хмерном пространстве с целочисленными координатами (Point3D), который наследуется от generic класса Point2D<T>, рассмотренного в уроке. В классе предусмотреть поле для хранения значения координаты z. Реализовать в классе:   
a. конструктор с параметрами, который принимает начальные значения для координат точки  
b. метод ToString().

***Point2D.cs***

using System;

namespace DZ\_6\_4

{

/// <summary>

/// Description of Point2D.

/// </summary>

/// <summary>

/// Обобщенный класс точки

/// </summary>

/// <typeparam name="T">

/// координаты точки могут быть любого типа

/// </typeparam>

public class Point2D<T> where T: struct

{

//параметр типа используется для задания типа переменных класса

T x;

T y;

//параметр типа используется для задания типа свойста

public T X

{

get { return x; }

set { x = value; }

}

public T Y

{

get { return y; }

set { y = value; }

}

//параметр типа используется для задания типов параметров метода

public Point2D(T x, T y)

{

this.x = x; this.y = y;

}

public Point2D()

{

this.x = default(T); this.y = default(T);

}

}

}

***Point3D.cs***

using System;

namespace DZ\_6\_4

{

/// <summary>

/// Description of Point3D.

/// </summary>

public class Point3D : Point2D<double>

{

private object z;

public object Z

{

get { return z; }

set { z = value; }

}

public Point3D() : base()

{

Z = default(object);

}

public Point3D(object X, object Y, object Z) : base()

{

try

{

base.X = (double)(int)X;

}

catch (InvalidCastException)

{

try

{

base.X = (double)(long)X;

}

catch (InvalidCastException)

{

try

{

base.X = (double)(float)X;

}

catch (InvalidCastException)

{

base.X = (double)X;

}

}

}

try

{

base.Y = (double)(int)Y;

}

catch (InvalidCastException)

{

try

{

base.Y = (double)(long)Y;

}

catch (InvalidCastException)

{

try

{

base.Y = (double)(float)Y;

}

catch (InvalidCastException)

{

base.Y = (double)Y;

}

}

}

this.Z = Z;

}

public override string ToString()

{

return string.Format("[Point3D X={0}, Y={1}, Z={2}]", X, Y, Z);

}

}

}

***Program.cs***

using System;

namespace DZ\_6\_4

{

class Program

{

public static void Main(string[] args)

{

var aPoint = new Point3D((long)15, (float)18.6, -8);

Console.WriteLine(aPoint);

Console.Write("Press any key to continue . . . ");

Console.ReadKey(true);

}

}

}



5. Создать обобщенный класс прямой на плоскости. В классе предусмотреть 2 поля типа обобщенной точки – точки, через которые проходит прямая. Реализовать в классе:   
a. конструктор, который принимает 2 точки  
b. конструктор, который принимает 4 координаты (x и у координаты для 1-ой и 2-ой точки)  
c. метод ToString()

***Line2D.cs***

namespace DZ\_6\_5

{

public class Line2D<T>

{

public class Point

{

private T x;

private T y;

public T X { get { return x; } set { x = value; } }

public T Y { get { return y; } set { y = value; } }

public Point(T x, T Y)

{

this.X = X;

this.Y = Y;

}

}

private Point p1;

private Point p2;

public Point P1 { get { return p1; } set { p1 = value; } }

public Point P2 { get { return p2; } set { p2 = value; } }

public Line2D(Point P1, Point P2)

{

this.P1 = P1;

this.P2 = P2;

}

public Line2D(T X1, T Y1, T X2, T Y2)

{

P1 = new Point(X1, Y1);

P2 = new Point(X2, Y2);

}

public override string ToString()

{

return string.Format("[Line2D: P1({0}, {1}), P2({2}, {3})]", P1.X, P1.Y, P2.X, P2.Y);

}

}

}

***Program.cs***

using System;

namespace DZ\_6\_5

{

class Program

{

static void Main(string[] args)

{

var Line1 = new Line2D<int>(15, -8, 18, 34);

var Line2 = new Line2D<double>(

new Line2D<double>.Point(0.6, -13.4),

new Line2D<double>.Point(7, -2.3)

);

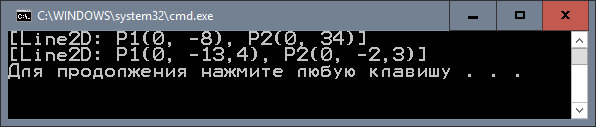
Console.WriteLine(Line1);

Console.WriteLine(Line2);

}

}

}



6. Подсчитать, сколько раз каждое слово встречается в заданном тексте. Результат записать в коллекцию Dictionary <TKey, TValue>.