



МИНОБРНАУКИ РОССИИ
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«МИРЭА – Российский технологический университет»
РТУ МИРЭА

Институт комплексной безопасности и специального приборостроения
Кафедра КБ-4 «Интеллектуальные системы информационной безопасности»

ОТЧЁТ ПО УЧЕБНОЙ ПРАКТИКЕ

Практика по получению первичных профессиональных умений и навыков, в том числе первичных умений и навыков научно-исследовательской деятельности

Тема практики: «Создание информационной системы на языке C#»

приказ Университета о направлении на практику от «08» февраля 2021 г. № 276-У

Отчет представлен к
рассмотрению:

Пашев А.С

Студент группы:

«__» ____ 2021 г.

(Подпись)

Отчет утвержден.
Допущен к защите:

Руководитель практики от
кафедры

«__» ____ 2021 г.

(Подпись)

Сачков В.Е.

Москва 2021 г.



МИНОБРНАУКИ РОССИИ
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«МИРЭА – Российский технологический университет»
РТУ МИРЭА

Институт комплексной безопасности и специального приборостроения
Кафедра КБ-4 «Интеллектуальные системы информационной безопасности»

ИНДИВИДУАЛЬНОЕ ЗАДАНИЕ НА УЧЕБНУЮ ПРАКТИКУ
Практика по получению первичных профессиональных умений и
навыков, в том числе первичных умений и навыков научно-
исследовательской деятельности

Студенту 1 курса учебной группы БСБО-04-20
Пашеву Антону Сергеевичу

Место и время практики: РТУ МИРЭА, кафедра КБ-4 «Интеллектуальные системы
информационной безопасности», с «09» февраля 2021 г. по «31» мая 2021 г.

Должность на практике: студент

1. ЦЕЛЕВАЯ УСТАНОВКА: развитие способностей в области анализа и моделирования прикладных процессов с учетом выбранной темы исследования

2. СОДЕРЖАНИЕ ПРАКТИКИ:

2.1. Изучить: исследовать информационные и прикладные процессы
2.2. Практически выполнить: применить современные инструментальные средства для моделирования информационных и прикладных процессов

2.3. Ознакомиться: с уровнем развития информационных и прикладных процессов с учетом темы исследования

3. ДОПОЛНИТЕЛЬНОЕ ЗАДАНИЕ: оформить отчет

4. ОРГАНИЗАЦИОННО-МЕТОДИЧЕСКИЕ УКАЗАНИЯ: в процессе практики рекомендуется использовать издания и отраслевую литературу годом издания не старше 5 лет

Заведующий кафедрой:

«09» февраля 2021 г.

(Магомедов Ш.Г.)

(подпись)

СОГЛАСОВАНО:

Руководитель практики от кафедры
«09» февраля 2021 г.

Сачков В.Е.

(подпись)

Задание получил
«09» февраля 2021 г.

Пашев А.С

(подпись)

**Проведенные
инструктажи:**

Охрана труда:

«09» февраля 2021 г.

Инструктирующий

(подпись)

(Сачков В.Е., ст.
преподаватель кафедры
КБ-4)

Инструктируемый

(подпись)

Пашев А.С

Техника безопасности:

«09» февраля 2021 г.

Инструктирующий

(подпись)

(Сачков В.Е., ст.
преподаватель кафедры
КБ-4)

Инструктируемый

(подпись)

Пашев А.С

Пожарная безопасность:

«09» февраля 2021 г.

Инструктирующий

(подпись)

(Сачков В.Е., ст.
преподаватель кафедры
КБ-4)

Инструктируемый

(подпись)

Пашев А.С

С правилами внутреннего распорядка ознакомлен:

«09» февраля 2021 г.

Пашев А.С

(подпись)



МИНОБРНАУКИ РОССИИ
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«МИРЭА – Российский технологический университет»
РТУ МИРЭА

**РАБОЧИЙ ГРАФИК ПРОВЕДЕНИЯ
УЧЕБНОЙ ПРАКТИКИ**

студента Пашева Антона Сергеевича 1 курса группы БСБО-04-20 очной формы обучения, обучающегося по направлению 09.03.02 Информационные системы и технологии профиль «Технологии искусственного интеллекта в безопасности»

Неделя	Сроки выполнения	Этап	Отметка о выполнении
1	«09» февраля 2021 г. - «09» марта 2021 г.	Подготовительный этап, включающий в себя организационное собрание (Вводная лекция о порядке организации и прохождения	
2	«10» марта 2021 г. – «30» апреля 2021 г.	Выполнение задания по практике в соответствии с выданным заданием студента. (Мероприятия по сбору, обработке и структурированию	
3	«01» мая 2021 г. – «17» мая 2021 г.	Подготовка отчета по практике (Оформление материалов отчета в полном соответствии с требованиями на	
4	«20» мая 2021 г.	Защита отчета по учебной практике у руководителя практики. (Представление	

Согласовано:

Заведующий кафедрой _____/Магомедов Ш.Г., к.т.н., доцент/

Руководитель практики от
кафедры _____/ Сачков В.Е., ст. преподаватель /

Обучающийся _____/Пашев А.С/

ОТЧЁТ
по учебной практике

студента 1 курса учебной группы БСБО-04-20 института КБСП
Пашева Антона Сергеевича

1. Практику проходил с 09.02.2021 г. по 31.05.2021 г. в ФГБОУ ВО «МИРЭА – Российский технологический университет», на кафедре КБ-4 «Интеллектуальные системы информационной безопасности», студент _____

(место прохождения практики и должность)

2. Задание на практику выполнил
в полном объеме

(указать: в полном объеме или частично)

Не выполнены следующие задания:

(указать также причины невыполнения)

Подробное содержание выполненной на практике работы и достигнутые результаты:
проведено исследование прикладной области в части изучения

Предложения по совершенствованию организации и прохождения практики:
предложений нет

Студент _____ (Пашев А.С.)
(подпись)

«__» _____ 20__ г

Заключение руководителя практики

Приобрел следующие профессиональные навыки: студент продемонстрировал профессиональные умения и навыки, знание и понимание прикладной области, задач, требующих решения в прикладной области, современные подходы и средства решения прикладных задач разных классов, умение находить и работать с различными источниками информации по профессиональной деятельности, структурировать отчет с учетом тематики исследования

Проявил себя как: дисциплинированный ответственный специалист: соблюдал сроки календарного графика практики, регулярно отчитывался о проделанных этапах работ; за срок прохождения практики не получил ни одного замечания - проявляет инициативу, четко и в определенные сроки выполняет задания; в любой ситуации уважителен в общении с другими.

«__» _____ 20__ г

Отчет проверил:

Руководитель практики от Университета
_____ (Сачков В.Е.) (подпись)

Содержание

Задание.....	7
ДОП. ЗАДАНИЕ:.....	8
Реализация задания	9
Список использованных источников	25

Задание

Программа «Симулятор жизни уток на озере “Большая охота”»

Долина диких озер прекрасное место для обитания множества разных видов и мастей уток. Все утки очень хвастливы и любят **рассказывать все о себе**, когда их спросят. Озеро в долине как живой организм знает **кто на нем живет, что кто умеет и может об этом рассказать** (сколько уток всего, сколько умеет летать/плавать и т. д.). Но иногда в долину приходят **охотники**, на некоторые дикие озера, которые выбирают случайно и ловят уток, которые там живут **один раз в день в течение сезона охоты**. Пойманных уток отвозят на домашнее озеро на ферме. Но уткам не нравится ферма, и каждая пойманная утка пытается вернуться на свое родное озеро. Если утка сбежала с фермы и ее опять поймали и привезли на ту же самую ферму, то ей подрезают крылья (если летает) или вешают груз на лапку (если плавает), и они больше не могут сбежать. Также в долине каждый день может произойти, что-нибудь необычное...

Охотники ловят уток до тех пор, пока на диких озерах не останется ни одной утки, либо не закончится сезон охоты.

ВАРИАНТ#24

В долине озер: 4, уток там живет: 44, а ферм: 1,
дней сезона охоты: 8

На Озере 'Байкал' живут следующие виды уток:

Утка вид: 'Чернети', умеет: летать и не знает где она живет, а также имеет атрибуты: имя, вес, высота, форма лап

Утка вид: 'Чернети', умеет: плавать и не знает где она живет, а также имеет атрибуты: имя, вес, возраст, пол

На Озере 'Кратерное Озеро' живут следующие виды уток:

Утка вид: 'Мраморные чирки', умеет: плавать и не знает где она живет, а также имеет атрибуты: имя, вес, окрас крыльев, форма крыльев

Утка вид: 'Широконоска', умеет: летать и знает где она живет, а также имеет атрибуты: имя, вес, цвет глаз, цвет

На Озере 'Челан' живут следующие виды уток:

Утка вид: 'Широконоска', умеет: летать и не знает где она живет, а также имеет атрибуты: имя, вес, форма крыльев, ширина

Утка вид: 'Крохали', умеет: летать и не знает где она живет, а также имеет атрибуты: имя, вес, окрас крыльев, высота

На Озере 'Ньяса' живут следующие виды уток:

Утка вид: 'Чернети', умеет: плавать и знает где она живет, а также имеет атрибуты: имя, вес, ширина, здоровье

Утка вид: 'Каменушки', умеет: плавать и знает где она живет, а также имеет атрибуты: имя, вес, форма клюва, любимое блюдо

Ферма 'ВОНІFUT' имеет охотников в

количестве: 1 Охотник #1 может

поймать уток в количестве: 1-8

С фермы 'ВОНІFUT' могут сбежать утки, которые умеют: летать и не знают где они живут, то есть возвращаются в случайное озеро

ДОП. ЗАДАНИЕ:

В долине в случайном месте (озере или ферме) может появиться: НитроХряк (Кабанчик, уничтожает место где появился, себя и всех кто там есть) на дней: 2

Реализация задания

Задание выполнено на языке программирования C#, в среде разработки Microsoft Visual Studio. Для его реализации потребовалось подключение пространств имен – System. Реализация программы происходила в пространстве имён – namespace Program. Исходный код см. в Приложении.

Исходя из условия задачи, утки обладают перечнем параметров и умений. Поэтому удобно создать абстрактный класс Duck, который будет содержать переменные, которые будут являться параметрами уток и метод voice, который будет отвечать за описание определённой утки. Помимо этого необходимо создать конструктор этого класса, который будет принимать параметр id (номер утки) по ссылке и затем увеличивать его на единицу при создании следующей утки.

Листинг 1 – Абстрактный класс Duck

```
abstract public class Duck
{
    public string type = "";
    public bool swim = false;
    public bool knowhome = false;
    public bool fly = false;
    public string home = "";
    public int ID = 0;
    public bool cut;
    public int caught;
    public Duck(ref int id)
    {
        ID = id;
        id++;
    }

    public abstract void voice();
}
```

Далее для удобства необходимо создать статический класс par, в котором описаны методы генерации параметров уток, предварительно подключив к нему класс Random для генерации числовых параметров уток (ширина, возраст, вес и тд.). Например метод genscolour(), возвращает цвет утки из массива содержащего строки(цвета уток), а метод genage() просто возвращает возраст утки при помощи метода Next класса Random. Полностью класс par см. в приложении.

Листинг 2 – Статический класс par(частично)

```
public static class par
{
    static Random random = new Random();
    public static string gencolour()
    {
        var colour = new string[] { "красный", "голубой", "зелёный",
"жёлтый", "белый", "чёрный" };
        return colour[random.Next(0, 6)];
    }
    public static int genage()
    {
        return random.Next(1, 9);
    }
}
```

Теперь создаём типы уток в виде классов, которые будут наследоваться от абстрактного класса Duck. На примере типа Chernety1 (Чернети 1) рассмотрим описание кода типа уток. Создаём конструктор класса Chernety1, который будет наследовать от класса Duck следующие переменные: type (тип утки), home (дом утки), fly (умеет ли она летать), swim (умеет ли она плавать), cut (подрезана ли она), count_cut (переменная которая определяет надо ли подрезать утку или нет). Также этот класс наследует по ссылке уникальный номер утки (id). Далее создаём переменные, которые принимают значения параметров, сгенерированные в абстрактном классе par. Для каждого вида утки это индивидуальные параметры в случае для класса Chernety1 это – вес, форма лап, высота и имя. И переопределяем метод voice() от абстрактного класса Duck, который будет выводить данные об утке. По аналогии проделываем те же действия для каждого вида утки, опираясь на условия. Все виды уток см. в Приложении.

Листинг 3 – Тип уток(Chernety1)

```
class Chernety1 : Duck
{
    public Chernety1(ref int id) : base(ref id)
    {
        type = "Чернети 1";
        home = "Байкал";
        fly = true;
        swim = false;
        knowhome = false;
        cut = false;
    }
}
```

```

    }

    int weight = par.genweight();
    string paws = par.genpaws();
    int height = par.genheight();
    string name = par.genname();

    public override void voice()
    {
        Console.WriteLine("-----");
        string a = $"Имя: {name}\nТип утки: {type}\nВес: {weight}\nВысота:
{height}\nФорма лап: {paws}\n";
        if (fly) a += "Умеет летать\n";
        if (swim) a += "Умеет плавать\n";
        if (knowhome) a += $"Живёт на озере {home}\n";
        if (cut == true)
        {
            a += "Подрезана";
        }
        else
        {
            a += "Не подрезана";
        }
        Console.WriteLine(a);
        Console.WriteLine("-----");
    }
}

```

Далее создаём класс озера Lake, который будет содержать в себе пустой массив типа уток, который будет содержать в себе экземпляры типа той или иной утки `Duck [] lake`, размерность озера `size`, разрушено озеро или нет булевая переменная `lake_live`, имя `name`, и количество дней НитроХряка(после разрушения). Создаём конструктор, который будет принимать имя, кол-во дней НитроХряка и переменную `lake_live`. После создаём метод `add_ducks`, который будет принимать экземпляры класса `Duck` и увеличивать размерность пустого массива `lake`, добавляя в него экземпляры класса `Duck`, метод `Lakevoice(int n)`, который будет принимать номер утки для дальнейших действий, и выдавать нам число суммарное количество уток на нем, а также количество уток по умениям, предварительно проверив целостность озера, цикл, который будет подсчитывать плавающих и летающих уток (по условию) и метод `remove(int num)`, который будет убирать утку с позиции `num`, уменьшать массив `lake` на 1 и возвращать экземпляр типа `Duck`.

Листинг 4 – Класс Lake

```
class Lake
{
    Random random = new Random();

    public Duck[] lake = new Duck[0];
    int size = 0;
    public bool lake_live;
    public string name;
    public int days_Nitro;
    public Lake(string name)
    {
        this.name = name;
        this.lake_live = true;
        this.days_Nitro = 1;
    }

    public void add_ducks(Duck duck)
    {
        Array.Resize(ref lake, size + 1);
        lake[size] = duck;
        size++;
    }

    virtual public void Lakevoice(int n)
    {
        int swim = 0;
        int fly = 0;

        for (int i = 0; i < size; i++)
        {
            if (lake[i].swim == true) swim++;
            if (lake[i].fly == true) fly++;
        }

        if (lake_live == true)
        {
            Console.WriteLine("-----");
            Console.WriteLine($"Озеро {name}");
            if (size > 0)
            {
                Console.WriteLine($"Всего уток: {size}");
                if (swim > 0) Console.WriteLine($"Умеют плавать: {swim}");
                if (fly > 0) Console.WriteLine($"Умеют летать: {fly}");
            }
            else
            {
                Console.WriteLine("Уток нет");
            }
        }
        else
        {
            Console.WriteLine("-----\nОзеро уничтожено!");
        }
        Console.WriteLine("-----");
    }

    public Duck remove(int num)
```

```

    {
        Duck duck = lake[num];
        while (num < size - 1)
        {
            lake[num] = lake[num + 1];
            num++;
        }
        Array.Resize(ref lake, size - 1);
        size--;

        return duck;
    }
}

```

Далее создадим класс `Farm` наследованный от класса `Lake`, который будет наследовать имя в конструкторе, и иметь параметры количество дней НитроХряка и целостность озера. Метод `Lakevoice(int n)` реализован почти точно также с единственной разницей в выводе. Также этот класс содержит метод `escape`, который получает на вход массив уток, которые находятся на ферме, проверяет могут ли они сбежать оттуда и, если могут, то уменьшаем массив уток, удаляем эту утку с позиции и подрезаем её если она уже улетала.

Листинг 5 – Класс `Farm`

```

class Farm : Lake
{

    public Farm(string name) : base(name)
    {
        this.name = name;
        this.lake_live = true;
        this.days_Nitro = 1;
    }

    public override void Lakevoice(int n)
    {
        int swim = 0;
        int fly = 0;
        for (int i = 0; i < lake.Length; i++)
        {
            if (lake[i].swim == true) swim++;
            if (lake[i].fly == true) fly++;
        }
        Console.WriteLine("-----");
        if (lake_live == true)
        {
            Console.WriteLine($"Ферма {name}");

            if (lake.Length > 0)

```

```

        {
            Console.WriteLine($"Всего уток: {lake.Length}");
            if (swim > 0) Console.WriteLine($"Умеют плавать: {swim}");
            if (fly > 0) Console.WriteLine($"Умеют летать: {fly}");
        }

        else
        {
            Console.WriteLine("Уток нет");
        }
        Console.WriteLine("-----");
    }
    else
    {
        Console.WriteLine($"Ферма {this.name} с охотниками
уничтожена");
    }
}

public bool escape(ref Duck[] duck)
{
    int i = 0;
    int arrl = 0;
    bool freeducks = false;
    while (i < lake.Length)
    {
        if (lake[i].fly == true && lake[i].knowhome == false &&
lake[i].cut == false)
        {
            Array.Resize(ref duck, arrl + 1);
            duck[arrl] = remove(i);
            if (duck[arrl].count_cut == 1)
            {
                duck[arrl].cut = true;
                duck[arrl].fly = false;
            }

            arrl++;

            freeducks = true;

        }
        else i++;
    }
    return freeducks;
}
}

```

После этого занимаемся реализацией основного класса `Simulyator_ducks`. Для начала создадим три статических метода, которые будут принимать в себя озёра и выдавать соответствующий вывод. Метод `showLakes(Lake[] lake)` будет вызывать метод `Lakevoice(int n)` для каждого озера и соответственно выдаёт информацию о нём. Метод `showLake(Lake[] lake)` отличается от предыдущего лишь тем, что вызывает метод `Lakevoice (int n)` определённого

озера. Метод `showDuck(Lake[] lake)` вызывает метод `voice()` у утки под номером `duck_id`, на озере под номером `lake_id`.

Листинг 6 – Методы показа различных параметров

```
static void showLakes(Lake[] lake)
{
    for (int i = 0; i < 5; i++)
    {
        lake[i].Lakevoice(i);
    }
    Console.WriteLine("-----");
}

static void showLake(Lake[] lake)
{
    Console.Write("Введите номер озера: ");
    int lake_id = Convert.ToInt32(Console.ReadLine());

    lake[lake_id].Lakevoice(lake_id);

    Console.WriteLine("-----");
}

static void showDuck(Lake[] lake)
{
    Console.Write("Введите номер озера: ");
    int lake_id = Convert.ToInt32(Console.ReadLine());
    Console.Write("Введите номер утки: ");
    int duck_id = Convert.ToInt32(Console.ReadLine());

    lake[lake_id].lake[duck_id].voice();

    Console.WriteLine("-----");
}
```

Далее создаём основной метод `Main()`, который будет содержать основной код программы. Для начала объявим переменные: `ducks`(суммарное количество уток), `id` (уникальный номер каждой утки начиная с 1), `hunt_days` (количество дней охоты), `count_lakes`(количество озёр в долине). Потом создаём экземпляры `Lakes` (учитываем, что класс `Farm` наследуется от класса `Lake`). Затем в цикле `for` заполняем озёра случайным количеством уток, предварительно выбрав типы уток, которые могут там жить.

Листинг 7 – Описание переменных и заполнение озёр утками

```
Random random = new Random();
int ducks = 44;
int id = 1;
int hunt_days = 8;
int count_lakes = 4;
```

```

var lake = new Lake[5];
lake[0] = new Lake("Байкал");
lake[1] = new Lake("Кратерное озеро");
lake[2] = new Lake("Челан");
lake[3] = new Lake("Ньяса");

lake[4] = new Farm("BOHIFUT");

for (int i = 0; i < ducks; i++)
{
    switch (random.Next(1, 5))
    {
        case 1:
        {
            if (random.Next(0, 2) == 0)
            {
                lake[0].add_ducks(new Chernety1(ref id));
            }
            else
            {
                lake[0].add_ducks(new Chernety2(ref id));
            }
        }
        break;
        case 2:
        {
            if (random.Next(0, 2) == 0)
            {
                lake[1].add_ducks(new Mramchir(ref id));
            }
            else
            {
                lake[1].add_ducks(new Shirok1(ref id));
            }
        }
        break;
        case 3:
        {
            if (random.Next(0, 2) == 0)
            {
                lake[2].add_ducks(new Shirok2(ref id));
            }
            else
            {
                lake[2].add_ducks(new Krohali(ref id));
            }
        }
        break;
        case 4:
        {
            if (random.Next(0, 2) == 0)
            {
                lake[3].add_ducks(new Chernety3(ref id));
            }
            else
            {
                lake[3].add_ducks(new Stoneduck(ref id));
            }
        }
    }
}

```



```

        }
        break;
    }
}

```

Далее запускаем цикл, который будет идти 8 раз (количество дней охоты). После добавляем определённые переменные для НитроХряка(Nitro – цифра, при которой НитроХряк появится на определённом озере; num_Nitro – номер озера на котором может появиться НитроХряк), а также переменные для подсчета суммарного количества уток (sumducks) и номера озера, на которое отправится охотник(randomlake). Затем делаем проверку уток на озёрах, если их нет – то программа выдаст сообщение об этом и выйдет из цикла, завершив программу.

Листинг 8 – Описание переменных и проверка уток на наличие

```

int Nitro = 2;
    int sumducks = 0;
    int num_Nitro = random.Next(count_lakes + 1);
    int hunt_ducks = random.Next(1, 9);
    int randomlake = random.Next(count_lakes);
for (int s = 0; s < count_lakes; s++)
{
    sumducks += lake[s].lake.Length;
} // Chech sumducks
if (sumducks == 0)
{
    Console.WriteLine("Уток на диких озёрах больше нет!");
    break;
}

```

После необходимо проверить озеро на его целостность. Если прошло два дня с момента появления НитроХряка на озере, то нужно изменить переменную lake_live на true, если же нет, то уменьшить количество дней на один с момента появления кабанчика. Затем надо запустить бесконечный цикл while, в котором кабанчик делает проверку на целостность озера. Если оно целое (озеро) и генератор случайных чисел выдаёт число кабанчика, то выводится сообщение, что кабанчик прибыл на озеро удаляет всех уток при помощи метода remove(int num), изменяет переменную lake_live на false и изменяется количество дней НитроХряка на озере и происходит выход из цикла, если же озеро разрушено, переменная изменяется следующим образом

(изменяется номер озера) : num_Nitro = random.Next(count_lakes + 1) и опять делается проверка на число кабанчика в генераторе.

Листинг 9 – Реализация НитроХряка

```
for (int p = 0; p < count_lakes + 1; p++)
{
    if (lake[p].lake_live == false)
    {
        if (lake[p].days_Nitro == 0)
        {
            lake[p].lake_live = true;
        }
        else
        {
            lake[p].days_Nitro--;
        }
    }
} //NitroCrack
while (true)
{
    if (lake[num_Nitro].lake_live == true)
    {
        if (random.Next(0, 4) == Nitro)
        {
            Console.WriteLine($"O, нет! На озере
{lake[num_Nitro].name} появился НитроХряк! Теперь оно и его обитатели
уничтожено!");

            int b = lake[num_Nitro].lake.Length;
            while (b > 0)
            {
                int farm_ducks = random.Next(1,
lake[num_Nitro].lake.Length) - 1;
                Duck exploded_duck =
lake[num_Nitro].remove(farm_ducks);
                b--;
            }

            lake[num_Nitro].lake_live = false;
            lake[num_Nitro].days_Nitro = 1;

            }
            break;
        }
    }
    else
    {
        num_Nitro = random.Next(count_lakes + 1);
    }
}
```

Далее производим проверку на целостность фермы. Если она целая, то охотники могут идти охотиться и утки могут сбегать с неё соответственно,

иначе выводить сообщение о том, что ферма с охотниками разрушена, и цикл охоты и побега прекращается. Для начала разберёмся с охотниками. Делаем проверку на целостность озера, куда они отправятся, а также проверку на наличие уток. Если оба этих условия выполняются, тогда запускаем бесконечный цикл охоты. Если охота происходит, тогда выходим из цикла, если же нет, тогда делаем проверку. Если озеро не разрушено, но уток на нём нет, либо оно просто разрушено, тогда меняем номер озера на которое отправятся охотники. Потом реализуем побег. Делаем проверку на наличие уток на ферме. Если они есть, тогда создаём экземпляр типа фермы и применяем повышающее преобразование до типа озера, создаем пустой массив в котором будем хранить элементы типа Duck. После делаем проверку на то, смогут ли утки, которые находятся на озере сбежать (они должны уметь летать и не знать, где живут, а также быть не подрезаны, эта логика прописана в методе escape класса Farm) при вызове метода escape(ref escaped). Если же могут, то тогда для каждой утки, которая может сбежать делаем проверку на целостность озера, на которое она собирается улететь. Если озеро целое, тогда этой утке меняем переменную home на то озеро куда она сбегает и добавляем эту утку на соответствующее озеро, иначе меняем номер озера.

Листинг 10 – Реализация охоты

```
if (lake[4].lake_live == true)
{
    while (true)
    {
        if (lake[randomlake].lake_live == true &&
lake[randomlake].lake.Length != 0)
        {

            bool a = true;
            while (a)
            {

                if (lake[randomlake].lake.Length != 0)
                {
                    if (hunt_ducks <=
lake[randomlake].lake.Length)
                    {
                        Console.WriteLine($"Охотник с фермы
{lake[4].name} отправился на охоту на озеро {lake[randomlake].name} и поймал
{hunt_ducks} уток");
```

```

while (hunt_ducks > 0)
{
    lake[randomlake].lake.Length) - 1;
    lake[randomlake].remove(lake_ducks);

    int lake_ducks = random.Next(1,
    Duck catchDuck =
    catchDuck.home = lake[4].name;
    catchDuck.count_cut += 1;
    lake[4].add_ducks(catchDuck);
    lake_ducks--;
    hunt_ducks--;

    if (lake[randomlake].lake.Length <=
0) hunt_ducks = 0;
}

a = false;
}

else
{
    Console.WriteLine($"Охотник с фермы
{lake[4].name} отправился на охоту на озеро {lake[randomlake].name} и поймал
{lake[randomlake].lake.Length} уток");
    int c = lake[randomlake].lake.Length;
    while (c > 0)
    {
        lake[randomlake].lake.Length) - 1;
        lake[randomlake].remove(lake_ducks);

        Duck catchDuck =
        catchDuck.home = lake[4].name;
        catchDuck.count_cut += 1;
        lake[4].add_ducks(catchDuck);
        lake_ducks--;
        c--;
        if (lake[randomlake].lake.Length <=
0) c = 0;
    }

    a = false;
}

}

else
{
    randomlake = random.Next(count_lakes);
}
}
break;
}
else if ((lake[randomlake].lake_live == true &&
lake[randomlake].lake.Length == 0) || (lake[randomlake].lake_live == false))
{
    randomlake = random.Next(count_lakes);
}
if (lake[4].lake.Length != 0)
{
    Farm farm = (Farm)lake[4];

```

```

        var escaped = new Duck[0];
        if (farm.escape(ref escaped) == true)
        {

            Console.WriteLine($"Сбежало с фермы
{lake[4].name} уток: {escaped.Length}");

            for (int z = 0; z < escaped.Length; z++)
            {

                int id_lake = random.Next(count_lakes);
                while (true)
                {

                    if (lake[id_lake].lake_live ==
true)

                    {
                        Console.Write("Утка под номером: "
+ Convert.ToString(escaped[z].ID));
                        lake[id_lake].name;

                        lake[id_lake].add_ducks(escaped[z]);

                        Console.WriteLine($" сбежала на
озеро {lake[id_lake].name}");

                        break;
                    }
                    else
                    {
                        id_lake = random.Next(count_lakes);
                    }
                }
            }
        }
    }
}

else
{
    Console.Write($"Охотник был тяжело ранен НитроХряком
восстановится через {lake[4].days_Nitro} ");
    switch (lake[4].days_Nitro)
    {
        case 1:
            Console.WriteLine("день");
            break;
        case 2:
            Console.WriteLine("дня");
            break;
    }
}
}

```

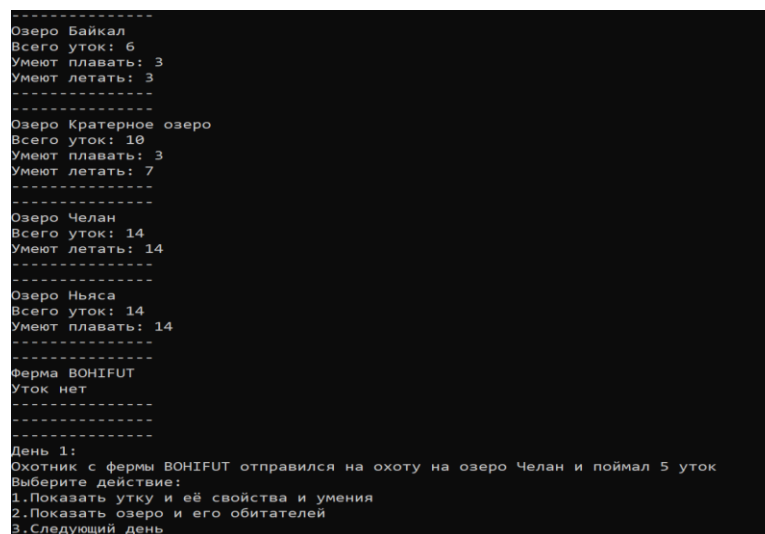
В заключении создаем интерфейс при котором, пользователю будет удобно взаимодействовать с утками и озёрами

Листинг 11 – Интерфейс

```
while (true)
{
    Console.WriteLine("Выберите действие:\n1.Показать утку и её
свойства и умения\n2.Показать озеро и его обитателей\n3.Следующий день");
    int move = Convert.ToInt32(Console.ReadLine());

    if (move == 1)
    {
        showDuck(lake);
    }
    else if (move == 2)
    {
        showLake(lake);
    }
    else if (move == 3)
    {
        break;
    }
    else
    {
        Console.WriteLine("Неправильная команда! Введите номер
команды ещё раз!");
    }
}
```

И когда сезон охоты закончен, выводим сообщение об этом. Далее приведены скриншоты работы программы (рис. 1-5)



```
-----
Озеро Байкал
Всего уток: 6
Умеют плавать: 3
Умеют летать: 3
-----
Озеро Кратерное озеро
Всего уток: 10
Умеют плавать: 3
Умеют летать: 7
-----
Озеро Челан
Всего уток: 14
Умеют летать: 14
-----
Озеро Ньяса
Всего уток: 14
Умеют плавать: 14
-----
Ферма ВОНИФУТ
Уток нет
-----
-----
День 1:
Охотник с фермы ВОНИФУТ отправился на охоту на озеро Челан и поймал 5 уток
Выберите действие:
1.Показать утку и её свойства и умения
2.Показать озеро и его обитателей
3.Следующий день
```

Рис. 1 – Начало программы

```

День 1:
Охотник с фермы BONIFUT отправился на охоту на озеро Челан и поймал 5 уток
Выберите действие:
1.Показать утку и её свойства и умения
2.Показать озеро и его обитателей
3.Следующий день
1
Введите номер озера: 0
Введите номер утки: 5
-----
Имя: Атава
Тип утки: Чернети 1
Вес: 7
Высота: 10
Форма лап: вытянутые
Умеет летать
Не подрезана
-----
Выберите действие:
1.Показать утку и её свойства и умения
2.Показать озеро и его обитателей
3.Следующий день

```

Рис.2 – Демонстрация утки

```

-----
День 2:
О, нет! На озере Челан появился НитроХряк! Теперь оно и его обитатели уничтожено!
Охотник с фермы BONIFUT отправился на охоту на озеро Ньяса и поймал 4 уток
Выберите действие:
1.Показать утку и её свойства и умения
2.Показать озеро и его обитателей
3.Следующий день
2
Введите номер озера: 3
-----
Озеро Ньяса
Всего уток: 10
Умеют плавать: 10
-----
Выберите действие:
1.Показать утку и её свойства и умения
2.Показать озеро и его обитателей
3.Следующий день
2
Введите номер озера: 2
-----
Озеро уничтожено!
-----
Выберите действие:
1.Показать утку и её свойства и умения
2.Показать озеро и его обитателей
3.Следующий день

```

Рис. 3 – Показ озера и реализация НитроХряка

```

3.Следующий день
3
-----
День 3:
0, нет! На озере Байкал появился НитроХряк! Теперь оно и его обитатели уничтожено!
Сбежало с фермы BONIFUT уток: 5
Утка под номером: 40 сбежала на озеро Кратерное озеро
Утка под номером: 22 сбежала на озеро Кратерное озеро
Утка под номером: 27 сбежала на озеро Кратерное озеро
Утка под номером: 20 сбежала на озеро Кратерное озеро
Утка под номером: 34 сбежала на озеро Ньяса
Охотник с фермы BONIFUT отправился на охоту на озеро Кратерное озеро и поймал 6 уток
Выберите действие:
1.Показать утку и её свойства и умения
2.Показать озеро и его обитателей
3.Следующий день

```

Рис. 4 – Побег уток

```

-----
День 6:
Охотник с фермы BONIFUT отправился на охоту на озеро Ньяса и поймал 4 уток
Выберите действие:
1.Показать утку и её свойства и умения
2.Показать озеро и его обитателей
3.Следующий день
3
-----
День 7:
Охотник с фермы BONIFUT отправился на охоту на озеро Ньяса и поймал 3 уток
Выберите действие:
1.Показать утку и её свойства и умения
2.Показать озеро и его обитателей
3.Следующий день
3
-----
День 8:
Охотник с фермы BONIFUT отправился на охоту на озеро Ньяса и поймал 4 уток
Выберите действие:
1.Показать утку и её свойства и умения
2.Показать озеро и его обитателей
3.Следующий день
3
-----
Сезон охоты закончен!

```

Рис. 5 – Завершение программы

Список использованных источников

1. Шилдт, Г. С# 4.0: полное руководство / Г. Шилдт; пер. с англ. – М.: ООО «И.Д. Вильямс», 2011 – 1056 с.: ил. – Парал. тит. англ.
2. Рихтер, Дж. CLR via C#. Программирование на платформе Microsoft .NET Framework 4.5 на языке C#. 4-е изд. / Дж. Рихтер; пер. с англ. – СПб.: Питер, 2013. – 896 с.: ил. – (Серия «Мастер-класс»).
3. Хейлсберг, А., Торгерсен М., Вилтамут С., Голд П. Язык программирования C#. Классика Computers science. 4-е изд. / А. Хейлсберг, М. Торгерсен, С. Вилтамут, П. Голд. – СПб.: Питер, 2012. – 784 с.: ил.

Приложение

Исходный код программы:

```
using System;

namespace Program
{
    class Simulyator_ducks
    {
        static void Main(string[] args)
        {

            Random random = new Random();
            int ducks = 44;
            int id = 1;
            int hunt_days = 8;
            int count_lakes = 4;

            var lake = new Lake[5];
            lake[0] = new Lake("Байкал");
            lake[1] = new Lake("Кратерное озеро");
            lake[2] = new Lake("Челан");
            lake[3] = new Lake("Ньяса");

            lake[4] = new Farm("BOHIFUT");

            for (int i = 0; i < ducks; i++)
            {
                switch (random.Next(1, 5))
                {
                    case 1:
                    {
                        if (random.Next(0, 2) == 0)
                        {
                            lake[0].add_ducks(new Chernety1(ref id));
                        }
                        else
                        {
                            lake[0].add_ducks(new Chernety2(ref id));
                        }
                    }
                    break;
                    case 2:
                    {
                        if (random.Next(0, 2) == 0)
                        {
                            lake[1].add_ducks(new Mramchir(ref id));
                        }
                        else
                        {
                            lake[1].add_ducks(new Shirok1(ref id));
                        }
                    }
                    break;
                }
            }
        }
    }
}
```

```

        case 3:
        {
            if (random.Next(0, 2) == 0)
            {
                lake[2].add_ducks(new Shirok2(ref id));
            }
            else
            {
                lake[2].add_ducks(new Krohali(ref id));
            }
        }
        break;
        case 4:
        {
            if (random.Next(0, 2) == 0)
            {
                lake[3].add_ducks(new Chernety3(ref id));
            }
            else
            {
                lake[3].add_ducks(new Stoneduck(ref id));
            }
        }
        break;
    }

}

showLakes(lake);

for (int i = 1; i < hunt_days + 1; i++)
{
    int Nitro = 2;
    int sumducks = 0;
    int num_Nitro = random.Next(count_lakes + 1);
    int hunt_ducks = random.Next(1, 9);
    int randomlake = random.Next(count_lakes);

    Console.WriteLine("-----");
    Console.WriteLine($"День {i}:");

    for (int s = 0; s < count_lakes; s++)
    {
        sumducks += lake[s].lake.Length;
    } // Chech sumducks
    if (sumducks == 0)
    {
        Console.WriteLine("Уток на диких озёрах больше нет!");
        break;
    }

    for (int p = 0; p < count_lakes + 1; p++)
    {
        if (lake[p].lake_live == false)
        {
            if (lake[p].days_Nitro == 0)

```

```

        {
            lake[p].lake_live = true;
        }
        else
        {
            lake[p].days_Nitro--;
        }
    }
} //NitroCrack
while (true)
{
    if (lake[num_Nitro].lake_live == true)
    {
        if (random.Next(0, 4) == Nitro)
        {
            Console.WriteLine($"O, нет! На озере
{lake[num_Nitro].name} появился НитроХряк! Теперь оно и его обитатели
уничтожено!");

            int b = lake[num_Nitro].lake.Length;
            while (b > 0)
            {
                int farm_ducks = random.Next(1,
lake[num_Nitro].lake.Length) - 1;
                Duck exploded_duck =
lake[num_Nitro].remove(farm_ducks);
                b--;
            }

            lake[num_Nitro].lake_live = false;
            lake[num_Nitro].days_Nitro = 1;
        }
        break;
    }
    else
    {
        num_Nitro = random.Next(count_lakes + 1);
    }
}

if (lake[4].lake_live == true)
{
    while (true)
    {
        if (lake[randomlake].lake_live == true &&
lake[randomlake].lake.Length != 0)
        {

            bool a = true;
            while (a)
            {

                if (lake[randomlake].lake.Length != 0)

```

```

        {
            if (hunt_ducks <=
lake[randomlake].lake.Length)
            {
                Console.WriteLine($"Охотник с фермы
{lake[4].name} отправился на охоту на озеро {lake[randomlake].name} и поймал
{hunt_ducks} уток");
                while (hunt_ducks > 0)
                {
                    int lake_ducks = random.Next(1,
lake[randomlake].lake.Length) - 1;
                    Duck catchDuck =
lake[randomlake].remove(lake_ducks);
                    catchDuck.home = lake[4].name;
                    catchDuck.count_cut += 1;
                    lake[4].add_ducks(catchDuck);
                    lake_ducks--;
                    hunt_ducks--;
                    if (lake[randomlake].lake.Length <=
0) hunt_ducks = 0;
                }
                a = false;
            }
            else
            {
                Console.WriteLine($"Охотник с фермы
{lake[4].name} отправился на охоту на озеро {lake[randomlake].name} и поймал
{lake[randomlake].lake.Length} уток");
                int c = lake[randomlake].lake.Length;
                while (c > 0)
                {
                    int lake_ducks = random.Next(1,
lake[randomlake].lake.Length) - 1;
                    Duck catchDuck =
lake[randomlake].remove(lake_ducks);
                    catchDuck.home = lake[4].name;
                    catchDuck.count_cut += 1;
                    lake[4].add_ducks(catchDuck);
                    lake_ducks--;
                    c--;
                    if (lake[randomlake].lake.Length <=
0) c = 0;
                }
                a = false;
            }
        }
        else
        {
            randomlake = random.Next(count_lakes);
        }
    }
    break;
}

```

```

        else if ((lake[randomlake].lake_live == true &&
lake[randomlake].lake.Length == 0) || (lake[randomlake].lake_live == false))
        {
            randomlake = random.Next(count_lakes);
        }
        if (lake[4].lake.Length != 0)
        {
            Farm farm = (Farm)lake[4];
            var escaped = new Duck[0];
            if (farm.escape(ref escaped) == true)
            {

                Console.WriteLine($"Сбежало с фермы
{lake[4].name} уток: {escaped.Length}");

                for (int z = 0; z < escaped.Length; z++)
                {

                    int id_lake = random.Next(count_lakes);
                    while (true)
                    {

                        if (lake[id_lake].lake_live ==

true)

                        {
                            Console.Write("Утка под номером: "
+ Convert.ToString(escaped[z].ID));
                            escaped[z].home =
lake[id_lake].name;

lake[id_lake].add_ducks(escaped[z]);

                            Console.WriteLine($" сбежала на
озеро {lake[id_lake].name}");
                            break;
                        }
                        else
                        {
                            id_lake = random.Next(count_lakes);
                        }
                    }
                }
            }
        }
    }
}

else
{
    Console.Write($"Охотник был тяжело ранен НитроХряком
восстановится через {lake[4].days_Nitro} ");
    switch (lake[4].days_Nitro)
    {
        case 1:

```

```

        Console.WriteLine("день");
        break;
    case 2:
        Console.WriteLine("дня");
        break;
    }
}

while (true)
{
    Console.WriteLine("Выберите действие:\n1.Показать утку и её
свойства и умения\n2.Показать озеро и его обитателей\n3.Следующий день");
    int move = Convert.ToInt32(Console.ReadLine());

    if (move == 1)
    {
        showDuck(lake);
    }
    else if (move == 2)
    {
        showLake(lake);
    }
    else if (move == 3)
    {
        break;
    }
    else
    {
        Console.WriteLine("Неправильная команда! Введите номер
команды ещё раз!");
    }
}

Console.WriteLine("-----\nСезон охоты закончен!");
Console.ReadKey();
}

static void showLakes(Lake[] lake)
{
    for (int i = 0; i < 5; i++)
    {
        lake[i].Lakevoice(i);
    }
    Console.WriteLine("-----");
}

static void showLake(Lake[] lake)
{

```

```

        Console.WriteLine("Введите номер озера: ");
        int lake_id = Convert.ToInt32(Console.ReadLine());

        lake[lake_id].Lakevoice(lake_id);

        Console.WriteLine("-----");
    }

    static void showDuck(Lake[] lake)
    {

        Console.WriteLine("Введите номер озера: ");
        int lake_id = Convert.ToInt32(Console.ReadLine());
        Console.WriteLine("Введите номер утки: ");
        int duck_id = Convert.ToInt32(Console.ReadLine());

        lake[lake_id].lake[duck_id].voice();

        Console.WriteLine("-----");
    }

}
abstract public class Duck
{
    public string type = "";
    public bool swim = false;
    public bool knowhome = false;
    public bool fly = false;
    public string home = "";
    public int ID = 0;
    public bool cut;
    public int count_cut;

    public Duck(ref int id)
    {
        ID = id;
        id++;
    }

    public abstract void voice();
}

public static class par
{
    static Random random = new Random();
    public static string gencolour()
    {
        var colour = new string[] { "красный", "голубой", "зелёный",
"жёлтый", "белый", "чёрный" };
        return colour[random.Next(0, 6)];
    }
    public static string gencolourwings()
    {
        var colourwings = new string[] { "красный", "голубой", "зелёный",
"жёлтый", "белый", "чёрный" };
        return colourwings[random.Next(0, 6)];
    }
    public static string genformwings()
    {

```



```

        var formwings = new string[] { "круглые", "овальные", "вытянутые"
};
        return formwings[random.Next(0, 2)];
    }
    public static string geneyecolour()
    {
        var eyecolour = new string[] { "чёрный", "коричневый", "синий" };
        return eyecolour[random.Next(0, 2)];
    }
    public static int genage()
    {
        return random.Next(1, 9);
    }
    public static int genheight()
    {
        return random.Next(5, 15);
    }
    public static string genpaws()
    {
        var formpaws = new string[] { "круглые", "овальные", "вытянутые" };
        return formpaws[random.Next(0, 3)];
    }
    public static string genfavouritedish()
    {
        var favouritedish = new string[] { "травы", "рыба", "корм" };
        return favouritedish[random.Next(0, 3)];
    }
    public static int genwidth()
    {
        return random.Next(5, 10);
    }
    public static string genformbeak()
    {
        var forms = new string[] { "кривой", "овальный", "вытянутый" };
        return forms[random.Next(0, 3)];
    }
    public static int genhealth()
    {
        return random.Next(50, 100);
    }
    public static string gensex()
    {
        var sex = new string[] { "самец", "самка" };
        return sex[random.Next(0, 1)];
    }
    public static string genname()
    {
        var name = new string[] { "Озон", "Атава", "Перди", "Красавка",
"Филя", "Мамба", "Алок", "Вин", "Ерон", "Чепчик", "Митч", "Дюна", "Ловкий",
"Мариус", "Фредди" };
        return name[random.Next(0, 15)];
    }
    public static int genweight()
    {
        return random.Next(5, 17);
    }
}
class Chernety1 : Duck
{
    public Chernety1(ref int id) : base(ref id)
    {

```

```

        type = "Чернети 1";
        home = "Байкал";
        fly = true;
        swim = false;
        knowhome = false;
        cut = false;
        count_cut = 0;
    }

    int weight = par.genweight();
    string paws = par.genpaws();
    int height = par.genheight();
    string name = par.genname();

    public override void voice()
    {
        Console.WriteLine("-----");
        string a = $"Имя: {name}\nТип утки: {type}\nВес: {weight}\nВысота:
{height}\nФорма лап: {paws}\n";
        if (fly) a += "Умеет летать\n";
        if (swim) a += "Умеет плавать\n";
        if (knowhome) a += $"Живёт на озере {home}\n";
        if (cut == true)
        {
            a += "Подрезана";

        }
        else
        {
            a += "Не подрезана";
        }
        Console.WriteLine(a);

        Console.WriteLine("-----");
    }

}

class Chernety2 : Duck
{
    public Chernety2(ref int id) : base(ref id)
    {
        type = "Чернети 2";
        home = "Байкал";
        fly = false;
        knowhome = false;
        swim = true;
        cut = false;
        count_cut = 0;
    }

    int weight = par.genweight();
    int age = par.genage();
    string sex = par.gensex();
    string name = par.genname();
    public override void voice()
    {
        Console.WriteLine("-----");
    }
}

```

```

        string a = $"Имя: {name}\nТип утки: {type}\nВес: {weight}\nПол:
{sex}Возраст: {age}\n";
        if (fly) a += "Умеет летать\n";
        if (swim) a += "Умеет плавать\n";
        if (knowhome) a += $"Живёт на озере {home}\n";
        if (cut == true)
        {
            a += "Подрезана";

        }
        else
        {
            a += ("Не подрезана");
        }
        Console.WriteLine(a);
        Console.WriteLine("-----");
    }

}

class Chernety3 : Duck
{
    public Chernety3(ref int id) : base(ref id)
    {

        type = "Чернети 3";
        home = "Ньяса";
        fly = false;
        knowhome = true;
        swim = true;
        cut = false;
        count_cut = 0;
    }

    int width = par.genwidth();
    int health = par.genhealth();
    int weight = par.genweight();
    string name = par.genname();
    public override void voice()
    {
        Console.WriteLine("-----");
        string a = $"Имя: {name}\nТип утки: {type}\nВес: {weight}\nШирина:
{width}\nЗдоровье: {health}\n";
        if (fly) a += "Умеет летать\n";
        if (swim) a += "Умеет плавать\n";
        if (knowhome) a += $"Живёт на озере {home}\n";
        Console.WriteLine(a);
        if (cut == true)
        {
            a += "Подрезана";

        }
        else
        {
            a += ("Не подрезана");
        }
        Console.WriteLine(a);
        Console.WriteLine("-----");
    }
}

```

```

    }

    class Mramchir : Duck
    {
        public Mramchir(ref int id) : base(ref id)
        {
            type = "Мраморная чирка";
            home = "Кратерное озеро";
            fly = false;
            knowhome = false;
            swim = true;
            cut = false;
            count_cut = 0;
        }

        string formwings = par.genformbeak();
        string colourwings = par.gencolourwings();
        int weight = par.genweight();
        string name = par.genname();
        public override void voice()
        {
            Console.WriteLine("-----");
            string a = $"Имя: {name}\nТип утки: {type}\nВес: {weight}\nФорма
крыльев: {formwings}\nЦвет крыльев: {colourwings}\n";
            if (fly) a += "Умеет летать\n";
            if (swim) a += "Умеет плавать\n";
            if (knowhome) a += $"Живёт на озере {home}\n";
            if (cut == true)
            {
                a += "Подрезана";
            }
            else
            {
                a += ("Не подрезана");
            }
            Console.WriteLine(a);
            Console.WriteLine("-----");
        }
    }

    class Shirok1 : Duck
    {
        public Shirok1(ref int id) : base(ref id)
        {
            type = "Широконоска 1";
            home = "Кратерное озеро";
            fly = true;
            knowhome = true;
            swim = false;
            cut = false;
            count_cut = 0;
        }

        string eyecolour = par.geneyecolour();
        string colour = par.gencolour();
        int weight = par.genweight();
    }

```

```

        string name = par.genname();
        public override void voice()
        {
            Console.WriteLine("-----");
            string a = $"Имя: {name}\nТип утки: {type}\nВес: {weight}\nЦвет
глаз: {eyecolour}\nЦвет: {colour}\n";
            if (fly) a += "Умеет летать\n";
            if (swim) a += "Умеет плавать\n";
            if (knowhome) a += $"Живёт на озере {home}\n";
            if (cut == true)
            {
                a += "Подрезана";
            }
            else
            {
                a += ("Не подрезана");
            }
            Console.WriteLine(a);
            Console.WriteLine("-----");
        }

    }

    class Shirok2 : Duck
    {
        public Shirok2(ref int id) : base(ref id)
        {
            type = "Широконоска 2";
            home = "Челан";
            fly = true;
            knowhome = false;
            swim = false;
            cut = false;
            count_cut = 0;
        }

        string formwings = par.genformwings();
        int width = par.genwidth();
        int weight = par.genweight();
        string name = par.genname();
        public override void voice()
        {
            Console.WriteLine("-----");
            string a = $"Имя: {name}\nТип утки: {type}\nВес: {weight}\nФорма
крыльев: {formwings}\nШирина: {width}\n";
            if (fly) a += "Умеет летать\n";
            if (swim) a += "Умеет плавать\n";
            if (knowhome) a += $"Живёт на озере {home}\n";
            if (cut == true)
            {
                a += "Подрезана";
            }
            else
            {
                a += ("Не подрезана");
            }
            Console.WriteLine(a);
        }
    }

```

```

        Console.WriteLine("-----");
    }

}

class Krohali : Duck
{
    public Krohali(ref int id) : base(ref id)
    {
        type = "Крохали";
        home = "Челан";
        fly = true;
        knowhome = false;
        swim = false;
        cut = false;
        count_cut = 0;
    }

    string colourwings = par.gencolourwings();
    int height = par.genheight();
    int weight = par.genweight();
    string name = par.genname();
    public override void voice()
    {
        Console.WriteLine("-----");
        string a = $"Имя: {name}\nТип утки: {type}\nВес: {weight}\nОкрас
крыльев: {colourwings}\nВысота: {height}\n";
        if (fly) a += "Умеет летать\n";
        if (swim) a += "Умеет плавать\n";
        if (knowhome) a += $"Живёт на озере {home}\n";

        if (cut == true)
        {
            a += "Подрезана";
        }
        else
        {
            a += "Не подрезана";
        }
        Console.WriteLine(a);
        Console.WriteLine("-----");
    }
}

class Stoneduck : Duck
{
    public Stoneduck(ref int id) : base(ref id)
    {
        type = "Каменушки";
        home = "Ньяса";
        fly = false;
        knowhome = true;
        swim = true;
        cut = false;
        count_cut = 0;
    }
}

```

```

        string formbeak = par.genformbeak();
        string favouritedish = par.genfavouritedish();
        int weight = par.genweight();
        string name = par.genname();
        public override void voice()
        {
            Console.WriteLine("-----");
            string a = $"Имя: {name}\nТип утки: {type}\nВес {weight}\nФорма
клюва: {formbeak}\nЛюбимое блюдо: {favouritedish}\n";
            if (fly) a += "Умеет летать\n";
            if (swim) a += "Умеет плавать\n";
            if (knowhome) a += $"Живёт на озере {home}\n";

            if (cut == true)
            {
                a += "Подрезана";
            }
            else
            {
                a += ("Не подрезана");
            }
            Console.WriteLine(a);
            Console.WriteLine("-----");
        }
    }

class Lake
{
    Random random = new Random();

    public Duck[] lake = new Duck[0];
    int size = 0;
    public bool lake_live;
    public string name;
    public int days_Nitro;
    public Lake(string name)
    {
        this.name = name;
        this.lake_live = true;
        this.days_Nitro = 1;
    }

    public void add_ducks(Duck duck)
    {
        Array.Resize(ref lake, size + 1);
        lake[size] = duck;
        size++;
    }

    virtual public void Lakevoice(int n)
    {
        int swim = 0;
        int fly = 0;

        for (int i = 0; i < size; i++)
        {
            if (lake[i].swim == true) swim++;
            if (lake[i].fly == true) fly++;
        }
    }
}

```

```

    }

    if (lake_live == true)
    {
        Console.WriteLine("-----");
        Console.WriteLine($"Озеро {name}");
        if (size > 0)
        {
            Console.WriteLine($"Всего уток: {size}");
            if (swim > 0) Console.WriteLine($"Умеют плавать: {swim}");
            if (fly > 0) Console.WriteLine($"Умеют летать: {fly}");
        }

        else
        {
            Console.WriteLine("Уток нет");
        }
    }
    else
    {
        Console.WriteLine("-----\nОзеро уничтожено!");
    }
    Console.WriteLine("-----");
}

public Duck remove(int num)
{
    Duck duck = lake[num];
    while (num < size - 1)
    {
        lake[num] = lake[num + 1];
        num++;
    }
    Array.Resize(ref lake, size - 1);
    size--;

    return duck;
}

}

class Farm : Lake
{

    public Farm(string name) : base(name)
    {
        this.name = name;
        this.lake_live = true;
        this.days_Nitro = 1;
    }

    public override void Lakevoice(int n)
    {
        int swim = 0;
        int fly = 0;
        for (int i = 0; i < lake.Length; i++)
        {
            if (lake[i].swim == true) swim++;
            if (lake[i].fly == true) fly++;
        }
    }
}

```



```

    }
    Console.WriteLine("-----");
    if (lake_live == true)
    {
        Console.WriteLine($"Ферма {name}");

        if (lake.Length > 0)
        {
            Console.WriteLine($"Всего уток: {lake.Length}");
            if (swim > 0) Console.WriteLine($"Умеют плавать: {swim}");
            if (fly > 0) Console.WriteLine($"Умеют летать: {fly}");
        }

        else
        {
            Console.WriteLine("Уток нет");
        }
        Console.WriteLine("-----");
    }
    else
    {
        Console.WriteLine($"Ферма {this.name} с охотниками
уничтожена");
    }
}

public bool escape(ref Duck[] duck)
{
    int i = 0;
    int arrl = 0;
    bool freeducks = false;
    while (i < lake.Length)
    {
        if (lake[i].fly == true && lake[i].knowhome == false &&
lake[i].cut == false)
        {
            Array.Resize(ref duck, arrl + 1);
            duck[arrl] = remove(i);
            if (duck[arrl].count_cut == 1)
            {
                duck[arrl].cut = true;
                duck[arrl].fly = false;
            }

            arrl++;

            freeducks = true;
        }
        else i++;
    }
    return freeducks;
}
}
}

```