Министерство образования и науки Российской Федерации

Федеральное государственное автономное образовательное учреждение высшего профессионального образования

«Уральский федеральный университет имени первого Президента России Б.Н. Ельцина»

Институт фундаментального образования

Кафедра «Информационные системы и технологии»

Абстракция Объекта

ОТЧЕТ

по лабораторной работе №1

Руководитель : Александров О.Е

Студент: Антропов А.Ю.

Группа: ФО-250003

Екатеринбург

2016

**Абстракция объекта**

Модификатор abstract указывает, что реализация изменяемого объекта является неполной или отсутствует. Модификатор abstract может использоваться с классами, методами, свойствами, индексаторами и событиями. Модификатор abstract в объявлении класса указывает, что класс предназначен только для использования в качестве базового класса для других классов. Члены, помеченные как абстрактные или включенные в абстрактный класс, должны быть реализованы с помощью классов, производных от абстрактных классов.

Возможности абстрактных классов.

* Создавать экземпляры абстрактного класса нельзя.
* Абстрактные классы могут содержать абстрактные методы и методы доступа.
* Абстрактный класс с модификатором [запечатанные](https://msdn.microsoft.com/ru-ru/library/88c54tsw.aspx) изменить нельзя, поскольку эти два модификатора имеют взаимоисключающие значения. Модификатор sealed запрещает наследовать классу, в то время как модификатор abstract указывает, что класс обязан иметь производные классы.
* Неабстрактный класс, являющийся производным от абстрактного, должен содержать фактические реализации всех наследуемых абстрактных методов и методов доступа.

**Классы и их описание.**

Для написания программы по созданию и дальнейшему использованию генеалогического древа какой-либо семьи были использованы следующие абстрактные классы:

* Class Person
* Class FamilyTree

Class Person

Public char Sex – Свойство. Используется для задания экземпляру класса Person’a - Пола человека. Для проверки корректности введенных данных используется get/set.

private char \_Sex;

public char Sex

{

get

{

return \_Sex;

}

set

{

if (value == 'm' || value == 'f')

{

\_Sex = value;

}

else

{

throw new System.Exception("Неверное значение пола " + value);

}

}

}

public Person Mothers – Свойство. Используется для задания экземпляру класса Person - Матери человека. Для проверки корректности введенных данных используется get/set.В set проводится проверка на правильность введенных данных -пол и единственность (конструкция if).

private Person \_Mothers;

public Person Mothers

{

get

{

return \_Mothers;

}

set

{

if (value.Sex != 'f')

{

throw new System.Exception("Мама должна быть f п");

}

CheckChildes(value);

\_Mothers = value;

if (!\_Mothers.Childes.Contains(this)) { \_Mothers.Childes.Add(this); }

}

}

public Person Fathers – Свойство. Используется для задания экземпляру класса Person - Отца человека. Для проверки корректности введенных данных используется get/set.В set проводится проверка на правильность введенных данных -пол и единственность (конструкция if).

private Person \_Fathers;

public Person Fathers

{

get

{

return \_Fathers;

}

set

{

if (value.Sex != 'm')

{

throw new System.Exception("Папа должен быть m п");

}

CheckChildes(value);

\_Fathers = value;

if (!\_Fathers.Childes.Contains(this)) { \_Fathers.Childes.Add(this); }

}

}

public bool CheckChildes(Person p) - Данное свойство используется для проверки списка детей заданного экземпляра класса с помощью оператора foreach(Person c in Childes), если объекта нет в списке детей- return true.

public bool CheckChildes(Person p)

{

if (p == this)

{

throw new System.Exception("Персона" + p.ToString() + " уже в потмках");

}

foreach (Person c in Childes)

{

c.CheckChildes(p);

}

return true;

}

public bool CheckName(string **CheckedName**) – свойство. Необходимо для проверки, существует ли имя с таким же значением, если да, то выводит ошибку.

public bool CheckName(string CheckedName)

{

foreach (Person p in AllPersons)

{

if (p.Name == CheckedName)

{

throw new System.Exception("Имя " + CheckedName + " уже используется");

}

}

return true;

}

public string Name – свойство. Используется для задания экземпляру класса Person - Имени. Также проводится проверка на повторность введенных данных.

public string Name

{

get

{

return PName;

}

set

{ //проверили предков

if (Mothers != null)

{

if (Mothers.PName == value)

{

throw new Exception("Имя " + value + " уже используется");

}

foreach (Person Child in Childes)

{

Child.CheckName(CheckedName);

}

}

if (Fathers != null)

{

if (Fathers.PName == value)

{

throw new Exception("Имя " + value + " уже используется");

}

foreach (Person Child in Childes)

{

Child.CheckName(CheckedName);

}

}

PName = value;

}

}

Class FamilyTree

public Person FindPerson(string Name) - С помощью метода Find(p => p.Name == Name) находим нужный для нас элемент в списке AllPersons.

public Person FindPerson(string Name)

{

return Person.AllPersons.Find(p => p.Name == Name);

}

public void ParseLine(string) - Из заданной строки методом s.IndexOf()находим " -> ", если данная строка не найдена, метод возвращает -1, следовательно выводим сообщение о не нахождении символа. После нахождения данного символа разбиваем строку на две строки, получится "Имя1" до символа и "Имя2" после символа. Заранее создав двух людей, мы присваиваем к "Имя2"- "Имя1" как отца, если "Имя1" -мужчина или как мать в противном случае.

public void ParseLine(string s)

{

int x = s.IndexOf(" -> ");

if (x == -1) { throw new System.Exception("Строка не содержит' -> '."); }

string s1 = s.Substring(0, x);

string s2 = s.Substring(x + 4, s.Length - (x + 4));

Person p1 = CreatePerson(s1);

Person p2 = CreatePerson(s2);

if (p1.Sex == 'f') { p2.Mothers = p1; }

if (p1.Sex == 'm') { p2.Fathers = p1; }

}

private Person CreatePerson(string s) - Данный метод используется для создания нового экземпляра Person. Ему присваивается Имя и Пол, Пол определяется путем отделения от строки первых 3 символов (например"(м)") с помощью s.Substring(0,3). Имя определяется с помощью s.Substring(3,s.Length).

private Person CreatePerson(string s)

{

char sex;

string \_Name;

s = s.Trim();

switch (s.Substring(0, 3))// f/m

{

case "(m)": sex = 'm'; break;

case "(f)": sex = 'f'; break;

default: throw new System.Exception("Строка не содержит'f or m'.");

}

\_Name = s.Substring(3, s.Length - 3).Trim();// имя

Person p = FindPerson(\_Name);

if (p == null) { p = new Person(this, \_Name, sex); Person.AllPersons.Add(p); }

else if (p.Sex == sex) { return p; }

else { throw new System.Exception("Противоречивое имя " + \_Name); }

return p;

}

}