# Отчет по практической работе №11 «Абстрактные типы данных. Коллекции.»

## Задание

1. Реализовать обобщенную коллекцию, указанную в варианте.

**Для всех коллекций реализовать конструкторы:**

1. public MyCollection() - предназначен для создания пустой коллекции с заранее определенной начальной емкостью.
2. public MyCollection (int capacity*)* - создает пустую коллекцию с начальной емкостью, заданной параметром capacity.
3. public MyCollection (MyCollection c) - служит для создания коллекции, которая инициализируется элементами и емкостью коллекции, заданной параметром с.

**Для всех коллекций реализовать интерфейсы IEnumerable и IEnumertor.**

1. Написать демонстрационную программу, в которой создаются коллекции, и демонстрируется работа всех реализованных методов.

При работе с коллекцией использовать объекты из иерархии классов, разработанной в работе №10.

**Вариант 8:**

|  |  |  |  |
| --- | --- | --- | --- |
|  | Hashtable | Dictionary<K,T> | MyQueue<T> - очередь (на базе списка) |

Классы: студент, преподаватель, персона, сотрудник;

Запросы:

1. Имена служащих заданной профессии

2. Имена студентов указанного курса.

3. Имя самого старого человека

## 1. Анализ задачи

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **№** | **Описание функции** | **Входные данные** | **Классы входных данных** | | **Выходные данные** |
| **Основные функции** | | | | | |
| 1. | Создание массива класса «персона» | Целое число элементов, для каждого элемента имя персоны (строка), возраст (натуральное число), телефонный номер (пятизначное натуральное число), профессия (строка), зарплата (натуральное число) | |  | Массив класса «персона» |
| 2. | Вывод данных о персоне | Объект класса «персона» | |  | Имя персоны (строка), возраст (натуральное число), телефонный номер (пятизначное натуральное число), профессия (строка), зарплата (натуральное число) |
| 3. | Вывод всех студентов на выбранном курсе | Массив класса «персона», натуральный номер курса | | Персона с таким значением курса существует | Объект(ы) класса «персона» |
| Персона с таким значением курса не существует | Сообщение об ошибке |
| 4. | Вывод всех сотрудников на должности | Массив класса «персона», профессия (строка) | | Персона с таким значением профессии существует | Объект(ы) класса «персона» |
| Персона с таким значением профессии не существует | Сообщение об ошибке |
| 5. | Вывод самой старой персоны | Массив класса «персона» | | Массив пуст | Сообщение об ошибке |
| Массив не пуст | Объект класса «персона» |

## 2. Проектирование

|  |  |  |  |
| --- | --- | --- | --- |
| **№** | **Функция** | **Входные данные** | **Выходные данные** |
| **Основные функции** | | | |
| 1. | Part1 // Создание и вывод массива | Int n, Person [n] | Person [] persons, сообщение об ошибке |
| 2 | Part2 // Поиск в массиве | Person [n], int oper | Person person, сообщение об ошибке |
| 3 | Part3 // Демонстрация работы абстрактных методов | Person [3] | Person [] persons, сообщение об ошибке |
| 4 | ShowStudentsByCourse  // Вывод студентов выбранного курса | Person [] persons, int course | Person [] persons, сообщение об ошибке |
| 5 | ShowEmployeeByPost  // Вывод сотрудников на выбранной должности | Person [] persons, string post | Person [] persons, сообщение об ошибке |
| 6 | FindOldPerson  // Вывод старейшего человека | Person [] persons | Person person |

## 3. Листинг программы

using System;

using System.Collections;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Lab11

{

abstract class Person : IComparable

{

protected int birthYear;

protected int birthMonth;

protected int birthDay;

protected string name;

protected string surname;

protected int codeNumber;

public Person()

{

Console.WriteLine("Введите имя");

name = Console.ReadLine();

Console.WriteLine("Введите фамилию");

surname = Console.ReadLine();

Console.WriteLine("Введите год рождения");

birthYear = Program.ReadInteger(1900, 2018, "Введенное вами число не может быть номером года рождения. Пожалуйста, введите другое");

Console.WriteLine("Введите месяц рождения");

birthMonth = Program.ReadInteger(1, 12, "Введенное вами число не может быть номером месяца рождения. Пожалуйста, введите другое");

Console.WriteLine("Введите день рождения");

birthDay = Program.ReadInteger(1, 31, "Введенное вами число не может быть номером дня рождения. Пожалуйста, введите другое");

Console.WriteLine("Введите кодовый номер");

codeNumber = Program.ReadInteger(100000, 999999, "Введенное вами число не может являться кодовым номером. Пожалуйста, введите другое");

}

public Person(string Name, string Surname, int BirthYear, int BirthMonth, int BirthDay, int CodeNumber)

{

name = Name;

surname = Surname;

birthYear = BirthYear;

birthMonth = BirthMonth;

birthDay = BirthDay;

codeNumber = CodeNumber;

}

virtual public void Display()

{

Console.Write("Человек по имени ");

Console.Write(name);

Console.Write(" ");

Console.Write(surname);

Console.Write(", род. ");

Console.Write(birthDay);

Console.Write(".");

Console.Write(birthMonth);

Console.Write(".");

Console.WriteLine(birthYear);

Console.WriteLine("");

}

public string Name

{

get

{

return name;

}

set

{

name = value;

}

}

public string Surname

{

get

{

return surname;

}

set

{

surname = value;

}

}

public int CodeNumber

{

get

{

return codeNumber;

}

set

{

if ((value > 99999) && (value < 1000000))

{

codeNumber = value;

}

else

{

codeNumber = 100000;

}

}

}

public int CompareTo(object obj)

{

Person temp = (Person)obj;

if (String.Compare(this.name, temp.name) > 0) return 1;

if (String.Compare(this.name, temp.name) < 0) return -1;

return 0;

}

}

class Student : Person, ICloneable, IComparable

{

private double averageMark;

private string speciality;

private int year;

public Student() : base()

{

Console.WriteLine("Введите средний балл");

averageMark = Program.ReadDouble(0, 5, "Введенное вами число не может являться средним баллом студента. Пожалуйста, введите другое");

Console.WriteLine("Введите название специальности");

speciality = Console.ReadLine();

Console.WriteLine("Введите курс обучения");

year = Program.ReadInteger(1, 5, "Введенное вами число не может являться номером курса студента. Пожалуйста, введите другое");

}

public Student(string Name, string Surname, int BirthYear, int BirthMonth, int BirthDay, double AverageMark, int CodeNumber, string Speciality, int Year) : base(Name, Surname, BirthYear, BirthMonth, BirthDay, CodeNumber)

{

averageMark = AverageMark;

speciality = Speciality;

year = Year;

}

override public void Display()

{

Console.Write("Студент ");

Console.Write(name);

Console.Write(" ");

Console.Write(surname);

Console.Write(", род. ");

Console.Write(birthDay);

Console.Write(".");

Console.Write(birthMonth);

Console.Write(".");

Console.Write(birthYear);

Console.Write(", студент ");

Console.Write(year);

Console.Write(" курса специальности ''");

Console.Write(speciality);

Console.Write("'', средний балл - ");

Console.Write(averageMark);

Console.Write(", личный код - ");

Console.WriteLine(codeNumber);

Console.WriteLine("");

}

public object Clone()

{

return new Student(this.name, this.surname, this.birthYear, this.birthMonth, this.birthDay, this.averageMark, this.codeNumber, this.speciality, this.year);

}

public double AverageMark

{

get

{

return averageMark;

}

set

{

if ((value >= 0) && (value <= 5))

{

averageMark = value;

}

else

{

averageMark = 0;

}

}

}

}

class Staff : Person, ICloneable, IComparable

{

protected string occupation;

protected double salary;

public Staff() : base()

{

occupation = "Не указана";

salary = 0;

}

public Staff(string Name, string Surname, int birthYear, int birthMonth, int birthDay, int codeNumber, string Occupation, double Salary) : base(Name, Surname, birthYear, birthMonth, birthDay, codeNumber)

{

occupation = Occupation;

salary = Salary;

}

override public void Display()

{

Console.Write("Сотрудник ");

Console.Write(name);

Console.Write(" ");

Console.Write(surname);

Console.Write(", род. ");

Console.Write(birthDay);

Console.Write(".");

Console.Write(birthMonth);

Console.Write(".");

Console.Write(birthYear);

Console.Write(", должность - ");

Console.Write(occupation);

Console.Write(", зарплата - ");

Console.Write(salary);

Console.Write(", личный код - ");

Console.WriteLine(codeNumber);

Console.WriteLine("");

}

virtual public object Clone()

{

return new Staff(this.name, this.surname, this.birthYear, this.birthMonth, this.birthDay, this.codeNumber, this.occupation, this.salary);

}

}

class Teacher : Staff, ICloneable, IComparable

{

protected string speciality;

public Teacher() : base()

{

speciality = "Не указана";

}

public Teacher(string Name, string Surname, int birthYear, int birthMonth, int birthDay, int codeNumber, string Occupation, double Salary, string Speciality) : base(Name, Surname, birthYear, birthMonth, birthDay, codeNumber, Occupation, Salary)

{

speciality = Speciality;

}

override public void Display()

{

Console.Write("Преподаватель ");

Console.Write(name);

Console.Write(" ");

Console.Write(surname);

Console.Write(", род. ");

Console.Write(birthDay);

Console.Write(".");

Console.Write(birthMonth);

Console.Write(".");

Console.Write(birthYear);

Console.Write(" по предмету ''");

Console.Write(speciality);

Console.Write("'', зарплата - ");

Console.Write(salary);

Console.Write(", личный код - ");

Console.WriteLine(codeNumber);

Console.WriteLine("");

}

override public object Clone()

{

return new Teacher(this.name, this.surname, this.birthYear, this.birthMonth, this.birthDay, this.codeNumber, this.occupation, this.salary, this.speciality);

}

}

public class SortByName : IComparer

{

int IComparer.Compare(object ob1, object ob2)

{

Person s1 = (Person)ob1;

Person s2 = (Person)ob2;

return String.Compare(s1.Name, s2.Name);

}

}

class Program

{

public static int ReadInteger(int LowerLimit, int HigherLimit, string ErrorMessage)

{

int count;

bool confirmed;

do

{

confirmed = Int32.TryParse(Console.ReadLine(), out count);

if ((count < LowerLimit) || (count > HigherLimit) || (confirmed != true))

{

Console.WriteLine(ErrorMessage);

}

} while ((count < LowerLimit) || (count > HigherLimit) || (confirmed != true));

return count;

}

public static double ReadDouble(double LowerLimit, double HigherLimit, string ErrorMessage)

{

double count;

bool confirmed;

do

{

confirmed = Double.TryParse(Console.ReadLine(), out count);

if ((count < LowerLimit) || (count > HigherLimit) || (confirmed != true))

{

Console.WriteLine(ErrorMessage);

}

} while ((count < LowerLimit) || (count > HigherLimit) || (confirmed != true));

return count;

}

static ArrayList AddElement(ArrayList Collection)

{

int SwitchNumber, Position;

Console.WriteLine("Элемент какого типа вы хотите добавить в коллекцию?");

Console.WriteLine("1 - Студент");

Console.WriteLine("2 - Сотрудник");

Console.WriteLine("3 - Преподаватель");

SwitchNumber = ReadInteger(1, 3, "Введенное вами число не соответствует ни одному из предложенных вариантов. Пожалуйста, введите другое");

Console.WriteLine("В коллекции содержится всего {0} элементов. Введите, на какую позицию вы хотите добавить новый элемент?", Collection.Count);

Position = ReadInteger(1, Collection.Count + 1, "Введенное вами число не соответствует ни одной позиции. Пожалуйста, введите другое");

Collection.Capacity++;

Collection.Add(null);

for (int i = Collection.Count - 1; i > Position - 1; i--)

{

Collection[i] = Collection[i - 1];

}

switch (SwitchNumber)

{

case 1:

{

Collection[Position] = new Student();

break;

}

case 2:

{

Collection[Position] = new Staff();

break;

}

case 3:

{

Collection[Position] = new Teacher();

break;

}

}

return Collection;

}

static ArrayList RemoveElement(ArrayList Collection)

{

int Position, Code;

bool Found = false;

Console.WriteLine("Введите индивидуальный кодовый номер, по которому вы хотите удалить элемент из коллекции");

Code = ReadInteger(100000, 999999, "Введенное вами число не может являться значением кодового номера. Пожалуйста, введите другое");

for (int i = 0; i > Collection.Count - 1; i++)

{

Person Temp = (Person)Collection[i];

if (Temp.CodeNumber == Code)

{

Found = true;

Position = i;

}

if (Found == true)

{

Collection[i] = Collection[i + 1];

}

}

if (Found != true)

{

Console.WriteLine("Элемента с таким кодовым номером не существует");

}

Collection.RemoveAt(Collection.Count);

return Collection;

}

static void ShowCollection(ArrayList Collection)

{

for (int i = 0; i < Collection.Count; i++)

{

if (Collection[i] is Student)

{

Student Temp = (Student)Collection[i];

Temp.Display();

}

if ((Collection[i] is Staff) && (!(Collection[i] is Teacher)))

{

Staff Temp = (Staff)Collection[i];

Temp.Display();

}

if (Collection[i] is Teacher)

{

Teacher Temp = (Teacher)Collection[i];

Temp.Display();

}

}

}

static void ShowStudents(ArrayList Collection)

{

for (int i = 0; i < Collection.Count; i++)

{

if (Collection[i] is Student)

{

Student Temp = (Student)Collection[i];

Temp.Display();

}

}

}

static ArrayList DeleteWorst(ArrayList Collection)

{

Console.WriteLine("Введите балл, студентов с баллом ниже которого вы хотите удалить из коллекции?");

int Mark = ReadInteger(0, 5, "Введенное вами число не может являться средним баллом студента. Пожалуйста, введите другое");

for (int i = 0; i < Collection.Count; i++)

{

if (Collection[i] is Student)

{

Student Temp = (Student)Collection[i];

if (Temp.AverageMark < Mark)

{

Collection.Remove(Collection[i]);

}

}

}

return Collection;

}

static void CountWorkers(ArrayList Collection)

{

int Count = 0;

for (int i = 0; i < Collection.Count; i++)

{

if (Collection[i] is Staff)

{

Count++;

}

}

Console.WriteLine("Всего в коллекции {0} сотрудников университета", Count);

}

static ArrayList CloneCollection(ArrayList Collection)

{

ArrayList Temp = (ArrayList)Collection.Clone();

return Temp;

}

static ArrayList SortCollection(ArrayList Collection)

{

Collection.Sort();

return Collection;

}

static void Main(string[] args)

{

int SwitchNumber;

ArrayList Collection = new ArrayList(13);

ArrayList Clone;

Collection.Add(new Staff("Елена", "Кузнецова", 1990, 12, 27, 765234, "Уборщица", 31500));

Collection.Add(new Staff("Александр", "Александров", 1985, 1, 9, 452433, "Системный администратор", 55000));

Collection.Add(new Student("Петр", "Петров", 1999, 6, 11, 2.5, 123255, "Бизнес-информатика", 1));

Collection.Add(new Teacher("Ирина", "Калашникова", 1978, 3, 14, 887431, "Преподаватель", 75000, "Программирование"));

Collection.Add(new Student("Сергей", "Сергеев", 1995, 11, 11, 3.31, 123557, "История", 5));

Collection.Add(new Teacher("Алексей", "Алексеев", 1969, 7, 22, 439017, "Преподаватель", 95000, "Компьютерный практикум"));

Collection.Add(new Student("Макар", "Макаров", 1995, 11, 11, 3.31, 123572, "История", 5));

Collection.Add(new Student("Абдул", "Абдулов", 1995, 7, 23, 1.81, 123552, "История", 5));

Collection.Add(new Student("Денис", "Денисов", 1997, 8, 13, 5, 123553, "Программная инженерия", 3));

Collection.Add(new Student("Максим", "Максимов", 1996, 5, 30, 2.92, 123554, "Бизнес-информатика", 4));

Collection.Add(new Student("Матвей", "Матвеев", 1999, 6, 22, 3.8, 123890, "История", 1));

Collection.Add(new Student("Никита", "Никитин", 1999, 2, 19, 4.1, 123567, "Программная инженерия", 1));

Collection.Add(new Student("Иван", "Иванов", 1997, 12, 27, 4.33, 123907, "Программная инженерия", 3));

do

{

Console.WriteLine("Вас приветствует программа для работы с коллекцией. Что вы хотите сделать?");

Console.WriteLine("1 - Добавить новый элемент в коллекцию");

Console.WriteLine("2 - Удалить заданный элемент из коллекции");

Console.WriteLine("3 - Просмотреть коллекцию");

Console.WriteLine("4 - Вывести на экран список студентов в коллекции");

Console.WriteLine("5 - Удалить из коллекции всех студентов со средним баллом ниже заданного");

Console.WriteLine("6 - Посчитать кол-во сотрудников в коллекции");

Console.WriteLine("7 - Клонировать коллекцию");

Console.WriteLine("8 - Отсортировать коллекцию");

Console.WriteLine("9 - Завершить работу с программой");

SwitchNumber = ReadInteger(1, 9, "Введенное вами число не соответствует ни одному из предложенных вариантов. Пожалуйста, введите другое");

switch (SwitchNumber)

{

case 1:

{

Collection = AddElement(Collection);

break;

}

case 2:

{

Collection = RemoveElement(Collection);

break;

}

case 3:

{

ShowCollection(Collection);

break;

}

case 4:

{

ShowStudents(Collection);

break;

}

case 5:

{

Collection = DeleteWorst(Collection);

break;

}

case 6:

{

CountWorkers(Collection);

break;

}

case 7:

{

Clone = CloneCollection(Collection);

break;

}

case 8:

{

Collection = SortCollection(Collection);

break;

}

case 9:

{

break;

}

}

} while (SwitchNumber != 9);

}

}

}

## 4. Тестирование

1) Выбор операции (стартовое меню)

|  |  |  |  |
| --- | --- | --- | --- |
|  | *Входные данные* | *Ожидаемый результат* | *Результат* |
| **Т1** | u = 0 | Завершение работы с программой | Выход из программы |
| **Т2** | u = 3 | Демонстрация работы абстрактных методов | Выполнение подпрограммы 3 |
| **Т3** | u = “abc” | Натуральное число введено неверно. | Натуральное число введено неверно. Повторите ввод |
| **Т4** | u = 2.25 | Натуральное число введено неверно. | Натуральное число введено неверно. Повторите ввод |
| **Т5** | u = 1 | Создание и вывод массива | Выполнение подпрограммы 1 |
| **Т6** | u = 2 | Поиск в массиве | Выполнение подпрограммы 2 |
| **Т7** | u = -1 | Натуральное число введено неверно. | Натуральное число введено неверно. Повторите ввод |

2) Ввод числа элементов массива

|  |  |  |  |
| --- | --- | --- | --- |
|  | *Входные данные* | *Ожидаемый результат* | *Результат* |
| **Т1** | N = 0 | Натуральное число введено неверно. | Натуральное число введено неверно. Повторите ввод |
| **Т2** | N = -1 | Натуральное число введено неверно. | Натуральное число введено неверно. Повторите ввод |
| **Т3** | N = 2.25 | Натуральное число введено неверно. | Натуральное число введено неверно. Повторите ввод |
| **Т4** | N = “abc” | Натуральное число введено неверно. | Натуральное число введено неверно. Повторите ввод |
| **Т5** | N = 2 | Продолжение работы с программой | Продолжение работы с программой |

3) Ввод натурального числа (номер курса, зарплата, возраст, телефон)

|  |  |  |  |
| --- | --- | --- | --- |
|  | *Входные данные* | *Ожидаемый результат* | *Результат* |
| **Т1** | t = 0 | Продолжение работы с программой | Продолжение работы с программой |
| **Т2** | t = 5 | Продолжение работы с программой | Продолжение работы с программой |
| **Т3** | t = “abc” | Обнуление ввода. Продолжение работы с программой | Обнуление ввода. Продолжение работы с программой |
| **Т4** | t = 1.25 | Обнуление ввода. Продолжение работы с программой | Обнуление ввода. Продолжение работы с программой |
| **Т5** | t = -1 | Обнуление ввода. Продолжение работы с программой | Обнуление ввода. Продолжение работы с программой |