Пенза 2020

Министерство образования Российской Федерации

Пензенский государственный университет

Кафедра «Вычислительная техника»

Выполнил:

студенты группы 17ВВ1

Кокин Денис

Беленков Никита

Принял:

Дорошенко И.Н.

**ОТЧЁТ**

по лабораторной работе №1

по курсу «Технологии программирования»

на тему «Класс и наследование»

**Цель работы:** Требуется разработать набор классов описания исключительной ситуации.

Разработать консольное приложение, в котором должен быть реализован набор классов в соответствии с вариантом задания.

Должны быть разработаны базовый класс (**Base**) не менее 3 производных классов (**Derived1**, **Derived2** и **Derived3**). Каждый производный класс должен расширять базовый. В каждый производный класс необходимо добавить переменные и функции, уточняющие базовый класс.

Структура классов должна быть следующей:

* **Base** : **Derived1**
* **Base** : **Derived2**
* **Derived2** : **Derived3**

Производные классы **Derived1** и **Derived2** должны быть пронаследованы от базового класса (**Base**). Класс **Derived3** должен быть пронаследован от производного класса **Derived2**.

Требования к классам:

* Класс должен содержать конструктор без параметров. В конструкторе должны инициализироваться переменные класса значением по умолчанию.
* Класс должен содержать два или более конструкторов с параметрами. В конструкторах должны инициализироваться переменные класса значениями, которые переданы в параметрах конструктора. Если для какой-либо переменной класса в параметрах конструктора не предусмотрен параметр, то переменная должна инициализироваться значением по умолчанию.
* Класс должен содержать деструктор. В деструкторе все переменные класса должны быть затерты (присвоены недействительные значения).
* Класс должен содержать три или более методов класса (функций). Методы должны быть реализованы в трех секциях: public, protected и private. В каждом методе должно быть обращение к одной или более переменной класса.
* Класс должен содержать три или более (в соответствии с вариантом задания) переменных класса. Переменные класса должны быть реализованы в трех секциях: public, protected и private.
* Класс должен содержать одну или более переменную со строковыми данными.
* В классе должны быть реализованы два метода: Input и Output. Метод Input должен выполнять ввод данных класса с клавиатуры. Метод Output должен печатать на экране данные класса. Методы должны располагаться в секции public. В производных классах (**Derived1**, **Derived2** и **Derived3**) методы должны использовать вызовы из родительского класса. Т.е. в производном классе в функциях первоначально должны быть вызваны функции базового класса (для **Derived1** и **Derived2** функции из класса **Base**, для **Derived3** функции из класса **Derived2**), а затем выполнен ввод и вывод данных текущего класса.

Требования к программе:

* В основной функции программы создать объект класса с вызовом конструктора без параметров.
* В основной функции программы создать несколько объектов класса с вызовом конструкторов с параметрами.
* Для любого созданного объекта класса вызвать функции Input, Output и все остальные методы, реализованные в классе. Вызов можно выполнять как из основной функции программы, так и из любой функции класса (это касается функций, объявленных в секциях protected и private).

**Листинг программы**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Confrontation {

class Locality {

public String Name;

public String Region;

public Locality() {

Population = 0;

Budget = 0;

Code = 0;

}

public Locality(int Population, int Budget, int Code, String Name = "Unnamed", String Region = "Unnamed")

{

this.Population = Population;

this.Budget = Budget;

this.Code = Code;

this.Name = Name;

this.Region = Region;

}

public Locality(String Name, String Region, int Population = 0, int Budget = 0, int Code = 0)

{

this.Population = Population;

this.Budget = Budget;

this.Code = Code;

this.Name = Name;

this.Region = Region;

}

public void ClassInfo()

{

Console.WriteLine("Full region name: " + this.Name + " in " + this.Region);

this.AbleBodied();

this.BudgetToUSD();

}

~Locality()

{

Name = null;

Region = null;

Population = 0;

Budget = 0;

Code = 0;

}

public void input() {

Console.Write("Input a name of the region: ");

this.Name = Console.ReadLine();

Console.Write("Input a regional center of the region: ");

this.Region = Console.ReadLine();

Console.Write("Input a population of the region: ");

this.Population = Int32.Parse(Console.ReadLine());

Console.Write("Input a budget of the region: ");

this.Budget = Int32.Parse(Console.ReadLine());

Console.Write("Input a code of the region: ");

this.Code = Int32.Parse(Console.ReadLine());

}

public void output() {

Console.WriteLine("A name of the region: " + this.Name);

Console.WriteLine("A regional center of the region: " + this.Region);

Console.WriteLine("A population of the region: " + this.Population);

Console.WriteLine("A budget of the region: " + this.Budget);

Console.WriteLine("A code of the region: " + this.Code);

}

protected int Population;

protected int Code;

protected int Budget;

protected void AbleBodied() {

Console.WriteLine("Able-bodied population: " + (this.Population / 2));

}

private void BudgetToUSD() {

Console.WriteLine("Budget of locality: " + (this.Budget / 70) + " USD");

}

}

class City : Locality {

public bool isRegionCenter;

public void output(){

base.output();

Console.WriteLine("This is a region center: " + this.isRegionCenter);

}

public void input(){

base.input();

Console.Write("Is this a region center?: ");

isRegionCenter = System.Convert.ToBoolean(Console.ReadLine());

}

}

class Township : Locality

{

public int SchoolCount;

public void output()

{

base.output();

Console.WriteLine("A count of schools: " + this.SchoolCount);

}

public void input()

{

base.input();

Console.Write("Enter a count of schools: ");

SchoolCount = Int32.Parse(Console.ReadLine());

}

}

class Village : Township

{

public int ShopCount;

public void output()

{

base.output();

Console.WriteLine("A count of shops: " + this.ShopCount);

}

public void input()

{

base.input();

Console.Write("Enter a count of shops: ");

ShopCount = Int32.Parse(Console.ReadLine());

}

}

class Program{

static void Main(string[] args){

Locality EmptyLocality = new Locality();

EmptyLocality.input();

EmptyLocality.ClassInfo();

Locality LocalityWithParams = new Locality("TestName", "TestRegion");

LocalityWithParams.input();

LocalityWithParams.ClassInfo();

City CitySample = new City();

CitySample.input();

CitySample.ClassInfo();

Township TownshipCample = new Township();

TownshipCample.input();

TownshipCample.ClassInfo();

Village VillageSample = new Village();

VillageSample.input();

VillageSample.ClassInfo();

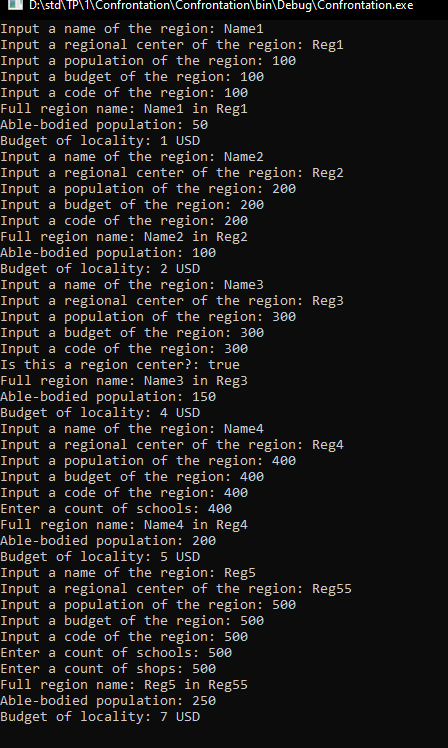
Console.Read();

}

}

}

**Результаты работы программы**



**Вывод:** студенты изучили механизмы сохранения и восстановления объектов данных. Получили навыки разработки самовосстанавливающихся структур данных.