

МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ
«БРЕСТСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

ФАКУЛЬТЕТ ЭЛЕКТРОННО-ИНФОРМАЦИОННЫХ СИСТЕМ

Кафедра интеллектуальных информационных технологий

Отчет по лабораторной работе №5

Выполнила
Радиончик С.С.,
студентка группы ПО-5

Проверил
Крощенко А.А.,
ст. преп. Кафедры ИИТ,
«__» _____ 2021 г.

Брест, 2021

Цель работы: приобрести практические навыки в области объектно-ориентированного проектирования.

Вариант 6.

Задание 1. Реализовать абстрактные классы или интерфейсы, а также наследование и полиморфизм для следующих классов:

6) *interface Mobile ← abstract class Samsung Mobile ← class Model.*

Задание 2. Требуется создать суперкласс (абстрактный класс, интерфейс) и определить общие методы для данного класса. Создать подклассы, в которых добавить специфические свойства и методы. Часть методов переопределить. Создать массив объектов суперкласса и заполнить объектами подклассов. Объекты подклассов идентифицировать конструктором по имени или идентификационному номеру. Использовать объекты подклассов для моделирования реальных ситуаций и объектов.

6) *Создать суперкласс Домашнее животное и подклассы Собака, Кошка, Попугай. С помощью конструктора установить имя каждого животного и его характеристики*

Задание 3. В задании 3 ЛР №4, где возможно, заменить объявления суперклассов объявлениями абстрактных классов или интерфейсов.

Выполнение:

Код программы.

1) IMobile.cs

```
namespace Lab5_1
{
    interface IMobile
    {
        string GetTitle();
        int GetPrice();
        void Print();
    }
}
```

SamsungMobile.cs

```
using System;

namespace Lab5_1
{
    abstract class SamsungMobile: IMobile
    {
        private string model;
        private int price;

        protected SamsungMobile(string model, int price)
        {
            this.model = model;
            this.price = price;
        }

        protected void SetPrice(int price)
```

```

        {
            this.price = price;
        }

        public int GetPrice()
        {
            return this.price;
        }

        public string GetTitle()
        {
            return $"Samsung { this.model}";
        }

        public void Print()
        {
            Console.WriteLine($"Model: Samsung {this.model}\nPrice: {this.price}\n\n");
        }
    }
}

```

Model.cs

```

using System;

namespace Lab5_1
{
    class Model: SamsungMobile
    {
        private string description;

        public Model(string model, int price, string description) : base(model, price)
        {
            this.description = description;
        }

        public void SetDescription(string description)
        {
            this.description = description;
        }

        public string GetDescription()
        {
            return this.description;
        }

        public new void Print()
        {
            Console.WriteLine($"Model: {this.GetTitle()}\nPrice:
{this.GetPrice()}\nDescription: {this.description}\n\n");
        }
    }
}

```

Program.cs

```

using System;

namespace Lab5_1
{
    class Program
    {
        static void Main(string[] args)
        {
            Model model = new Model("x77", 360, "new version.");
        }
    }
}

```

```

        model.Print();
    }
}

```

2) *Pet.cs*

```

namespace Lab5_2
{
    abstract class Pet
    {
        private string name;
        private string color;

        protected Pet(string name, string color)
        {
            this.name = name;
            this.color = color;
        }

        public void SetName(string name)
        {
            this.name = name;
        }

        public void SetColor(string color)
        {
            this.color = color;
        }

        public string GetName()
        {
            return this.name;
        }

        public string GetColor()
        {
            return this.color;
        }

        public abstract void DetData();
    }
}

```

Dog.cs

```

using System;

namespace Lab5_2
{
    class Dog : Pet
    {
        private bool IsTrained;

        public Dog(string name, string color, bool isTrained) : base(name, color)
        {
            this.IsTrained = isTrained;
        }

        public void SetIsTrained(bool isTrained)
        {
            this.IsTrained = isTrained;
        }

        private bool GetIsTrained()

```

```

        {
            return this.IsTrained;
        }

        public override void DetData()
        {
            Console.WriteLine($"--Dog--\nName: {this.GetName()}\nColor:
{this.GetColor()}\nHousehold: {this.GetIsTrained()}");
        }
    }
}

```

Cat.cs

```

using System;

namespace Lab5_2
{
    class Cat : Pet
    {
        private bool IsHousehold;

        public Cat(string name, string color, bool isHousehold) : base(name, color)
        {
            this.IsHousehold = isHousehold;
        }

        public void SetIsHousehold(bool isHousehold)
        {
            this.IsHousehold = isHousehold;
        }

        private bool GetIsHousehold()
        {
            return this.IsHousehold;
        }

        public override void DetData()
        {
            Console.WriteLine($"--Cat--\nName: {this.GetName()}\nColor:
{this.GetColor()}\nHousehold: {this.GetIsHousehold()}");
        }
    }
}

```

Parrot.cs

```

using System;

namespace Lab5_2
{
    class Parrot : Pet
    {
        private bool IsTalking;

        public Parrot(string name, string color, bool isTalking) : base(name, color)
        {
            this.IsTalking = isTalking;
        }

        public void SetIsTalking(bool isTalking)
        {
            this.IsTalking = isTalking;
        }
    }
}

```

```

        private bool GetIsTalking()
        {
            return this.IsTalking;
        }

        public override void DetData()
        {
            Console.WriteLine($"--Parrot--\nName: {this.GetName()}\nColor:
{this.GetColor()}\nHousehold: {this.GetIsTalking()}");
        }
    }
}

```

Program.cs

```

using System;
using System.Collections.Generic;

namespace Lab5_2
{
    class Program
    {
        static void Main(string[] args)
        {
            List<Pet> pets = new List<Pet>();
            pets.Add(new Dog("Bobik", "Black", true));
            pets.Add(new Cat("Simon", "White", true));
            pets.Add(new Parrot("Keshha", "Yellow", false));

            foreach(var item in pets)
            {
                item.DetData();
                Console.WriteLine();
            }
        }
    }
}

```

3)

```
abstract class SystemUser
```

Результаты работы программы:


1)

```

C:\Users\User\Desktop\Света\Бг
Model: Samsung x77
Price: 360
Description: new version.

```

2)

 Консоль отладки Microsoft Visual Studio

--Dog--

Name: Bobik

Color: Black

Household: True

--Cat--

Name: Simon

Color: White

Household: True

--Parrot--

Name: Kesha

Color: Yellow

Household: False

Вывод: приобрела практические навыки в области объектно-ориентированного проектирования.