Answers To Exercises In Getting Started With .NET 4.5 and C# 5.0

1. Accept inches from user and display the equivalent centimeters

using System;

namespace st

{

class InchToCm

{

public static void Main()

{

Console.Write("Enter Inches :");

string input = Console.ReadLine();

int inches = Int32.Parse(input);

Console.WriteLine("There are {0} centimeters for {1} inches",

inches \* 2.5, inches);

} // main

} // class

} // namespace

2. Write a program to take a number and display highest factor for that number

using System;

namespace st

{

class HighestFactor

{

static void Main()

{

Console.Write("Enter a number :");

string input = Console.ReadLine();

int num = Int32.Parse(input);

for (int i = num / 2; i > 0; i--)

{

if (num % i == 0)

{

Console.WriteLine("Highest factor for {0} is {1}", num, i);

break;

}

} // for

} // main

} // class

} // namespace

3. Write a program to take a series of numbers from user and display the highest of all the numbers

using System;

namespace st

{

class Highest

{

static void Main()

{

int highest = 0;

while (true)

{

Console.Write("Enter a number :");

string input = Console.ReadLine();

int num = Int32.Parse(input);

// stop if user's input is 0

if (num == 0)

break;

if (num > highest)

highest = num;

}

Console.WriteLine("Highest of all numbers is : {0}", highest);

} // main

} // class

} // st

4. Create a function that takes an array and returns the first two highest numbers

using System;

namespace st

{

class TwoHighest

{

public static void Main()

{

int fh, sh;

int[] a = { 10, 30, 40, 22, 55 };

FirstTwoHighest(a, out fh, out sh);

Console.WriteLine("First Highest {0}, Second Highest : {1}", fh, sh);

}

public static void FirstTwoHighest(int[] a, out int fh, out int sh)

{

fh = sh = 0;

foreach (int n in a)

{

if (n > fh)

{

sh = fh;

fh = n;

}

else

if ( n > sh )

sh = n;

} // foreach

} // FirstTwoHighest

} // class

} // namespace

5. Create a function that takes two numbers and returns the highest common factor

using System;

namespace st

{

class HCF

{

public static void Main()

{

Console.WriteLine(Hcf(30, 55));

Console.WriteLine(Hcf(28, 27));

}

public static int Hcf(int n1, int n2)

{

// start the loop from half of the smallest of two numbers

for (int i = n1 < n2 ? n1 / 2 : n2 / 2; i > 1; i--)

{

if (n1 % i == 0 && n2 % i == 0)

return i; // found Highest common factor

} // for

return 1; // Didn't find highest common factor, so 1 is treated as so

} // HFC

} // class

} // namespace

6. Declare an array of 10 elements. Read values from user and insert them into array. Display how many values in the array are positive, negative and zeros

using System;

namespace st

{

public class ArrayCount

{

public static void Main()

{

int[] a = new int[10];

for (int i = 0; i < a.Length; i++)

{

Console.Write("Enter a number :");

a[i] = Int32.Parse(Console.ReadLine());

}

int nop, non, noz;

nop = non = noz = 0;

foreach (int n in a)

{

if (n > 0)

nop++;

else

if (n < 0)

non++;

else

noz++;

}

Console.WriteLine("No. of positives : {0}", nop);

Console.WriteLine("No. of negatives : {0}", non);

Console.WriteLine("No. of zeroes : {0}", noz);

} // main

} // class

} // namespace

7. Accept a number on the command line and display whether it is a strong number

using System;

namespace st

{

class StrongNumber

{

public static void Main(string[] args)

{

int num = Int32.Parse(args[0]);

int orgnum = num;

int total = 0;

while (num != 0)

{

// take a digit

int digit = num % 10;

int factorial = 1;

for (int i = 1; i <= digit; i++)

factorial \*= i;

total += factorial;

// remove digit

num /= 10;

}

if (total == orgnum)

Console.WriteLine("{0} is a strong number", orgnum);

else

Console.WriteLine("{0} is not a strong number", orgnum);

} // main

} // class

} // st

8. Create a function that takes a number by value and returns smallest and largest factors of the number through pass by reference.

using System;

namespace st

{

class HighestLowestFactors

{

public static void Main()

{

int highfact, lowfact;

Console.WriteLine("Enter a number : ");

int num = Int32.Parse(Console.ReadLine());

GetHighestLowestFactors( num, out highfact, out lowfact);

Console.WriteLine(

"For number {0}, Highest Factor is {1} and Lowest Factor is : {2}",

num, highfact,lowfact );

}

public static void GetHighestLowestFactors(int num, out int hf, out int lf)

{

// get lowest factor other than 1. In case of prime number 1

lf = 1;

for (int i = 2; i <= num / 2; i++)

{

if (num % i == 0)

{

lf = i;

break;

}

}

// get highest factor other than the number. In case of prime number it is number itself

hf = num;

for (int i = num/2 ; i > 1; i--)

{

if (num % i == 0)

{

hf = i;

break;

}

} // for

} // GetHighestLowestFactors

} // class

} // namespace

Answers to exercises in OOP with C# And .NET Library (Part 1)

1. Create a class to store details of student like rollno, name, course joined and fee paid so far. Assume courses are C# and ASP.NET with course fees being 2000 and 3000.

Provide the a constructor to take rollno, name and course.

Provide the following methods:

Payment(amount)

Print()

DueAmount property

TotalFee property

using System;

namespace st

{

class Student

{

private int rollno;

private string name;

private string course;

private int feepaid;

public Student(int rollno, string name, string course)

{

this.rollno = rollno;

this.name = name;

this.course = course;

}

public void Payment(int amount)

{

feepaid += amount;

}

public void Print()

{

Console.WriteLine(rollno);

Console.WriteLine(name);

Console.WriteLine(course);

Console.WriteLine(feepaid);

}

public int DueAmount

{

get

{

return TotalFee - feepaid;

}

}

public int TotalFee

{

get

{

return course == "c#" ? 2000 : 3000;

}

}

}

class UseStudent

{

public static void Main()

{

Student s = new Student(1, "ABC", "c#");

s.Payment(1000);

s.Print();

Console.WriteLine(s.DueAmount);

}

}

}

Add a static member to store Service Tax, which is set to 12.3%. Also allow a property through which we can set and get service tax.

Modify TotalFee and DueAmount properties to consider service tax.

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

namespace oop1

{

class Student2

{

private int rollno;

private string name;

private string course;

private int feepaid;

private static double servicetax = 12.3;

public Student2(int rollno, string name, string course)

{

this.rollno = rollno;

this.name = name;

this.course = course;

}

public void Payment(int amount)

{

feepaid += amount;

}

public void Print()

{

Console.WriteLine(rollno);

Console.WriteLine(name);

Console.WriteLine(course);

Console.WriteLine(feepaid);

}

public int DueAmount

{

get

{

return TotalFee - feepaid;

}

}

public int TotalFee

{

get

{

double total = course == "c#" ? 2000 : 3000;

// service tax

total = total + total \* servicetax / 100;

return (int) total;

}

}

public static double ServiceTax

{

get

{

return servicetax;

}

set

{

servicetax = value;

}

}

} // Student2

class UseStudent2

{

public static void Main()

{

Student2 s = new Student2(1, "ABC", "asp.net");

s.Payment(1000);

s.Print();

Console.WriteLine(s.DueAmount);

}

}

}

2.Create the classes required to store data regarding different types of Courses. All courses have name, duration and course fee. Some courses are part time where you have to store the timing for course. Some courses are onsite where you have to store the company name and the no. of candidates for the course. For onsite course we charge 10% more on the course fee. For part-time course, we offer 10% discount.

Provide constructors and the following methods.

Print()

GetTotalFee()

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

namespace st

{

abstract class Course

{

protected string name;

protected int duration;

protected int coursefee;

public Course(string name, int duration, int coursefee)

{

this.name = name;

this.duration = duration;

this.coursefee = coursefee;

}

public virtual void Print()

{

Console.WriteLine(name);

Console.WriteLine(duration);

Console.WriteLine(coursefee);

}

public abstract int GetTotalFee();

}

class ParttimeCourse : Course

{

private string timings;

public ParttimeCourse(string name, int duration, int coursefee, string timings) : base(name,duration,coursefee)

{

this.timings = timings;

}

public override void Print()

{

base.Print();

Console.WriteLine(timings);

}

public override int GetTotalFee()

{

return (int) (coursefee \* 0.90); // 10% discount

}

}

class OnsiteCourse : Course

{

private string company;

private int nostud;

public OnsiteCourse(string name, int duration, int coursefee, string company, int nostud)

: base(name, duration, coursefee)

{

this.company = company;

this.nostud = nostud;

}

public override void Print()

{

base.Print();

Console.WriteLine(company);

Console.WriteLine(nostud);

}

public override int GetTotalFee()

{

return (int)(coursefee \* 1.1); // 10% more

}

}

class TestCourse

{

public static void Main()

{

Course c = new OnsiteCourse("ASP.NET", 30, 5000, "ABC Tech", 10);

c.Print();

Console.WriteLine(c.GetTotalFee());

c = new ParttimeCourse("C#", 30, 3000, "7-8pm");

c.Print();

Console.WriteLine(c.GetTotalFee());

}

}

}

3. Create an interface called Stack with methods Push(), Pop() and property Length. Create a class that implements this interface. Use an array to implement stack data structure.

Create user-defined exceptions and ensure Push() and Pop() methods throw those exceptions when required.

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

namespace oop1

{

interface IStack

{

void Push(int v);

int Pop();

int Length { get; }

}

class StackFullException : Exception

{

public StackFullException()

: base("Stack Full")

{

}

}

class StackEmptyException : Exception

{

public StackEmptyException()

: base("Stack Empty")

{

}

}

class Stack : IStack

{

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

private int top = 0;

public void Push(int v)

{

if (top == 10)

throw new StackFullException();

a[top] = v;

top++;

}

public int Pop()

{

if (top == 0)

throw new StackEmptyException();

top--;

return a[top];

}

public int Length

{

get

{

return top;

}

}

}

class UseStack

{

public static void Main()

{

Stack s = new Stack();

s.Push(20);

Console.WriteLine(s.Pop());

Console.WriteLine(s.Length);

Console.WriteLine(s.Pop());

}

}

}

4.Answer the following.

Can a static method access instance variable? [True/False]

False

A partial method must be in a \_\_\_\_\_\_\_ class.

Partial class

You cannot overload two methods that return different return types when names and parameters are same. [True/False]

True

In order to make property read-only, we need to use read-only keyword.[True/False]

False. We need to omit Set method.

A class can have more than one indexer with the same type of parameter.[True/False]

False

A class marked as sealed can have an abstract method. [True/False]

False

When you create a method in the derived class with the same name as a method in base class, by default, it is said to [override/shadow] the method in the base class.

Shadow

If a function returns control from a try block for which we have finally block then finally block is not executed and control returns from the function to caller. [True/False]

False. Finally block is executed then control returns to caller

5. Create a form that takes loan amount and period, which is either 1 year or 2 years or 3 years.

The interest rates are as follows:

1 year - 10%

2 years - 12%

3 years - 15%

Calculate the total amount to be paid (amount + interest) and find out EMI (Every month installment) and display it to user.

using System;

using System.Drawing;

using System.Windows.Forms;

namespace st

{

public partial class Form1 : Form

{

public Form1()

{

InitializeComponent();

}

private void btnCalculateEMI\_Click(object sender, EventArgs e)

{

try

{

double amount = Double.Parse(txtLoanAmount.Text);

int period = cmbPeriod.SelectedIndex + 1;

double emi = 0;

switch (period)

{

case 1: emi = (amount + (amount \* 10 / 100)) / 12; break;

case 2: emi = (amount + (amount \* 12 / 100)) / 24; break;

case 3: emi = (amount + (amount \* 15 / 100)) / 36; break;

default :

MessageBox.Show("Please select period for your instalment!", "Error");

return;

}

MessageBox.Show( String.Format ("Monthly Installment : {0}", emi), "EMI");

}

catch (Exception ex)

{

MessageBox.Show("Invalid Input. Please enter a valid amount for loan amount!", "Error");

}

}

} // class

} // namespace

6. Accept a string from user through keyboard and display it vertically.

using System;

namespace st

{

class StringVertical {

public static void Main() {

Console.Write("Enter a string : ");

string st = Console.ReadLine();

for ( int i = 0 ; i < st.Length; i ++)

{

Console.WriteLine ( st[i]);

}

} // main

} // class

} // namespace

7. Accept 10 strings from user and display the highest of all strings.

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

namespace st

{

class LargestString

{

public static void Main()

{

string largest = "";

for (int i = 1; i <= 10; i++)

{

Console.Write("Enter a string : ");

string st = Console.ReadLine();

if (st.CompareTo(largest) > 0)

largest = st;

} // for

Console.WriteLine("Largest of all strings : {0}", largest);

} // main

} // class

} // namespace

8. Create a class to store day, month and year. Ensure objects of this class are comparable.

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

namespace st

{

class Date : IComparable

{

private int day, month, year;

public Date(int day, int month, int year)

{

this.day = day;

this.month = month;

this.year = year;

}

public int CompareTo(Date other)

{

if ( this.year - other.year != 0 )

return this.year - other.year;

if (this.month - other.month != 0)

return this.month - other.month;

return this.day - other.day;

}

}

class TestDate

{

public static void Main()

{

Date d1 = new Date(10, 2, 2011);

Date d2 = new Date(20, 1, 2011);

Console.WriteLine(d1.CompareTo(d2));

}

} // TestDate

} // namespace

9. Accept a folder from user and a string. Display all files in the folder that contain the given string in the filename. Use a textbox to take folder name and another textbox for string. Use a Listbox to display the list of selected files.

private void btnSearch\_Click(object sender, EventArgs e)

{

DirectoryInfo d = new DirectoryInfo(txtFolder.Text);

FileInfo[] files = d.GetFiles();

lstFiles.Items.Clear();

foreach (FileInfo f in files)

{

if (f.Name.Contains(txtString.Text))

lstFiles.Items.Add(f.Name);

}

}

10. Display only non blank lines of the given file.

using System;

using System.IO;

namespace st

{

class NonBlankLines

{

public static void Main()

{

Console.Write("Enter a filename : ");

string filename = Console.ReadLine();

StreamReader sr = new StreamReader(filename);

string line = sr.ReadLine();

while (line != null)

{

if (line.Length > 0)

Console.WriteLine(line);

line = sr.ReadLine();

}

sr.Close();

} // Main

} // class

} // namespace

11. Display lines that contain the given string in the given file.

using System;

using System.IO;

namespace st

{

class SearchFile

{

public static void Main()

{

Console.Write("Enter a filename : ");

string filename = Console.ReadLine();

Console.Write("Enter search string : ");

string searchstring = Console.ReadLine();

StreamReader sr = new StreamReader(filename);

string line = sr.ReadLine();

while (line != null)

{

if (line.Contains(searchstring))

Console.WriteLine(line);

line = sr.ReadLine();

}

sr.Close();

} // Main

} // class

} // namespace

12. Take source file, target file, source string and target string. Replace all occurrences of source string with target string while writing content from source file to target file.

using System;

using System.IO;

namespace st

{

class SearchReplace

{

public static void Main()

{

Console.Write("Enter a source file : ");

string srcfilename = Console.ReadLine();

Console.Write("Enter a target file : ");

string trgfilename = Console.ReadLine();

Console.Write("Enter search string : ");

string searchstring = Console.ReadLine();

Console.Write("Enter replace string : ");

string replacestring = Console.ReadLine();

StreamReader sr = new StreamReader(srcfilename);

StreamWriter sw = new StreamWriter(trgfilename);

string line = sr.ReadLine();

while (line != null)

{

string newline = line.Replace(searchstring, replacestring);

sw.WriteLine(newline);

line = sr.ReadLine();

}

sr.Close();

sw.Close();

} // Main

} // class

} // Namespace

Answers to exercises in Advanced C# and .NET Library (Part 2)

Exercise : Provide operator functions for ==, !=, >, < and ++ for MyTime class. Add conversion operators to MyTime class to convert int (total seconds) to MyTime object

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Booklet3\_exercises

{

class MyTime

{

private int hour, min, sec;

public MyTime(int hour, int min, int sec)

{

this.hour = hour;

this.min = min;

this.sec = sec;

}

public int TotalSeconds

{

get

{

return hour \* 3600 + min \* 60 + sec;

}

}

public static bool operator ==(MyTime t1, MyTime t2)

{

return t1.TotalSeconds == t2.TotalSeconds;

}

public static bool operator !=(MyTime t1, MyTime t2)

{

return t1.TotalSeconds != t2.TotalSeconds;

}

public static bool operator >(MyTime t1, MyTime t2)

{

return t1.TotalSeconds > t2.TotalSeconds;

}

public static bool operator <(MyTime t1, MyTime t2)

{

return t1.TotalSeconds < t2.TotalSeconds;

}

public static MyTime operator ++(MyTime t1)

{

t1.sec++;

if (t1.sec == 60)

{

t1.sec = 0;

t1.min++;

if (t1.min == 60)

{

t1.min = 0;

t1.hour++;

if (t1.hour == 24)

t1.hour = 0;

}

}

return t1;

}

public override string ToString()

{

return String.Format("{0}:{1}:{2}", hour, min, sec);

}

// Conversion operator

public static explicit operator int(MyTime t)

{

return t.TotalSeconds;

}

}

class TestMyTime

{

public static void Main()

{

MyTime t1 = new MyTime(10, 20, 30);

MyTime t2 = new MyTime(10, 20, 30);

MyTime t3 = new MyTime(10, 20, 33);

MyTime t4 = new MyTime(10, 19, 30);

Console.WriteLine(t1);

Console.WriteLine(t1 == t2);

Console.WriteLine(t1 != t3);

Console.WriteLine(t2 > t4);

Console.WriteLine(t2 < t3);

Console.WriteLine(t1 == t3);

t1++;

Console.WriteLine(t1);

}

}

}

1. Create a class called Circle that stores x, y coordinates and radius. Overload == and != operators. Overload ++ operator that increases the x and y points by 1 unit. Add a couple of extension methods for Circle class to compare two objects, to print an object and to set x, y to center of given area in width and height and radius to 100.

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Booklet3\_exercises

{

public class Circle

{

public int x { get; set; }

public int y { get; set; }

public int r { get; set; }

public static bool operator ==(Circle c1, Circle c2)

{

return c1.x == c2.x && c1.y == c2.y && c1.r == c2.r;

}

public static bool operator !=(Circle c1, Circle c2)

{

return c1.x != c2.x || c1.y != c2.y || c1.r != c2.r;

}

public static Circle operator ++(Circle c1)

{

c1.x++;

c1.y++;

return c1;

}

}

class TestCircle

{

public static void Main()

{

Circle c1 = new Circle { x = 10, y = 10, r = 10 };

Circle c2 = new Circle { x = 10, y = 10, r = 10 };

Circle c3 = new Circle { x = 10, y = 5, r = 50 };

Console.WriteLine(c1 == c2);

Console.WriteLine(c1 == c3);

c1++;

c1.Print();

c2.set(100,200);

c2.Print();

}

}

public static class CircleExtensionsMethods

{

public static bool Compare(this Circle c1, Circle c2)

{

return c1 == c2;

}

public static void Print(this Circle c1)

{

Console.WriteLine("(x, y) : ({0}, {1})", c1.x, c1.y);

Console.WriteLine("Radius : {0}", c1.r);

}

public static void set(this Circle c1, int width, int height)

{

c1.x = width / 2;

c1.y = height / 2;

c1.r = 100;

}

}

}

2.Create a generic class that implements a Queue data structure with the following methods and properties:

Add(T)

Remove()

Contains(T)

Print()

Length

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Booklet3\_exercises

{

class GenericQueue

{

private T[] data = new T[10];

int rear = 0, front=0;

public void Add(T v)

{

if (rear == data.Length)

throw new Exception("Queue is full!");

data[rear] = v;

rear++;

}

public T Remove()

{

if (front < rear)

{

return data[front++];

}

else

throw new Exception("Empty Queue");

}

public bool Contains (T v) {

for (int i = front; i < rear; i++)

{

if (data[i].Equals(v))

return true;

}

return false;

}

public void Print()

{

for (int i = front; i < rear; i++)

Console.WriteLine(data[i]);

}

public int Length

{

get

{

return rear - front;

}

}

}

class TestQueue

{

public static void Main()

{

var q = new GenericQueue<string>();

q.Add("Abc");

q.Add("Xyz");

q.Print();

Console.WriteLine(q.Remove() );

Console.WriteLine(q.Remove());

Console.WriteLine(q.Length);

q.Add("Pqr");

q.Add("Def");

q.Print();

}

}

}

3. Create a delegate type that points to a method, which takes an array of int as the parameter and performs operations on the array. Make the delegate point to methods like Print(), Reverse() and Sort() and perform those operations.

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Booklet3\_exercises

{

class DelegateOperations

{

delegate void ArrayOperation(int [] a);

static void Main()

{

int[] a = { 10, 211, 50, 133, 144 };

ArrayOperation printop = Print;

ArrayOperation op;

printop(a);

op = Sort;

op(a);

Console.WriteLine("After Sorting");

printop(a);

op = Reverse;

op(a);

Console.WriteLine("After Reversing");

printop(a);

}

public static void Sort(int [] a)

{

Array.Sort(a);

}

public static void Reverse(int [] a)

{

Array.Reverse (a);

}

public static void Print(int [] a)

{

foreach (int n in a)

Console.WriteLine(n);

}

}

}

4. Create a function that takes any type of object and calls Print() method of the given object.

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Booklet3\_exercises

{

class Test1

{

public void Print()

{

Console.WriteLine("Print in Test1");

}

}

class Test2

{

public void Print()

{

Console.WriteLine("Print in Test2");

}

}

class DynamicExercise

{

public static void Main()

{

Print(new Test1());

Print(new Test2());

}

public static void Print(dynamic d)

{

d.Print();

}

}

}

5. Create a generic function called Insert() that takes an array, value to be inserted and position where it is to be inserted. Insert the value at the given position in the array. Similarly provide generic Delete() function that takes array and the position from where the value is to be deleted and deletes it.

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Booklet3\_exercises

{

class GenericMethods

{

public static void Main()

{

int[] arr = { 10, 20, 30, 40, 50 };

Insert(arr, 15, 1);

Print(arr);

Delete(arr, 0);

Print(arr);

}

public static void Print<T>(T[] a)

{

foreach (T v in a)

Console.WriteLine(v);

}

public static void Insert<T> ( T [] a, T v, int pos)

{

// push elements to right

for (int i = a.Length - 1; i > pos; i--)

a[i] = a[i - 1];

a[pos] = v;

}

public static void Delete<T>(T[] a, int pos)

{

// push elements to left

for (int i = pos; i < a.Length - 1; i ++ )

a[i] = a[i + 1];

a[a.Length - 1] = default(T); // set last element to default

}

}

}

6. Accept two files and display common lines between those two files (Intersection).

using System;

using System.Collections.Generic;

using System.IO;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Booklet3\_exercises

{

class Intersection

{

public static void Main()

{

Console.Write("Enter First File : ");

string filename1 = Console.ReadLine();

Console.Write("Enter Second File : ");

string filename2 = Console.ReadLine();

var lines = new HashSet<string>();

using (StreamReader sr1 = new StreamReader(filename1))

{

string line = sr1.ReadLine();

while (line != null)

{

lines.Add(line);

line = sr1.ReadLine();

}

}

var commonlines = new HashSet<string>();

using (StreamReader sr2 = new StreamReader(filename2))

{

string line = sr2.ReadLine();

while (line != null)

{

if (lines.Contains(line))

commonlines.Add(line);

line = sr2.ReadLine();

}

}

foreach (string line in commonlines)

Console.WriteLine(line);

}

}

}

7. Accept a file and display how many times each word is present in the file.

using System;

using System.Collections.Generic;

using System.IO;

namespace Booklet3\_exercises

{

class WordCount

{

public static void Main()

{

Console.Write("Enter Filename : ");

string filename = Console.ReadLine();

var wordscount = new SortedDictionary<string, int>();

using (StreamReader sr = new StreamReader(filename))

{

string line = sr.ReadLine();

while (line != null)

{

string[] words = line.Split('.', ' ');

foreach (string word in words)

{

if (word.Length == 0)

continue;

if (wordscount.ContainsKey(word))

wordscount[word]++;

else

wordscount.Add(word, 1);

}

line = sr.ReadLine();

}

}

foreach (string word in wordscount.Keys)

Console.WriteLine("[{0}] occurs [{1}]", word, wordscount[word]);

}

}

}

8. Accept a file and display all lines except duplicated lines

using System;

using System.Collections.Generic;

using System.IO;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Booklet3\_exercises

{

class NonDuplicateLines

{

public static void Main()

{

Console.Write("Enter Filename : ");

string filename = Console.ReadLine();

var lines = new HashSet<string>();

var duplines = new HashSet<string>();

using (StreamReader sr = new StreamReader(filename))

{

string line = sr.ReadLine();

while (line != null)

{

if (lines.Contains(line))

duplines.Add(line);

else

lines.Add(line);

line = sr.ReadLine();

}

}

foreach (string line in lines)

{

if (!duplines.Contains(line))

Console.WriteLine(line);

}

}

}

}

9. Accept a file from user and display all words in sorted order by eliminating duplicates.

using System;

using System.Collections.Generic;

using System.IO;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Booklet3\_exercises

{

class SortedWords

{

public static void Main()

{

Console.Write("Enter filename : ");

string filename = Console.ReadLine();

var sortwords = new SortedSet<string>();

using (StreamReader sr = new StreamReader(filename))

{

string line = sr.ReadLine();

while (line != null)

{

string[] words = line.Split('.', ' ');

foreach (string word in words)

{

if (word.Length != 0)

sortwords.Add(word);

}

line = sr.ReadLine();

}

}

foreach (string word in sortwords)

Console.WriteLine(word);

}

}

}

10. Accept a file and remove all blank lines from the file and then write the new content into the same file.

using System;

using System.Collections.Generic;

using System.IO;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Booklet3\_exercises

{

class RemoveBlankLines

{

public static void Main()

{

Console.Write("Enter Filename : ");

string filename = Console.ReadLine();

var lines = new List<string>();

using (StreamReader sr = new StreamReader(filename))

{

string line = sr.ReadLine();

while (line != null)

{

if (line.Trim().Length > 0)

lines.Add(line);

line = sr.ReadLine();

}

}

using (StreamWriter sw = new StreamWriter(filename))

{

foreach (string line in lines)

sw.WriteLine(line);

}

}

}

}