**Отчет**

**Лабораторная работа №2**

**С дисциплины: «C#»**

**Тема: «** **Шаблоны проектирования. Структурные шаблоны»**

**Автор: Ярцев Д.А.**

**Группа: ИС61**

**Цель**: ознакомиться с основными шаблонами проектирования, научиться применять их при проектировании и разработке ПО.

**Паттерн** — это не законченное архитектурное решение, которое можно напрямую преобразовать в исходный или машинный код. Это описание подхода к решению проблемы, который можно применять в разных ситуациях

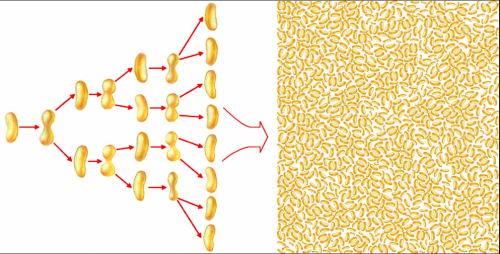
**Вариант №8**

Разработать систему учета процессов размножения колонии бактерий.

**Ход работы**

**Размножение бактерий**

Как и другие одноклеточные организмы, бактерии размножаются делением. Каждая бактерия делится на две дочерние, которые быстро растут и снова делятся. При благоприятных условиях деление клеток у многих бактерий может происходить через каждые 20 – 30 мин.



При таком быстром размножении одна бактерия могла бы дать за сутки (при условии выживания всех появляющихся особей) потомство общей массой в 1 800 000 кг, а потомство одной бактерии за 5 суток способно было бы заполнить все моря и океаны на Земле!

Однако в природе этого не происходит, так как большинство бактерий быстро погибает под действием солнечного света, при высушивании, недостатке пищи, нагревании до 65 – 100 °С, под действием дезинфицирующих веществ и т. д.колонии бактерий- скопления бактерий одного вида.

**Колония бактерий** - изолированное скопление клеток бактерий одного вида

**Описание программы:**

Система учета процессов размножения колонии бактерий, по определению должна вести учет количесва бактерий за определенный промежуток времени. Тк колонии бактерий могут отличаться между собой только типом размножения - входными данными в программу будут: К-тво бактерий в начале наблюдения, сколько времени будут проводиться наблюдения и способ деления бактерий.

(Для удобства проверки роботы врямя необходимое на деление бактерии будет 1с.)

В качестве выходных данных будет – к-тво бактерий.

**Общий алгоритм работы программы таков:**

1. Вывод меню возможных действий
2. Выбор действия
3. Выполнение действия
4. Выбор меню действий
5. Завершение программы

Привязка – это логический термин, который заставляет программиста задуматься над смыслом зависимости.

Чтобы минимизировать количество привязок и при этом сохранять логическую целостность смысла моделируемого процесса я хочу восполюзоваться паттреном Façade, так как именно этой цели он и служит.

**Реализация: вывод**

Lab 2 c#, Author: Yarcev D.A. from group IS-61

variant 8

================================================================================

Menu:

1) Create bacteria of colony

2) Show colony growth statistics

3) destroy the bacteria of colony

4) close this program

To do: 1

--------------------------------------------------------------------------------

input name for colony: name\_colony

input step for colony: 2

input start count bacterias: 16

================================================================================

Menu:

1) Create bacteria of colony

2) Show colony growth statistics

3) destroy the bacteria of colony

4) close this program

To do: 2

--------------------------------------------------------------------------------

05.12.2018 23:18:39 ~ 16

05.12.2018 23:18:40 ~ 32

================================================================================

Menu:

1) Create bacteria of colony

2) Show colony growth statistics

3) destroy the bacteria of colony

4) close this program

To do: 2

--------------------------------------------------------------------------------

05.12.2018 23:18:39 ~ 16

05.12.2018 23:18:40 ~ 32

05.12.2018 23:18:42 ~ 64

================================================================================

Menu:

1) Create bacteria of colony

2) Show colony growth statistics

3) destroy the bacteria of colony

4) close this program

To do: 3

--------------------------------------------------------------------------------

Bacteria colony has been removed.

================================================================================

Menu:

1) Create bacteria of colony

2) Show colony growth statistics

3) destroy the bacteria of colony

4) close this program

To do: 4

--------------------------------------------------------------------------------

Program is ending...

================================================================================

**Реализация: код**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Yarccev\_c\_lab\_2

{

class Colony

{

private string name = "-";

public string Name

{

get { return name; }

set { name = value; }

}

private int step = 2;

public int Step

{

get { return step; }

set { step = value; }

}

private int count = 0;

public int Count

{

get { return count; }

set { count = value; }

}

private DateTime time;

public DateTime Time

{

get { return time; }

set { time = value; }

}

//логи времени

private List<DateTime> logsTime;

public List<DateTime> LogsTime

{

get { return logsTime; }

set { logsTime = value; }

}

//логи значений

private List<int> logsСount;

public List<int> LogsСount

{

get { return logsСount; }

set { logsСount = value; }

}

}

}

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Yarccev\_c\_lab\_2

{

/// <summary>

/// для удобства демонстрации

/// бактерии будут размножаться

/// раз в 1 сек а не так как в природе

/// раз в 30 мин.

/// </summary>

class Program

{

static void lineCh(char s)

{

for (int i = 0; i < 80; i++)

Console.Write(s);

Console.WriteLine();

}

static void Main(string[] args)

{

Console.WriteLine(" Lab 2 c#, Author: Yarcev D.A. from group IS-61");

Console.WriteLine(" variant 8");

lineCh('=');

//ключ цыкла программы

bool keyLoop = true;

//наша колония

Colony colony = null;

//ключ выбора

int chose = 0;

while (keyLoop)

{

//меню

Console.WriteLine(" Menu:");

Console.WriteLine(" 1) Create bacteria of colony");

Console.WriteLine(" 2) Show colony growth statistics");

Console.WriteLine(" 3) destroy the bacteria of colony");

Console.WriteLine(" 4) close this program");

//выбор

bool k = true;

while(k)

{

k = false;

Console.Write(" To do: ");

chose = Convert.ToInt32(Console.ReadLine().ToString());

if (chose < 1 || chose > 4)

{

Console.WriteLine("!!!Error");

k = true;

}

}

lineCh('-');

//общий набор возможностей

Facade facade = new Facade();

//действие

switch (chose){

case 1:

//создать

colony = facade.OperationCreate();

break;

case 2:

//проверить

colony = facade.OperationShow(colony);

break;

case 3:

//очистить

colony = facade.OperationDel();

break;

//закрыть программу

case 4:

keyLoop = false;

Console.WriteLine(" Program is ending...");

break;

}

lineCh('=');

}

Console.ReadLine();

}

}

}

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Yarccev\_c\_lab\_2

{

/// <summary>

/// общий набор функционала

/// содержащийся в подсистемах

/// </summary>

class Facade

{

SubSystem1 subSystem1 = new SubSystem1();

SubSystem2 subSystem2 = new SubSystem2();

SubSystem3 subSystem3 = new SubSystem3();

//создать

public Colony OperationCreate()

{

return subSystem1.Operation();

}

//занести запись в логи и высветить текущее состояние

public Colony OperationShow(Colony colony)

{

return subSystem2.Operation(colony);

}

//удалить

public Colony OperationDel()

{

return subSystem3.Operation();

}

}

}

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Yarccev\_c\_lab\_2

{

class SubSystem1

{

public Colony Operation()

{

Colony colony = new Colony();

Console.Write(" input name for colony: ");

colony.Name = Console.ReadLine().ToString();

Console.Write(" input step for colony: ");

colony.Step = Convert.ToInt32(Console.ReadLine().ToString());

if (colony.Step < 1)

colony.Step = 1;

colony.Time = DateTime.Now;

Console.Write(" input start count bacterias: ");

colony.Count = Convert.ToInt32(Console.ReadLine().ToString());

//начало логов

colony.LogsTime = new List<DateTime>();

colony.LogsTime.Add(DateTime.Now);

colony.LogsСount = new List<int>();

colony.LogsСount.Add(colony.Count);

return colony;

}

}

}

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Yarccev\_c\_lab\_2

{

class SubSystem2

{

public Colony Operation(Colony colony)

{

if (colony != null)

{

//последние логи

double lastT = colony.LogsTime.Last().Subtract(DateTime.MinValue).TotalSeconds;

int lastC = colony.LogsСount.Last();

//дата сейчас

DateTime nowDataTime = DateTime.Now;

double nowT = nowDataTime.Subtract(DateTime.MinValue).TotalSeconds;

int deltaT = (int)Math.Floor((nowT - lastT) / 1);

//провести расчет текущего положения колонии

for (int i = deltaT; i > 0; i--)

colony.Count = colony.Count \* 2;

//сохранить логи

colony.LogsTime.Add(nowDataTime);

colony.LogsСount.Add(colony.Count);

//вывести все логи

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

Console.WriteLine(colony.LogsTime[i].ToString() + " ~ " + colony.LogsСount[i].ToString());

}

else

{

Console.WriteLine(" Colony is not found.");

}

return colony;

}

}

}

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Yarccev\_c\_lab\_2

{

class SubSystem3

{

public Colony Operation()

{

Console.WriteLine(" Bacteria colony has been removed.");

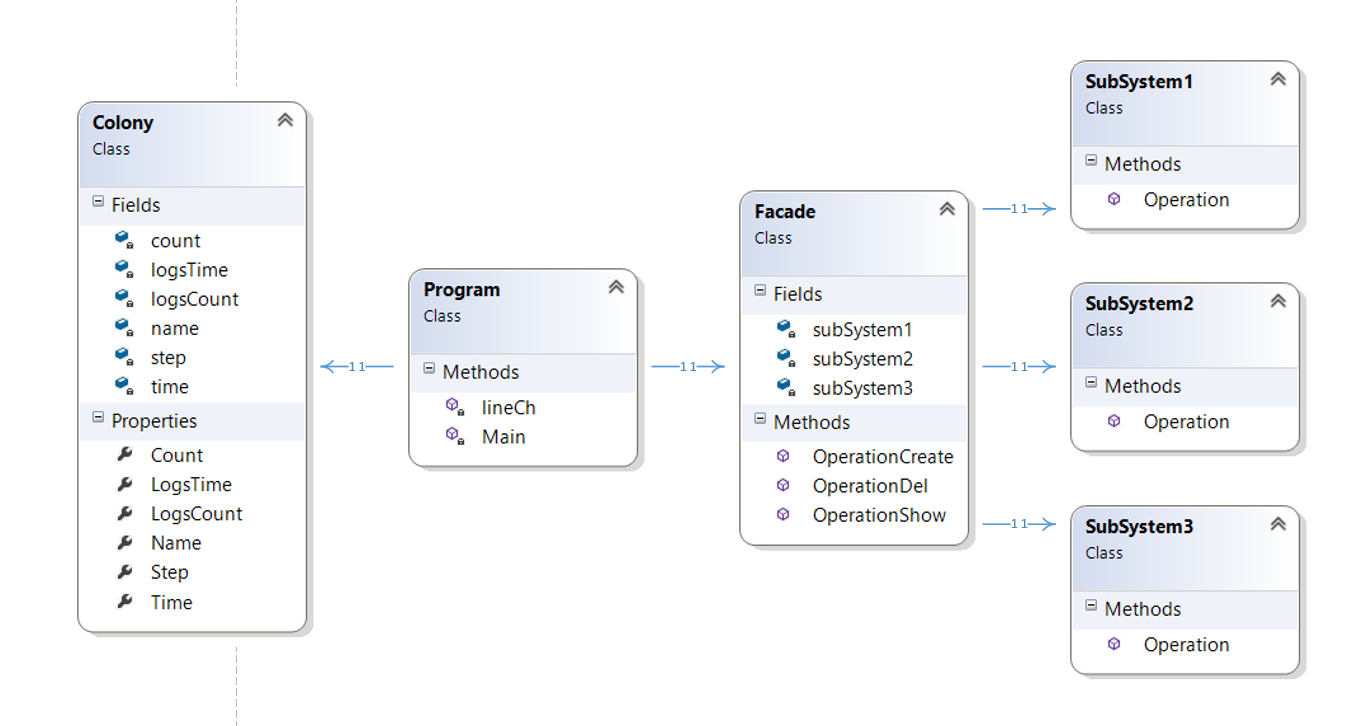
return null;

}

}

}

**Структура программы на языке UML**



**Классы**

|  |  |
| --- | --- |
| **Паттерн Facade**  **Составные классы и их описание** | |
| **Наименование** | **Описание** |
| Program | Главнай класс программы. |
| Facade - Фасад: | Перенаправляет запросы клиентов, другим объектам из которых состоит подсистема. |
| SubSystem1 | Реализует фактическую функциональность системы.  (создает колонию)  Ничего не знает о фасаде (не имеет ссылок на фасад). |
| SubSystem2 | Реализует фактическую функциональность системы.  (обновляет инф. о колонии)  Ничего не знает о фасаде (не имеет ссылок на фасад). |
| SubSystem3 | Реализует фактическую функциональность системы.  (удаляет колонию)  Ничего не знает о фасаде (не имеет ссылок на фасад). |
| **классы и их описание** | |
| **Наименование** | **Описание** |
| colony | Колония, имеет ряд атрибутов и методы, чтобы их изменять (get\set) |

**Вивод:**

Паттерны довольно сильно ускоряют процесс разработки ПО. Обеспечивают четко прослеживаемую структуру.