**Property**

**Create a class `Rectangle` with two properties `Length` and `Width`. These properties**

**should ensure that negative values are not allowed. If a user tries to set a negative value for**

**either `Length` or `Width`, it should automatically default to 0. Write a method**

**`GetArea()` that returns the area of the rectangle.**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Runtime.Remoting.Messaging;

using System.Text;

using System.Threading.Tasks;

namespace ConsoleApp1

{

class Reectangle

{

private double length;

private double width;

public double Length

{

get

{

return length;

}

set

{

length = value;

if (value < 0)

{

length = 0;

}

}

}

public double Width

{

get

{

return width;

}

set

{

width = value;

if (value < 0)

{

width = 0;

}

}

}

public double GetArea()

{

double area = Width \* Length;

return area;

}

}

class Access

{

public static void Main(string[] args)

{

Reectangle r = new Reectangle();

r.Length = -3.44;

r.Width = -2.12;

r.GetArea();

Console.WriteLine(r.GetArea());

}

}

}

**Design a class `Employee` that has three properties: `FirstName`, `LastName`, and a**

**\*\*read-only\*\* property `FullName`. The `FullName` property should return the**

**concatenation of `FirstName` and `LastName`. Instantiate the class and display the full**

**name of an employee.**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Runtime.Remoting.Messaging;

using System.Text;

using System.Threading.Tasks;

namespace ConsoleApp1

{

class Employee

{

public string FirstName

{

get;set;

}

public string LastName

{

get;set;

}

public string FullName

{

get

{

return FirstName + " " + LastName;

}

}

public void display()

{

Console.WriteLine(FullName);

}

}

class Access

{

public static void Main(string[] args)

{

Employee e = new Employee();

e.FirstName = "Tasnia";

e.LastName = "Falguni";

e.display();

}

}

}

**Indexer**

**Create a class `ShoppingCart` that stores a list of products using an array of strings.**

**Implement an indexer to access the products in the shopping cart by index. Write a**

**program that adds products to the shopping cart, retrieves them using the indexer, and**

**displays them.**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Runtime.Remoting.Messaging;

using System.Text;

using System.Threading.Tasks;

namespace ConsoleApp1

{

class ShoppingCart

{

private string[] products = new string[6];

public string this[int index]

{

get

{

if(index>=0 && index < products.Length)

{

return products[index];

}

throw new IndexOutOfRangeException("Invalid index");

}

set

{

if(index>=0 && index < products.Length)

{

products[index] = value;

}

}

}

}

class Access

{

public static void Main(string[] args)

{

ShoppingCart product = new ShoppingCart();

product[0] = "Mobile";

product[1] = "Laptop";

product[2] = "Desktop";

product[3] = "Tablet";

for(int i=0; i<5; i++)

{

Console.WriteLine(product[i]);

}

}

}

}

using System;

using System.Collections.Generic;

using System.Linq;

using System.Runtime.Remoting.Messaging;

using System.Text;

using System.Threading.Tasks;

namespace ConsoleApp1

{

class Car

{

private string model;

private string brand;

public string Model

{

get { return model; }

set { model = value; }

}

public string Brand

{

get { return brand; }

set { brand = value; }

}

public Car(string model, string brand)

{

Model = model;

Brand = brand;

}

}

class CarCollection

{

private Car[] car = new Car[5];

public Car this[int index]

{

get

{

if (index>=0 && index < car.Length)

{

if (car[index] != null)

{

return car[index];

}

}

throw new IndexOutOfRangeException("Invalid index");

}

set

{

if(index>=0 && index < car.Length)

{

car[index] = value;

}

}

}

}

class Access

{

public static void Main(string[] args)

{

CarCollection c = new CarCollection();

c[0] = new Car("Corolla", "Toyota");

c[1] = new Car("Civic", "Honda");

Console.WriteLine(c[0].Model);

Console.WriteLine(c[1].Brand);

}

}

}

**Inheritance**

**Single Inheritance**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Runtime.Remoting.Messaging;

using System.Text;

using System.Threading.Tasks;

namespace ConsoleApp1

{

class Vehicle

{

public void Move()

{

Console.WriteLine("Vehicle is moving");

}

}

class Car : Vehicle

{

public void StartEngine()

{

Console.WriteLine("Car engine started");

}

}

class Access

{

public static void Main(string[] args)

{

Car c = new Car();

c.Move();

c.StartEngine();

}

}

}

**Multilevel Inheritance**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Runtime.Remoting.Messaging;

using System.Text;

using System.Threading.Tasks;

namespace ConsoleApp1

{

class Vehicle

{

public void Move()

{

Console.WriteLine("Vehicle is moving");

}

}

class Car : Vehicle

{

public void StartEngine()

{

Console.WriteLine("Car engine started");

}

}

class ElectricCar : Car

{

public void Charging()

{

Console.WriteLine("Battery is charging");

}

}

class Access

{

public static void Main(string[] args)

{

ElectricCar c = new ElectricCar();

c.Move();

c.StartEngine();

c.Charging();

}

}

}

**Hierarchical Inheritance**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Runtime.Remoting.Messaging;

using System.Text;

using System.Threading.Tasks;

namespace ConsoleApp1

{

class Vehicle

{

public void Move()

{

Console.WriteLine("Vehicle is moving");

}

}

class Car : Vehicle

{

public void StartEngine()

{

Console.WriteLine("Car engine started");

}

}

class Bike : Vehicle

{

public void Sound()

{

Console.WriteLine("Bike make sound");

}

}

class Access

{

public static void Main(string[] args)

{

Car c = new Car();

Bike b = new Bike();

c.Move();

c.StartEngine();

b.Move();

b.Sound();

}

}

}

**Multiple Inheritance**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Runtime.Remoting.Messaging;

using System.Text;

using System.Threading.Tasks;

namespace ConsoleApp1

{

public interface Ifuel

{

void Refuel();

}

public interface Ielectric

{

void ChargeBattery();

}

public class MultiFuelCar : Ifuel, Ielectric

{

public void Refuel()

{

Console.WriteLine("MultiFuel Car is Refueling");

}

public void ChargeBattery()

{

Console.WriteLine("MultiFuel Car is Charging the battery");

}

}

class Access

{

public static void Main(string[] args)

{

MultiFuelCar car = new MultiFuelCar();

car.Refuel();

car.ChargeBattery();

}

}

}

**Hybrid Inheritance**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Runtime.Remoting.Messaging;

using System.Security.Cryptography.X509Certificates;

using System.Security.Permissions;

using System.Text;

using System.Threading.Tasks;

namespace ConsoleApp1

{

class Vehicle

{

public void Move()

{

Console.WriteLine("Vehicle is moving");

}

}

class Car : Vehicle

{

public void StartEngine()

{

Console.WriteLine("Car engine started");

}

}

class Bike : Vehicle

{

public void BikeSound()

{

Console.WriteLine("Bike make sound");

}

}

class ElectricCar : Car

{

public void Charging()

{

Console.WriteLine("Electric car is charging");

}

}

class Access

{

public static void Main(string[] args)

{

ElectricCar ecar = new ElectricCar();

ecar.Move();

ecar.StartEngine();

ecar.Charging();

Bike b = new Bike();

b.Move();

b.BikeSound();

}

}

}

**Enumeration**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Runtime.Remoting.Messaging;

using System.Security.Cryptography.X509Certificates;

using System.Security.Permissions;

using System.Text;

using System.Threading.Tasks;

namespace ConsoleApp1

{

enum Car { Start,Go,Back,Right,Left,Stop};

class Access

{

public static void Main(string[] args)

{

Console.WriteLine("Please Enter a command:");

string input = Console.ReadLine();

if(Enum.TryParse(input,true,out Car carcommand))

{

switch (carcommand)

{

case Car.Start:

Console.WriteLine("Car is start now");

break;

case Car.Go:

Console.WriteLine("Go forward");

break;

case Car.Left:

Console.WriteLine("Turning Left");

break;

case Car.Right:

Console.WriteLine("Turning Right");

break;

case Car.Stop:

Console.WriteLine("Car is stop now");

break;

}

}

}

}

}

using System;

using System.Collections.Generic;

using System.Linq;

using System.Runtime.Remoting.Messaging;

using System.Security.Cryptography.X509Certificates;

using System.Security.Permissions;

using System.Text;

using System.Threading.Tasks;

namespace ConsoleApp1

{

enum Days { Saturday=1,Sunday,Monday,Tuesday,Wednesday,Thursday,Friday};

class Access

{

public static void Main(string[] args)

{

Console.WriteLine("Please Enter your input");

string input = Console.ReadLine();

if (int.TryParse(input, out int daynum))

{

if (daynum >= 1 && daynum <= 7)

{

Console.WriteLine((Days)daynum);

}

}

else if(Enum.TryParse(input, true, out Days dayname))

{

Console.WriteLine((int)dayname);

}

}

}

}