Факультет компьютерных технологий

Лабораторная работа №8

«Разработка программ с использованием управляющих инструкций»

Вариант №10

Выполнил: студент гр. 981063 Ефименко Павел Викторович

Проверил: Воробей В.А.

Минск 2020

Задание №1

Условие:

Создать и реализовать интерфейсы, также использовать наследование и полиморфизм для указанных предметных областей. 10. interface Фильм <- class Отечественный Фильм <- class Комедия.

class Program

{

static void Main(string[] args)

{

var comedy = new Comedy { Duration = "1 час 20 мин.", Language = "Русский", Name = "Комедия. Как украсть соседа" };

var domestic = new DomesticFilm { Duration = "2 час 20 мин.", Language = "Русский", Name = "День танкиста" };

var films = new List<IFilm> { comedy, domestic };

foreach (var film in films)

{

Console.WriteLine(film);

}

}

}

public interface IFilm

{

string Name { get; set; }

string Duration { get; set; }

string Language { get; set; }

void Play();

void Stop();

void Pause();

}

internal class DomesticFilm : IFilm

{

public string Name { get; set; }

public string Duration { get; set; }

public string Language { get; set; }

public void Pause()

{

Console.WriteLine($"Фильм:\t{Name} - Приостановлен");

}

public void Play()

{

Console.WriteLine($"Приятного просмотра");

}

public void Stop()

{

Console.WriteLine($"Фильм:\t{Name} - Остановлен");

}

public override string ToString()

{

return $"Название фильма:\t{Name};\tПродолжительность:\t{Duration};\tЯзык озвучки:\t{Language}";

}

}

internal class Comedy : DomesticFilm

{

public string Name { get; set; }

public string Duration { get; set; }

public string Language { get; set; }

public void Pause() => Console.WriteLine($"Фильм:\t{Name} - Приостановлен");

public void Play() => Console.WriteLine($"Приятного просмотра");

public void Stop() => Console.WriteLine($"Фильм:\t{Name} - Остановлен");

public override string ToString()

{

return $"Название фильма:\t{Name};\tПродолжительность:\t{Duration};\tЯзык озвучки:\t{Language}";

}

}

Результат:



Задание №2

Условие:

Необходимо в задание 1 лабораторной работы №7: для всех цепочек классов связных наследованием, добавить абстрактные классы и интерфейсы, необходимые по логике.

public abstract class Bus : IRoadTransportEquipment

{

public Bus(string name, string color, string type, int power)

{

Name = name;

Color = color;

Type = type;

Power = power;

}

public Bus()

{

}

public string Name { get; set; }

public string Color { get; set; }

public string Type { get; set; }

public int Power { get; set; }

public virtual void OpenDoor()

{

Console.WriteLine("The door opened");

}

public virtual void OpenDoor(string additionalMessage)

{

Console.WriteLine($"The door opened.{additionalMessage}");

}

public virtual void CloseDoor()

{

Console.WriteLine("The door closed");

}

public virtual void CloseDoor(string additionalMessage)

{

Console.WriteLine($"The door closed.{additionalMessage}");

}

public virtual void Beep()

{

Console.WriteLine("Beeeeeeeep!");

}

public virtual void Beep(string additionalMessage)

{

Console.WriteLine($"Beeeeeeeep!\t{additionalMessage}");

}

public virtual void EngineStart()

{

Console.WriteLine("Engine started");

}

public virtual void EngineHalt()

{

Console.WriteLine("Engine halted");

}

public override string ToString()

{

return $"{Name}\t{Color}\t{Type}\t{Power}";

}

}

internal abstract class ElectroBus : Bus

{

public override string ToString()

{

return $"{Name}\t{Color}\t{Type}\t{Power}";

}

}

internal class ElectroBelarussianBus : Bus

{

public ElectroBelarussianBus(IRoadTransportEquipment bus)

{

this.Color = bus.Color;

this.Name = bus.Name;

this.Power = bus.Power;

this.Type = bus.Type;

}

public override void Beep()

{

Console.WriteLine($"Ту-туууууууу");

}

}

Результат:

