МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

Учреждение образование «Полоцкий государственный университет»

Факультет информационных технологий

Кафедра технологий программирования

**Лабораторная работа №6**

по дисциплине: «Объектно-ориентированные технологии

программирования и стандарты проектирования»

на тему: «Шаблоны классов»

Вариант №8 «Отдел кадров»

|  |  |
| --- | --- |
| Выполнил | Студент группы 21-ИТ-1  Макеёнок Д.И. |
|  |  |
| Проверил | Ассистент  Хирьянов И.Д. |

Полоцк 2023

**Цель работы:** научиться создавать шаблоны классов, применять шаблоны как для встроенных типов данных, так и для пользовательских классов, понимать особенности применения пользовательских классов в шаблонах.

**Ход работы**

**Задание:**

1. Дополнить и при необходимости модифицировать приложение, разработанное согласно варианта лабораторной работы №5.

2. Создать шаблон класса, в нем определить конструктор, в котором реализовано создание массива на заданное количество элементов n (количество передается в параметре конструктору).

3. В шаблоне определить метод добавления элемента массива и метод получения элемента массива по индексу.

4. В шаблоне определить метод int findItem(T obj), который ищет элемент в массиве и возвращает индекс элемента в массиве, а в случае неудачи возвращает -1. Для работы этого метода потребуется перегрузка оператора == в классах-наследниках.

5. В шаблоне определить метод T min() и метод T max(), которые возвращают минимальное и максимальное значение из объектов массива. Для работы этих методов потребуется перегрузка операторов сравнения в классах-наследниках. Сравнение объектов производить по любому из атрибутов класса.

6. В шаблоне определить метод void sort(), который должен отсортировать массив по возрастанию.

7. В основном теле программы применить шаблон для разных типов данных: int, char, каждый из классов-наследников.

8. Реализовать все необходимые проверки

9. Дополнить меню программы чтобы можно было продемонстрировать работу всех функций.

10.Сделать выводы.

**Вариант 8**

Тема проекта: приложение «Отдел кадров».

Спроектировать ПО, предназначенное для создания и просмотра информации по штатным работникам предприятия. Функции, которые должны быть реализованы в приложении: добавление, удаление, редактирование и просмотр работников, структуризация работников по отделам, детальная информация о работнике.

Листинг 1.1 - реализация меню программы:

using System.Data.Common;

namespace lab1;

class Program

{

static void Main()

{

int dep;

bool flag;

Department department = new Department();

Workers workers = new Workers();

while (true)

{

Console.WriteLine("Что делаем?\n" +

"1 - Добавить сотрудника\n" +

"2 - Удалить сотрудника\n" +

"3 - Просмотреть сотрудников\n" +

"4 - Редактировать сотрудника\n" +

"5 - Выдать сотруднику работу\n" +

"6 - Уволить сотрудника\n" +

"7 - Демонстарция операторов\n" +

"8 - Шаблоны\n");

int.TryParse(Console.ReadLine(), out int n);

switch (n)

{

case 1:

Console.WriteLine("В какой отдел добавить сотрудника?(1-3)");

while (true)

{

if (!(int.TryParse(Console.ReadLine(), out dep) && CorrectImput.InRange(1, 3, dep)))

{

Console.Write("Некоректный ввод, попробуй ещё раз: ");

continue;

}

dep--;

workers.NewEployment();

department.AddNewEployee(dep, workers);

Console.WriteLine("Успешно");

// Console.ReadKey();

Console.Clear();

break;

}

break;

case 2:

Console.WriteLine("Из какого отдела удалить сотрудника?(1-3)");

while (true)

{

if (!(int.TryParse(Console.ReadLine(), out dep) && CorrectImput.InRange(1, 3, dep)))

{

Console.Write("Некоректный ввод, попробуй ещё раз: ");

continue;

}

dep--;

flag = department.DeleteEployee(dep);

if (flag)

{

Console.WriteLine("Успешно");

break;

}

else

{

break;

}

}

// Console.ReadKey();

//Console.Clear();

break;

case 3:

Console.WriteLine("Какой отдел вывести?(1-3) / all - вывести всех");

while (true)

{

string tem = Console.ReadLine();

if (int.TryParse(tem, out dep))

{

if (!CorrectImput.InRange(1, 3, dep))

{

Console.WriteLine("Некоректный ввод, попробуй ещё раз: ");

continue;

}

dep--;

department.ShowEployee(dep);

break;

}

else

{

if (tem == "all")

{

department.ShowEployee();

break;

}

else

{

Console.WriteLine("Некоректный ввод, попробуй ещё раз: ");

continue;

}

}

}

//Console.ReadKey();

//Console.Clear();

break;

case 4:

Console.WriteLine("Из какого отдела редактируемый сотрудник?(1-3)");

while (true)

{

if (!(int.TryParse(Console.ReadLine(), out dep) && CorrectImput.InRange(1, 3, dep)))

{

Console.WriteLine("Некоректный ввод, попробуй ещё раз: ");

continue;

}

dep--;

flag = department.EditingEployee(dep);

if (flag)

{

Console.WriteLine("Успешно");

break;

}

else

{

break;

}

}

Console.ReadKey();

Console.Clear();

break;

case 5:

Console.WriteLine("Из какого отдела сотрудник?(1-3)");

while (true)

{

if (int.TryParse(Console.ReadLine(), out dep) && CorrectImput.InRange(1, 3, dep))

{

dep--;

Console.WriteLine("Введите номер сотрудника ");

if (!int.TryParse(Console.ReadLine(), out int number)) { break; }

flag = department.AddNewWorker(dep, number);

if (flag)

{

Console.WriteLine("Успешно");

break;

}

else

{

break;

}

Console.Clear();

break;

}

else Console.Write("Некоректный ввод, попробуй ещё раз: ");

}

break;

case 6:

Console.WriteLine("Из какого отдела сотрудник?(1-3)");

while (true)

{

if (int.TryParse(Console.ReadLine(), out dep) && CorrectImput.InRange(1, 3, dep))

{

dep--;

Console.WriteLine("Введите номер сотрудника ");

if (!int.TryParse(Console.ReadLine(), out int number)) { break; }

flag = department.DismissEmployee(dep, number);

if (flag)

{

Console.WriteLine("Успешно");

break;

}

else

{

break;

}

Console.Clear();

break;

}

else Console.Write("Некоректный ввод, попробуй ещё раз: ");

}

break;

case 7:

Console.Clear();

department.ShowOperators(department);

break;

case 8:

Template<int> intArr = new Template<int>(5);

Console.WriteLine("Введите числа");

for (int i = 0; i < intArr.GetArrayLength(); i++)

{

int.TryParse(Console.ReadLine(),out int number);

intArr.SetByIndex(i, number);

}

Template<char> charArr = new Template<char>(5);

Console.WriteLine("Введите символ");

for (int i = 0; i < charArr.GetArrayLength(); i++)

{

char.TryParse(Console.ReadLine(), out char symbol);

charArr.SetByIndex(i, symbol);

}

Template<Workers> workersArr = new Template<Workers>(5);

for (int i = 0; i < workersArr.GetArrayLength(); i++)

{

int startdate = 2000;

startdate += i;

DateTime birthday = new DateTime(1980, 1, 4);

Workers worker = new Workers(startdate, "Игорь", birthday, "Цискович", "работяга");

workersArr.SetByIndex(i, worker);

}

Template<Dismissed> dismissedArr = new Template<Dismissed>(5);

for (int i = 0; i < dismissedArr.GetArrayLength(); i++)

{

int startdate = 2000;

startdate += i;

DateTime birthday = new DateTime(1980, 1, 4);

DateTime dismissdate = new DateTime(2020, 12, 9);

Dismissed dismiss = new Dismissed(startdate, "Кирилл", birthday, "Беляцкий", dismissdate);

dismissedArr.SetByIndex(i, dismiss);

}

Template<int>.templateMenu(intArr, charArr, workersArr, dismissedArr);

break;

}

}

}

}

Листинг 1.2 - реализация главного класса:

using System;

using System.Data;

using System.Runtime.Intrinsics.Arm;

using System.Security.Cryptography.X509Certificates;

using System.Xml.Linq;

namespace lab1

{

class Department

{

List<Human>[] depart = new List<Human>[3];

public void AddNewEployee(int num, Workers hum)

{

depart[num].Add(new Workers(hum));

}

public Department()

{

depart[0] = new List<Human>();

depart[1] = new List<Human>();

depart[2] = new List<Human>();

}

public static Department operator +(Department department, Human human)

{

Console.WriteLine("В какой отдел добавить сотрудника?(1-3)");

while (true)

{

if (!(int.TryParse(Console.ReadLine(), out int dep) && CorrectImput.InRange(1, 3, dep)))

{

Console.Write("Некоректный ввод, попробуй ещё раз: ");

continue;

}

dep--;

department.depart[dep].Add(human);

return department;

}

}

public static Department operator ++(Department department)

{

Console.WriteLine("В какой отдел добавить сотрудника?(1-3)");

while (true)

{

if (!(int.TryParse(Console.ReadLine(), out int dep) && CorrectImput.InRange(1, 3, dep)))

{

Console.Write("Некоректный ввод, попробуй ещё раз: ");

continue;

}

dep--;

if (department.depart[dep].Count != 0) department.depart[dep].Add(department.depart[dep][0]);

else Console.WriteLine("Лист пуст");

return department;

}

}

public bool DeleteEployee(int num)

{

int count = 0;

string name = "";

bool flag = false, MoreOne = false;

if (depart[num].Count == 0)

{

Console.WriteLine("В этом отделе никого нет.");

return false;

}

Console.WriteLine("Фамилия удаляемого сотрудника:");

name = Console.ReadLine();

for (int i = 0; i < depart[num].Count; i++)

{

if (depart[num][i].GetSurename().IndexOf(name) != -1)

{

Console.Write(i + ".");

depart[num][i].Show();

count = i;

if (flag)

{

MoreOne = true;

}

flag = true;

}

}

if (MoreOne)

{

Console.Write("Сотрудника под каким номером нужно удалить?\n");

while (true)

{

if (int.TryParse(Console.ReadLine(), out int i))

{

depart[num].RemoveAt(i);

GC.Collect();

return true;

}

else Console.WriteLine("Некоректный ввод, попробуй ещё раз: ");

}

}

else

{

if (flag)

{

depart[num].RemoveAt(count);

GC.Collect();

return true;

}

else

{

Console.WriteLine("Неверная фамилия, или такой нет");

return false;

}

}

}

public void ShowEployee()

{

int num = 0;

for (int i = 0; i < 3; i++)

{

Console.WriteLine(++num + "-------------------------------");

for (int j = 0; j < depart[i].Count; j++)

{

Console.WriteLine(j + ".");

depart[i][j].Show();

}

if (depart[i].Count == 0) Console.WriteLine("Тут пусто\n");

}

}

public void ShowEployee(int num)

{

for (int i = 0; i < depart[num].Count; i++)

{

Console.WriteLine(i + ".");

depart[num][i].Show();

}

if (depart[num].Count == 0) Console.WriteLine("Тут пусто\n");

}

public bool EditingEployee(int num)

{

int count = 0;

string name = "";

bool flag = false, MoreOne = false;

if (depart[num].Count == 0)

{

Console.WriteLine("В этом отделе никого нет.");

return false;

}

Console.Write("Фамилия редактируемого сотрудника:");

name = Console.ReadLine();

for (int i = 0; i < depart[num].Count; i++)

{

if (depart[num][i].GetSurename() == name)

{

if (depart[num][i].GetSurename().IndexOf(name) != -1)

{

Console.Write(i + ".");

depart[num][i].Show();

count = i;

if (flag)

{

MoreOne = true;

}

flag = true;

}

}

}

if (MoreOne)

{

Console.Write("Сотрудника под каким номером нужно редактировать?\n");

while (true)

{

if (int.TryParse(Console.ReadLine(), out int i))

{

count = i;

break;

}

else Console.WriteLine("Некоректный ввод, попробуй ещё раз: ");

}

}

else

{

if (!flag)

{

Console.WriteLine("Неверная фамилия, или такой нет");

return false;

}

}

Console.WriteLine("Что нужно изменить?\n" +

"1 - Имя сотрудника\n" +

"2 - Фамилию сотрудника\n" +

"3 - День рождения сотрудника\n" +

"4 - Год начала работы сотрудника\n");

int edit = Convert.ToInt32(Console.ReadLine());

switch (edit)

{

case 1:

depart[num][count].AddHumanName();

return true;

case 2:

depart[num][count].AddHumanSurename();

return true;

case 3:

depart[num][count].AddHumanBirthday();

return true;

case 4:

depart[num][count].AddHumanStartYear();

return true;

}

return false;

}

public bool CheckEmployee(int dep)

{

if (depart[dep].Count > 0)

{

return true;

}

else return false;

}

public bool AddNewWorker(int dep, int number)

{

if (depart[dep].Count == 0)

{

Console.WriteLine("В этом отделе никого нет.");

return false;

}

Workers worker = new Workers(depart[dep][number].GetStartYear(), depart[dep][number].GetName(), depart[dep][number].Getbirthday(), depart[dep][number].GetSurename(), Workers.AddWorkersProfes());

depart[dep][number] = worker;

return true;

}

public bool DismissEmployee(int dep, int number)

{

if (depart[dep].Count == 0)

{

Console.WriteLine("В этом отделе никого нет.");

return false;

}

Dismissed dismiss = new Dismissed(depart[dep][number].GetStartYear(), depart[dep][number].GetName(), depart[dep][number].Getbirthday(), depart[dep][number].GetSurename(), Dismissed.AddDismissedProfes(depart[dep][number].GetStartYear()));

depart[dep][number] = dismiss;

return true;

}

public void ShowOperators(Department department)

{

bool menu = true;

int dep;

while (menu)

{

Workers workers = new Workers();

Console.WriteLine("Какой оператор используем?\n" +

"1 - оператор +\n" +

"2 - оператор ++\n" +

"3 - оператор ==\n" +

"4 - оператор !=\n" +

"5 - оператор <\n" +

"6 - оператор >\n" +

"7 - оператор <=\n" +

"8 - оператор >=\n" +

"9 - показать всё\n" +

"-1 - чтобы выйти\n");

int.TryParse(Console.ReadLine(), out int n);

switch (n)

{

case 1:

workers.NewEployment();

department += new Workers(workers);

break;

case 2:

department++;

break;

case 3:

Console.WriteLine("Из какого отдела сравниваемые сотрудники?(1-3)");

while (true)

{

if (int.TryParse(Console.ReadLine(), out dep) && CorrectImput.InRange(1, 3, dep))

{

dep--;

Console.WriteLine("Введите номер первого сотрудника ");

if (!int.TryParse(Console.ReadLine(), out int number1)) { break; }

Console.WriteLine("Введите номер второго сотрудника ");

if (!int.TryParse(Console.ReadLine(), out int number2)) { break; }

if (CorrectImput.InRange(0, depart[dep].Count() - 1, number1) && CorrectImput.InRange(0, depart[dep].Count() - 1, number2))

{

if (depart[dep][number1] == depart[dep][number2])

{

Console.WriteLine("Они равны ");

}

else Console.WriteLine("Они не равны ");

}

else Console.WriteLine("Сотрудников с такими номерами нет ");

break;

}

else Console.Write("Некоректный ввод, попробуй ещё раз: ");

}

break;

case 4:

Console.WriteLine("Из какого отдела сравниваемые сотрудники?(1-3)");

while (true)

{

if (int.TryParse(Console.ReadLine(), out dep) && CorrectImput.InRange(1, 3, dep))

{

dep--;

Console.WriteLine("Введите номер первого сотрудника ");

if (!int.TryParse(Console.ReadLine(), out int number1)) { break; }

Console.WriteLine("Введите номер второго сотрудника ");

if (!int.TryParse(Console.ReadLine(), out int number2)) { break; }

if (CorrectImput.InRange(0, depart[dep].Count() - 1, number1) && CorrectImput.InRange(0, depart[dep].Count() - 1, number2))

{

if (depart[dep][number1] != depart[dep][number2])

{

Console.WriteLine("Они не равны ");

}

else Console.WriteLine("Они равны ");

}

else Console.WriteLine("Сотрудников с такими номерами нет ");

break;

}

else Console.Write("Некоректный ввод, попробуй ещё раз: ");

}

break;

case 5:

Console.WriteLine("Из какого отдела сравниваемые сотрудники?(1-3)");

while (true)

{

if (int.TryParse(Console.ReadLine(), out dep) && CorrectImput.InRange(1, 3, dep))

{

dep--;

Console.WriteLine("Введите номер первого сотрудника ");

if (!int.TryParse(Console.ReadLine(), out int number1)) { break; }

Console.WriteLine("Введите номер второго сотрудника ");

if (!int.TryParse(Console.ReadLine(), out int number2)) { break; }

if (CorrectImput.InRange(0, depart[dep].Count() - 1, number1) && CorrectImput.InRange(0, depart[dep].Count() - 1, number2))

{

if (depart[dep][number1] < depart[dep][number2])

{

Console.WriteLine("Первый меньше второго ");

}

else Console.WriteLine("Первый не меньше второго ");

}

else Console.WriteLine("Сотрудников с такими номерами нет ");

break;

}

else Console.Write("Некоректный ввод, попробуй ещё раз: ");

}

break;

case 6:

Console.WriteLine("Из какого отдела сравниваемые сотрудники?(1-3)");

while (true)

{

if (int.TryParse(Console.ReadLine(), out dep) && CorrectImput.InRange(1, 3, dep))

{

dep--;

Console.WriteLine("Введите номер первого сотрудника ");

if (!int.TryParse(Console.ReadLine(), out int number1)) { break; }

Console.WriteLine("Введите номер второго сотрудника ");

if (!int.TryParse(Console.ReadLine(), out int number2)) { break; }

if (CorrectImput.InRange(0, depart[dep].Count() - 1, number1) && CorrectImput.InRange(0, depart[dep].Count() - 1, number2))

{

if (depart[dep][number1] > depart[dep][number2])

{

Console.WriteLine("Первый больше второго ");

}

else Console.WriteLine("Первый не больше второго ");

}

else Console.WriteLine("Сотрудников с такими номерами нет ");

break;

}

else Console.Write("Некоректный ввод, попробуй ещё раз: ");

}

break;

case 7:

Console.WriteLine("Из какого отдела сравниваемые сотрудники?(1-3)");

while (true)

{

if (int.TryParse(Console.ReadLine(), out dep) && CorrectImput.InRange(1, 3, dep))

{

dep--;

Console.WriteLine("Введите номер первого сотрудника ");

if (!int.TryParse(Console.ReadLine(), out int number1)) { break; }

Console.WriteLine("Введите номер второго сотрудника ");

if (!int.TryParse(Console.ReadLine(), out int number2)) { break; }

if (CorrectImput.InRange(0, depart[dep].Count() - 1, number1) && CorrectImput.InRange(0, depart[dep].Count() - 1, number2))

{

if (depart[dep][number1] <= depart[dep][number2])

{

Console.WriteLine("Первый меньше или равен второму ");

}

else Console.WriteLine("Первый не меньше или равен второму ");

}

else Console.WriteLine("Сотрудников с такими номерами нет ");

break;

}

else Console.Write("Некоректный ввод, попробуй ещё раз: ");

}

break;

case 8:

Console.WriteLine("Из какого отдела сравниваемые сотрудники?(1-3)");

while (true)

{

if (int.TryParse(Console.ReadLine(), out dep) && CorrectImput.InRange(1, 3, dep))

{

dep--;

Console.WriteLine("Введите номер первого сотрудника ");

if (!int.TryParse(Console.ReadLine(), out int number1)) { break; }

Console.WriteLine("Введите номер второго сотрудника ");

if (!int.TryParse(Console.ReadLine(), out int number2)) { break; }

if (CorrectImput.InRange(0, depart[dep].Count() - 1, number1) && CorrectImput.InRange(0, depart[dep].Count() - 1, number2))

{

if (depart[dep][number1] >= depart[dep][number2])

{

Console.WriteLine("Первый больше или равен второму");

}

else Console.WriteLine("Первый не больше или равен второму ");

}

else Console.WriteLine("Сотрудников с такими номерами нет ");

break;

}

else Console.Write("Некоректный ввод, попробуй ещё раз: ");

}

break;

case 9:

Console.WriteLine("Из какого отдела сравниваемые сотрудники?(1-3)");

while (true)

{

if (int.TryParse(Console.ReadLine(), out dep) && CorrectImput.InRange(1, 3, dep))

{

dep--;

Console.WriteLine("Введите номер первого сотрудника ");

if (!int.TryParse(Console.ReadLine(), out int number1)) { break; }

Console.WriteLine("Введите номер второго сотрудника ");

if (!int.TryParse(Console.ReadLine(), out int number2)) { break; }

if (CorrectImput.InRange(0, depart[dep].Count() - 1, number1) && CorrectImput.InRange(0, depart[dep].Count() - 1, number2))

{

bool check = depart[dep][number1] == depart[dep][number2];

Console.WriteLine($"human1 - {depart[dep][number1].GetStartYear()} == human2 - {depart[dep][number2].GetStartYear()} :" + check);

check = depart[dep][number1] != depart[dep][number2];

Console.WriteLine($"human1 - {depart[dep][number1].GetStartYear()} != human2 - {depart[dep][number2].GetStartYear()} :" + check);

check = depart[dep][number1] < depart[dep][number2];

Console.WriteLine($"human1 - {depart[dep][number1].GetStartYear()} < human2 - {depart[dep][number2].GetStartYear()} :" + check);

check = depart[dep][number1] > depart[dep][number2];

Console.WriteLine($"human1 - {depart[dep][number1].GetStartYear()} > human2 - {depart[dep][number2].GetStartYear()} :" + check);

check = depart[dep][number1] <= depart[dep][number2];

Console.WriteLine($"human1 - {depart[dep][number1].GetStartYear()} <= human2 - {depart[dep][number2].GetStartYear()} :" + check);

check = depart[dep][number1] >= depart[dep][number2];

Console.WriteLine($"human1 - {depart[dep][number1].GetStartYear()} >= human2 - {depart[dep][number2].GetStartYear()} :" + check);

}

else Console.WriteLine("Сотрудников с такими номерами нет ");

break;

}

else Console.Write("Некоректный ввод, попробуй ещё раз: ");

}

break;

case -1:

menu = false;

break;

}

}

}

}

}

Листинг 1.3 – реализация дополнительного класса

using System.Linq.Expressions;

using System.Security.Cryptography;

namespace lab1

{

abstract class Human : IComparable<Human>

{

protected string \_name;

protected string \_surename;

protected DateTime \_birthday;

protected int \_startYear;

public int CompareTo(Human other)

{

return GetStartYear().CompareTo(other.GetStartYear());

}

public override bool Equals(object obj)

{

if (obj == null || !this.GetType().Equals(obj.GetType()))

return false;

Human other = (Human)obj;

return \_startYear == other.\_startYear;

}

public override int GetHashCode()

{

return \_startYear.GetHashCode();

}

public Human()

{

\_startYear = 0;

\_name = "пусто";

\_surename = "пусто";

\_birthday = DateTime.Now;

Console.WriteLine("Вызван конструкотр без параметров");

}

public Human(int startYear, string name, DateTime birthday, string surename)

{

\_startYear = startYear;

\_name = name;

\_surename = surename;

\_birthday = birthday;

Console.WriteLine("Вызван конструкотр с параметрами");

}

public Human(Human human)

{

\_startYear = human.\_startYear;

\_name = human.\_name;

\_surename = human.\_surename;

\_birthday = human.\_birthday;

Console.WriteLine("Вызван конструкотр копирования");

}

~Human()

{

Console.WriteLine("Вызван деструктор");

}

public static bool operator ==(Human human1, Human human2)

{

if (human1.GetName() == human2.GetName() &&

human1.Getbirthday() == human2.Getbirthday() &&

human1.GetStartYear() == human2.GetStartYear() &&

human1.GetSurename() == human2.GetSurename() &&

human1.GetSurename() == human2.GetSurename())

return true;

else

return false;

}

public static bool operator !=(Human human1, Human human2)

{

if (human1.GetName() == human2.GetName() &&

human1.Getbirthday() == human2.Getbirthday() &&

human1.GetStartYear() == human2.GetStartYear() &&

human1.GetSurename() == human2.GetSurename() &&

human1.GetSurename() == human2.GetSurename())

return false;

else

return true;

}

public static bool operator >(Human human1, Human human2)

{

if (human1.GetStartYear() > human2.GetStartYear())

return true;

else

return false;

}

public static bool operator <(Human human1, Human human2)

{

if (human1.GetStartYear() < human2.GetStartYear()) return true;

else return false;

}

public static bool operator >=(Human human1, Human human2)

{

if (human1.GetStartYear() >= human2.GetStartYear())

return true;

else

return false;

}

public static bool operator <=(Human human1, Human human2)

{

if (human1.GetStartYear() <= human2.GetStartYear())

return true;

else

return false;

}

public string GetName()

{

return \_name;

}

public DateTime Getbirthday()

{

return \_birthday;

}

public int GetStartYear()

{

return \_startYear;

}

public string GetSurename()

{

return \_surename;

}

public void SetName(string name)

{

\_name = name;

}

public void Setbirthday(DateTime birthday)

{

\_birthday = birthday;

}

public void SetStartYear(int startyear)

{

\_startYear = startyear;

}

public void SetSurename(string surename)

{

\_surename = surename;

}

public void AddHumanName()

{

Console.Write("Введите имя сотрудника: ");

while (true)

{

string str = Console.ReadLine();

if (CorrectImput.IsLatters(str))

{

SetName(str);

break;

}

}

}

public void AddHumanSurename()

{

Console.Write("Введите фамилию сотрудника: ");

while (true)

{

string str = Console.ReadLine();

if (CorrectImput.IsLatters(str))

{

SetSurename(str);

break;

}

}

}

public void AddHumanBirthday()

{

DateTime birthday;

Console.WriteLine("Введите день рождения сотрудника: ");

while (true)

{

var chek = DateTime.TryParse(Console.ReadLine(), out birthday);

if (!chek)

{

Console.Write("Некоректный ввод, попробуй ещё раз: ");

continue;

}

break;

}

Setbirthday(birthday);

}

public void AddHumanStartYear()

{

Console.Write($"Введите год начала работы(2000 - {DateTime.Now.Year}): ");

while (true)

{

var chek = int.TryParse(Console.ReadLine(), out int year) && CorrectImput.InRange(2000, DateTime.Now.Year, year) && (year - 18) >= \_birthday.Year;

if (!chek)

{

Console.Write("Некоректный ввод, попробуй ещё раз: ");

continue;

}

SetStartYear(year);

break;

}

}

public void NewEployment()

{

AddHumanName();

AddHumanSurename();

AddHumanBirthday();

AddHumanStartYear();

}

public abstract void Show();

}

}

Листинг 1.4 – реализация наследственного класса Workers

namespace lab1

{

internal class Workers : Human

{

private string \_profession;

public Workers()

{

\_profession = "non";

Console.WriteLine("Вызван конструкотр без параметров наследовательного класса Workers");

}

public Workers(int startYear, string name, DateTime birthday, string surename, string profes) : base (startYear, name, birthday, surename)

{

\_profession = profes;

Console.WriteLine("Вызван конструкотр с параметрами наследовательного класса Workers");

Console.ReadKey();

}

public Workers(Workers workers): base(workers)

{

\_profession = workers.\_profession;

Console.WriteLine("Вызван конструкотр копирования наследовательного класса Workers");

}

~Workers()

{

Console.WriteLine("Вызван деструктор наследовательного класса Workers");

//Console.ReadKey();

}

public string GetProfession()

{

return \_profession;

}

public static string AddWorkersProfes()

{

Console.Write("Введите профессию сотрудника: ");

while (true)

{

string str = Console.ReadLine();

if (CorrectImput.IsLatters(str))

{

return str;

}

}

}

public override void Show()

{

Console.Write(

"Имя и Фамилия - " + GetName() + " " + GetSurename() + "\n" +

$"День рождения - {Getbirthday().ToString("D")}\n" +

"Год начала работы - " + GetStartYear() + "\n"+

"Профессия - " + \_profession + "\n" );

}

}

}

Листинг 1.5 – реализация наследственного класса Dismissed

namespace lab1

{

internal class Dismissed : Human

{

private DateTime \_dismissedDate;

public Dismissed()

{

\_dismissedDate = new DateTime();

Console.WriteLine("Вызван конструкотр без параметров наследовательного класса Dismissed");

}

public Dismissed(int startYear, string name, DateTime birthday, string surename, DateTime date) : base(startYear, name, birthday, surename)

{

\_dismissedDate = date;

Console.WriteLine("Вызван конструкотр с параметрами наследовательного класса Dismissed");

}

public Dismissed(Dismissed dismissed) : base(dismissed)

{

\_dismissedDate = dismissed.\_dismissedDate;

Console.WriteLine("Вызван конструкотр копирования наследовательного класса Dismissed");

}

~Dismissed()

{

Console.WriteLine("Вызван деструктор наследовательного класса Dismissed");

}

public DateTime GetDismissedDate()

{

return \_dismissedDate;

}

public static DateTime AddDismissedProfes(int startyear)

{

int day, month, year;

Console.WriteLine("Ввод Данных дня увольнения сотрудника\nВведите день(1 - 31): ");

while (true)

{

var chek = int.TryParse(Console.ReadLine(), out day) && CorrectImput.InRange(1, 31, day);

if (!chek)

{

Console.Write("Некоректный ввод, попробуй ещё раз: ");

continue;

}

break;

}

Console.Write("Введите месяц(1 - 12): ");

while (true)

{

var chek = int.TryParse(Console.ReadLine(), out month) && CorrectImput.InRange(1, 12, month);

if (!chek)

{

Console.Write("Некоректный ввод, попробуй ещё раз: ");

continue;

}

break;

}

Console.Write($"Введите год({startyear} - {DateTime.Now.Year}): ");

while (true)

{

var chek = int.TryParse(Console.ReadLine(), out year) && CorrectImput.InRange(startyear, DateTime.Now.Year, year);

if (!chek)

{

Console.Write("Некоректный ввод, попробуй ещё раз: ");

continue;

}

break;

}

return new DateTime(year, month, day);

}

public override void Show()

{

Console.Write(

"Имя и Фамилия - " + GetName() + " " + GetSurename() + "\n" +

$"День рождения - {Getbirthday().ToString("D")}\n" +

"Год начала работы - " + GetStartYear() + "\n" +

"Дата увольнения - " + \_dismissedDate.ToString("D") + "\n");

}

}

}

Листинг 1.6 – реализация проверки

namespace lab1

{

static class CorrectImput

{

public static bool IsLatters(string str)

{

foreach (char c in str)

{

if (!((c >= 'a' && c <= 'z') || (c >= 'A' && c <= 'Z') || (c >= 'а' && c <= 'я') || (c >= 'А' && c <= 'Я')))

{

Console.Write("Некоректный ввод, попробуй ещё раз: ");

return false;

}

}

return true;

}

public static bool InRange(int left, int right, int number)

{

if (number >= left && number <= right) return true;

return false;

}

}

}

Листинг 1.7 – реализация шаблонного класса

namespace lab1

{

class Template<T>

{

private T[] TArray;

private int ArrayLength;

public Template(int n)

{

TArray = new T[n];

ArrayLength = n;

}

public int GetArrayLength()

{

return ArrayLength;

}

public T GetElement(int n)

{

return TArray[n];

}

public T[] GetArray()

{

return TArray;

}

public T GetFromIndex(int index)

{

return TArray[index];

}

static public int operator ==(Template<T> template, T secondElement)

{

for (int i = 0; i < template.ArrayLength; i++)

{

if (Object.Equals(template.TArray[i], secondElement))

return i;

}

return -1;

}

static public int operator !=(Template<T> template, T secondElement)

{

for (int i = 0; i < template.ArrayLength; i++)

{

if (!Object.Equals(template.TArray[i], secondElement))

return i;

}

return -1;

}

static public bool templateMenu(Template<int> intArr, Template<char> charArr,

Template<Workers> workerArr, Template<Dismissed> dismissArr)

{

while (true)

{

Console.WriteLine("Что делаем?\n" +

"1 - Добавить элемент в массив\n" +

"2 - Достать элемент из массивов\n" +

"3 - Найти элемент\n" +

"4 - Вывести минимальные элементы массива\n" +

"5 - Вывести максимальные элементы массива\n" +

"6 - Отсортировать массив\n" +

"7 - Вывести массивы" +

"8 - выход");

if (int.TryParse(Console.ReadLine(), out int templateChoise))

{

switch (templateChoise)

{

case 1:

Console.WriteLine("В каком массиве заменить элемент?\n" +

"1 - Массив чисел\n" +

"2 - Массив символов\n" +

"3 - Массив работников\n" +

"4 - Массив уволенных\n");

if (int.TryParse(Console.ReadLine(), out int Choise))

{

Console.WriteLine("Номер элемента - ");

int.TryParse(Console.ReadLine(), out int index1);

switch (Choise)

{

case 1:

Console.WriteLine("Введите число - ");

int.TryParse(Console.ReadLine(), out int intElement);

intArr.SetByIndex(index1, intElement);

break;

case 2:

Console.WriteLine("Введите символ - ");

char.TryParse(Console.ReadLine(), out char charElement);

charArr.SetByIndex(index1, charElement);

break;

case 3:

Workers film = new Workers();

workerArr.SetByIndex(index1, film);

break;

case 4:

Dismissed blockedFilm = new Dismissed();

dismissArr.SetByIndex(index1, blockedFilm);

break;

}

}

break;

case 2:

int.TryParse(Console.ReadLine(), out int index2);

if (CorrectImput.InRange(0, intArr.GetArrayLength() - 1, index2))

{

Console.WriteLine(intArr.GetFromIndex(index2));

Console.WriteLine(charArr.GetFromIndex(index2));

workerArr.GetElement(index2).Show();

dismissArr.GetElement(index2).Show();

}

else

Console.WriteLine("Неправильный номер элемента");

break;

case 3:

Console.WriteLine("В каком массиве найти элемент?\n" +

"1 - Массив чисел\n" +

"2 - Массив символов\n" +

"3 - Массив работников\n" +

"4 - Массив уволенных\n");

if (int.TryParse(Console.ReadLine(), out int findChoise))

switch (findChoise)

{

case 1:

Console.WriteLine("Введите число");

int.TryParse(Console.ReadLine(), out int number);

Console.WriteLine(intArr.findItem(intArr, number));

break;

case 2:

Console.WriteLine("Введите символ");

char.TryParse(Console.ReadLine(), out char symbol);

Console.WriteLine(charArr.findItem(charArr, symbol));

break;

case 3:

Console.WriteLine(workerArr.findItem(workerArr, new Workers()));

break;

case 4:

Console.WriteLine(dismissArr.findItem(dismissArr, new Dismissed()));

break;

}

break;

case 4:

Console.WriteLine("Цифра - " + intArr.min() + "\n" +

"Символ - " + charArr.min() + "\n" +

"Работник -");

workerArr.min().Show();

Console.WriteLine("Уволенные -");

dismissArr.min().Show();

break;

case 5:

Console.WriteLine("Цифра - " + intArr.max() + "\n" +

"Символ - " + charArr.max() + "\n" +

"Работник -");

workerArr.max().Show();

Console.WriteLine("Уволенные -");

dismissArr.max().Show();

break;

case 6:

intArr.Sort();

charArr.Sort();

workerArr.Sort();

dismissArr.Sort();

Console.WriteLine("Все массивы отсортированны");

break;

case 7:

Template<int>.SeeArrays(intArr, charArr, workerArr, dismissArr);

break;

case 8:

return true;

}

}

}

}

public void SetByIndex(int index, T element)

{

if (TArray.Length == 0)

{

Console.WriteLine("Массив пуст");

return;

}

if (CorrectImput.InRange(0, TArray.Length - 1, index))

TArray[index] = element;

else

Console.WriteLine("номер элемента выходит за пределы массива");

}

public int findItem(Template<T> template, T obj)

{

return template == obj;

}

public void Sort()

{

List<T> list = new List<T>(TArray);

list.Sort();

TArray = list.ToArray();

}

public T max()

{

List<T> list = new List<T>(TArray);

list.Sort();

return list[list.Count - 1];

}

public T min()

{

List<T> list = new List<T>(TArray);

list.Sort();

return list[0];

}

static public void SeeArrays(Template<int> intArr, Template<char> charArr,

Template<Workers> favoriteFilmArray, Template<Dismissed> dismissArr)

{

Console.Write("Числа - ");

for (int i = 0; i < intArr.GetArrayLength(); i++)

{

Console.Write(intArr.GetFromIndex(i) + ", ");

}

Console.WriteLine("");

Console.Write("Символы - ");

for (int i = 0; i < charArr.GetArrayLength(); i++)

{

Console.Write(charArr.GetFromIndex(i) + ", ");

}

Console.WriteLine("");

Console.WriteLine("Работники - ");

for (int i = 0; i < favoriteFilmArray.GetArrayLength(); i++)

{

favoriteFilmArray.GetFromIndex(i).Show();

Console.WriteLine("");

}

Console.WriteLine("Уволенные - ");

for (int i = 0; i < dismissArr.GetArrayLength(); i++)

{

dismissArr.GetFromIndex(i).Show();

Console.WriteLine("");

}

}

}

}

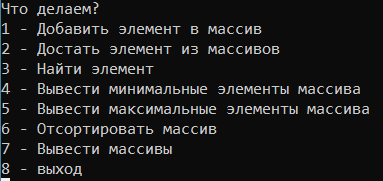


Рисунок 1. Меню шаблона

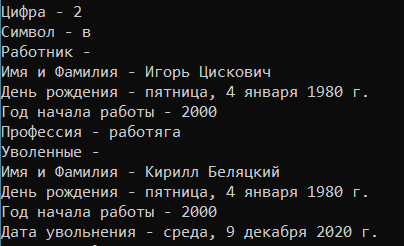


Рисунок 2. Вывод минимальных элементов массивов

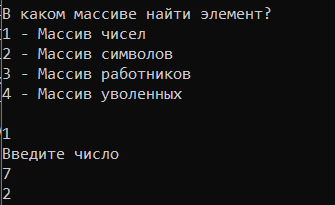


Рисунок 3. Поиск элементов

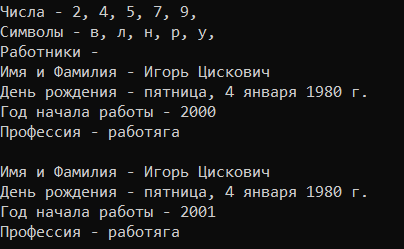


Рисунок 4. Отсортированные массивы

**Вывод:** в ходе лабораторной работы мы научились создавать шаблоны классов, применять шаблоны как для встроенных типов данных, так и для пользовательских классов, понимать особенности применения пользовательских классов в шаблонах.