Министерство образования Республики Беларусь

Учреждение образования «Брестский государственный технический университет»

Кафедра ИИТ

Отчёт по лабораторной работе №4

Выполнил:

студент 3 курса

Группы ПО-5

Поздняков Д.А.

Проверил:

Крощенко А.А.

Брест 2021

**Цель работы:** приобрести практические навыки в области объектно-ориентированного проектирования

**Вариант 5**

**Задание 1.** Реализовать указанный класс, включив в него вспомогательный внутренний класс или классы. Реализовать 2-3 метода (на выбор). Продемонстрировать использование реализованных классов.

Создать класс Department (отдел фирмы) с внутренним классом, с помощью объектов которого можно хранить информацию обо всех должностях отдела и обо всех сотрудниках, когда-либо занимавших конкретную должность.

**Задание 2.** Реализовать агрегирование. При создании класса агрегируемый класс объявляется как атрибут (локальная переменная, параметр метода). Включить в каждый класс 2-3 метода на выбор. Продемонстрировать использование разработанных классов.

5) Создать класс Абзац, используя класс Слово.

**Задание 3.** Построить модель программной системы с применением отношений (обобщения, агрегации, ассоциации, реализации) между классами. Задать атрибуты и методы классов. Реализовать (если необходимо) дополнительные классы. Продемонстрировать работу разработанной системы.

Система Библиотека. Читатель оформляет Заказ на Книгу. Система осуществляет поиск в Каталоге. Библиотекарь выдает Читателю Книгу на абонемент или в читальный зал. При невозвращении Книги Читателем он может быть занесен Администратором в «черный список».

**Код программы (задание 1):**

Program.cs

using System;

using System.Collections.Generic;

namespace lab4.\_1.\_5

{

class Program

{

static void Main(string[] args)

{

Department.Position.Employee James = new Department.Position.Employee(20, "James Grant"),

Ronnie = new Department.Position.Employee(21, "Ronnie Harris"),

John = new Department.Position.Employee(22, "John Torres"),

Matthew = new Department.Position.Employee(23, "Matthew McKinney"),

Francisco = new Department.Position.Employee(24, "Francisco Hall"),

Jerry = new Department.Position.Employee(25, "Jerry Ramsey"),

Chris = new Department.Position.Employee(26, "Chris Hamilton"),

Brian = new Department.Position.Employee(27, "Brian Gonzalez"),

David = new Department.Position.Employee(28, "David Riley");

Department.Position Security = new Department.Position("Security"),

TechnicalGroup = new Department.Position("Technical group"),

HeadOfService = new Department.Position("Head of service");

Security.AddEmployee(James);

Security.AddEmployee(Ronnie);

Security.AddEmployee(John);

TechnicalGroup.AddEmployee(Matthew);

TechnicalGroup.AddEmployee(Francisco);

TechnicalGroup.AddEmployee(Jerry);

HeadOfService.AddEmployee(Chris);

HeadOfService.AddEmployee(Brian);

HeadOfService.AddEmployee(David);

List<Department.Position> SecurityPositions = new List<Department.Position>();

SecurityPositions.Add(Security);

SecurityPositions.Add(TechnicalGroup);

SecurityPositions.Add(HeadOfService);

Department security = new Department(new string("Security Department"));

security.SetPositions(SecurityPositions);

Console.WriteLine(security.GetName() + ":");

foreach (Department.Position Position in security.GetPositions())

{

Console.WriteLine("\t" + Position.GetName() + ":");

foreach (Department.Position.Employee Employee in Position.GetEmployees())

{

Console.WriteLine("\t\t" + Employee.GetName());

}

Console.WriteLine();

}

/\*================================================================================================\*/

Department.Position.Employee Robert = new Department.Position.Employee(20, "Robert Rogers"),

Todd = new Department.Position.Employee(21, "Todd Reed"),

Ronald = new Department.Position.Employee(22, "Ronald White"),

William = new Department.Position.Employee(23, "William Gomez"),

Leroy = new Department.Position.Employee(24, "Leroy Torres"),

Claude = new Department.Position.Employee(25, "Claude McDonald"),

Terry = new Department.Position.Employee(26, "Terry Cook"),

Clifton = new Department.Position.Employee(27, "Clifton Simmons"),

Thomas = new Department.Position.Employee(28, "Thomas Hill");

Department.Position Programmer = new Department.Position("Programmer"),

Administrator = new Department.Position("Administrator"),

SecuritySpecialist = new Department.Position("Security Specialist");

Administrator.AddEmployee(Robert);

Administrator.AddEmployee(Todd);

Administrator.AddEmployee(Ronald);

Programmer.AddEmployee(William);

Programmer.AddEmployee(Leroy);

Programmer.AddEmployee(Claude);

SecuritySpecialist.AddEmployee(Terry);

SecuritySpecialist.AddEmployee(Clifton);

SecuritySpecialist.AddEmployee(Thomas);

Department IT = new Department(new string("IT Department"));

IT.AddPosition(Administrator);

IT.AddPosition(Programmer);

IT.AddPosition(SecuritySpecialist);

Console.WriteLine(IT.GetName() + ":");

foreach (Department.Position Position in IT.GetPositions())

{

Console.WriteLine("\t" + Position.GetName() + ":");

foreach (Department.Position.Employee Employee in Position.GetEmployees())

{

Console.WriteLine("\t\t" + Employee.GetName());

}

Console.WriteLine();

}

}

}

}

Department.cs

using System.Collections.Generic;

namespace lab4.\_1.\_5

{

public class Department

{

private string Name;

private List<Position> Positions = new List<Position>();

public Department()

{

}

public Department(string name)

{

Name = name;

}

public Department(string name, List<Position> positions)

{

Name = name;

Positions = positions;

}

public void SetName(string name)

{

Name = name;

}

public string GetName()

{

return Name;

}

public void SetPositions(List<Position> positions)

{

Positions = positions;

}

public List<Position> GetPositions() {

return Positions;

}

public void AddPosition(Position position)

{

Positions.Add(position);

}

public void RemovePosition(Position position)

{

Positions.Remove(position);

}

public class Position

{

private string Name;

private List<Employee> Employees = new List<Employee>();

public Position()

{

}

public Position(string name)

{

Name = name;

}

public Position(string name, List<Employee> employees)

{

Name = name;

Employees = employees;

}

public void SetName(string name)

{

Name = name;

}

public string GetName()

{

return Name;

}

public void SetEmployees(List<Employee> employees)

{

Employees = employees;

}

public List<Employee> GetEmployees() {

return Employees;

}

public void AddEmployee(Employee employee)

{

Employees.Add(employee);

}

public void RemoveEmployee(Employee employee)

{

Employees.Remove(employee);

}

public class Employee

{

private int Age;

private string Name;

public Employee(int age, string name)

{

Age = age;

Name = name;

}

public void SetAge(int age)

{

Age = age;

}

public int GetAge()

{

return Age;

}

public void SetName(string name)

{

Name = name;

}

public string GetName()

{

return Name;

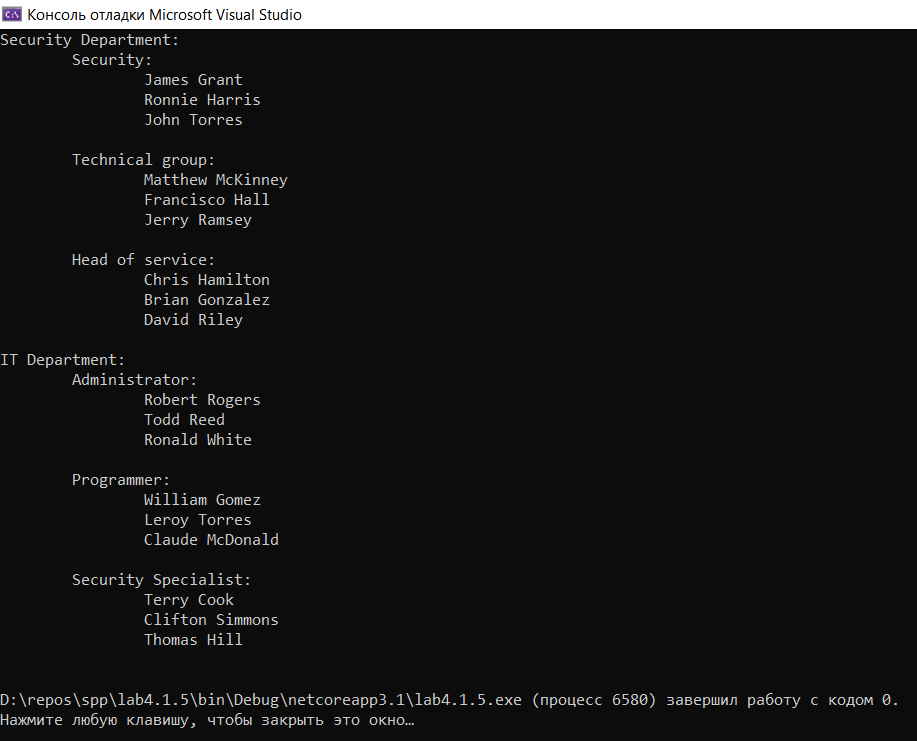
}

}

}

}

}



**Код программы (задание 2):**

Program.cs

using System;

using System.Collections.Generic;

namespace lab4.\_2.\_5

{

class Program

{

static void Main(string[] args)

{

Paragraph par1;

List<Word> words = new List<Word>();

Random rand = new Random();

string[] temp = { "da", "net", "ladno", "vozmojno", "pochemu"};

for (int i = 0; i < 10; i++)

{

string str = temp[rand.Next(0, 5)];

Word word = new Word(str, str.Length);

words.Add(word);

par1 = new Paragraph(words[i]);

par1.Print();

}

}

}

}

Paragraph.cs

using System;

using System.Collections.Generic;

using System.Text;

namespace lab4.\_2.\_5

{

class Word

{

public string name;

public int length;

public Word(string n, int l)

{

name = n;

length = l;

}

}

class Paragraph

{

int NumOfParagraph;

Word word;

public Paragraph(Word someWord)

{

NumOfParagraph = 1;

word = someWord;

}

public void Print()

{

Console.WriteLine($"Номер параграфа: {NumOfParagraph}. Слово: название - {word.name}, длина - {word.length}");

}

public void ChangeWord(string n, int l)

{

word.name = n;

word.length = l;

}

public void ChangeParagraph(int num)

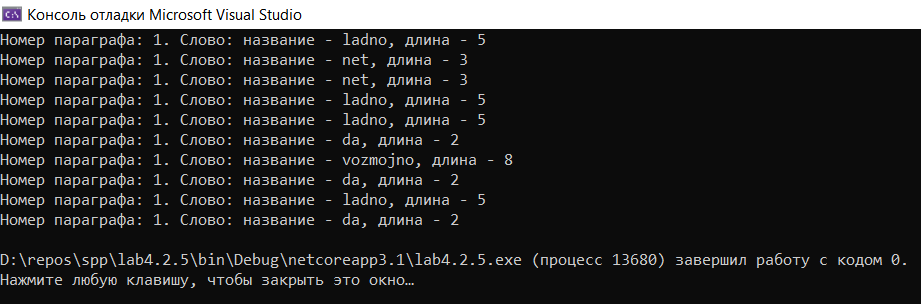
{

NumOfParagraph = num;

}

}

}



**Код программы (задание 3):**

Program.cs

using System;

namespace lab4.\_3.\_5

{

class Program

{

static void Main(string[] args)

{

Catalog Catalog = new Catalog();

Catalog.AddBook(new Book(2008, "Fahrenheit 451", "Ray Bradbury"));

Catalog.AddBook(new Book(1949, "1984", "George Orwell"));

Catalog.AddBook(new Book(2006, "The Master and Margarita", "Michael Bulgakov"));

Catalog.AddBook(new Book(2003, "Shantaram", "Gregory David Roberts"));

Catalog.AddBook(new Book(2002, "The Lovely Bones", "Alice Sebold"));

Library Library = new Library();

Library.SetCatalog(Catalog);

Librarian Librarian = new Librarian(23, "Michael Hunt", Library);

Administrator Administrator = new Administrator(31, "Bill Ford", Library);

Reader Reader = new Reader(17, "Charles Holloway");

Library.AddLibrarian(Librarian);

Library.AddAdministrator(Administrator);

Library.AddReader(Reader);

Librarian.Work();

Administrator.Work();

if (Reader.RequestBook(Library, new Book(1949, "1984", "George Orwell")))

{

Console.WriteLine("Book successfully taken");

}

else

{

Console.WriteLine("Can not take a book");

}

Library.Update();

}

}

}

Administrator.cs

namespace lab4.\_3.\_5

{

public class Administrator : WorkingPerson

{

private Library library = new Library();

public Administrator(int age, string name, Library library) : base (age, name)

{

this.library = library;

}

public void SetLibrary(Library library)

{

this.library = library;

}

public Library GetLibrary()

{

return library;

}

public void AddToBack(Reader reader)

{

library.AddToBack(reader);

}

public void Update()

{

foreach (Order order in library.GetOrders())

{

if (!library.CheckDeadline(order))

{

this.AddToBack(order.GetReader());

}

}

}

}

}

Book.cs

using System;

namespace lab4.\_3.\_5

{

public class Book

{

private int Year = 0;

private string Title, Author;

public Book(int year, string title, string author)

{

Year = year;

Title = title;

Author = author;

}

public void SetYear(int year)

{

Year = year;

}

public int GetYear()

{

return Year;

}

public void SetTitle(string title)

{

Title = title;

}

public string GetTitle()

{

return Title;

}

public void SetAuthor(string author)

{

Author = author;

}

public string GetAuthor()

{

return Author;

}

public bool \_Equals(object c\_Other)

{

if (GetYear().Equals(((Book)c\_Other).GetYear()) && GetTitle().Equals(((Book)c\_Other).GetTitle()) && GetAuthor().Equals(((Book)c\_Other).GetAuthor()))

{

return true;

}

return false;

}

}

}

Catalog.cs

using System.Collections.Generic;

namespace lab4.\_3.\_5

{

public class Catalog

{

private List<Book> Books = new List<Book>();

public Catalog()

{

}

public Catalog(List<Book> book)

{

Books = book;

}

public void SetBooks(List<Book> book)

{

Books = book;

}

public List<Book> GetBooks()

{

return Books;

}

public void AddBook(Book book)

{

Books.Add(book);

}

public void RemoveBook(Book book)

{

Books.Remove(book);

}

public bool BookSearch(Book book)

{

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

{

if (Books[i].GetAuthor() == book.GetAuthor() && Books[i].GetTitle() == book.GetTitle() && Books[i].GetYear() == book.GetYear())

{

return true;

}

}

return false;

}

public Book GiveBook(Book book)

{

int index = -1;

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

{

if (Books[i].GetAuthor() == book.GetAuthor() && Books[i].GetTitle() == book.GetTitle() && Books[i].GetYear() == book.GetYear())

{

index = i;

}

}

if (index == -1)

{

return null;

}

Book result = Books[index];

RemoveBook(book);

return result;

}

}

}

Librarian.cs

namespace lab4.\_3.\_5

{

public class Librarian : WorkingPerson

{

private Library Library = new Library();

public Librarian(int age, string name, Library library) : base(age, name)

{

Library = library;

}

public void SetLibrary(Library library)

{

Library = library;

}

public Library GetLibrary()

{

return Library;

}

public void AddOrder(Order order)

{

Library.AddOrder(order);

}

}

}

Library.cs

using System.Collections.Generic;

namespace lab4.\_3.\_5

{

public class Library

{

private Catalog Catalog = new Catalog();

private List<Administrator> Administrators = new List<Administrator>();

private List<Librarian> Librarians = new List<Librarian>();

private List<Reader> Readers = new List<Reader>(), BlackList = new List<Reader>();

private List<Order> Orders = new List<Order>();

private string Deadline = new string("20/12/2021");

public Library()

{

}

public Library(Catalog catalog)

{

Catalog = catalog;

}

public Library(Catalog catalog, List<Administrator> administrators)

{

Catalog = catalog;

Administrators = administrators;

}

public Library(Catalog catalog, List<Administrator> administrators, List<Librarian> librarians)

{

Catalog = catalog;

Administrators = administrators;

Librarians = librarians;

}

public Library(Catalog catalog, List<Administrator> administrators, List<Librarian> librarians, List<Reader> readers)

{

Catalog = catalog;

Administrators = administrators;

Librarians = librarians;

Readers = readers;

}

public Library(Catalog catalog, List<Administrator> administrators, List<Librarian> librarians, List<Reader> readers, List<Reader> blackList)

{

Catalog = catalog;

Administrators = administrators;

Librarians = librarians;

Readers = readers;

BlackList = blackList;

}

public Library(Catalog catalog, List<Administrator> administrators, List<Librarian> librarians, List<Reader> readers, List<Reader> blackList, List<Order> orders)

{

Catalog = catalog;

Administrators = administrators;

Librarians = librarians;

Readers = readers;

BlackList = blackList;

Orders = orders;

}

public void SetCatalog(Catalog catalog)

{

Catalog = catalog;

}

public Catalog GetCatalog()

{

return Catalog;

}

public void SetAdministrators(List<Administrator> administrators)

{

Administrators = administrators;

}

public List<Administrator> GetAdministrators()

{

return Administrators;

}

public void SetLibrarians(List<Librarian> librarians)

{

Librarians = librarians;

}

public List<Librarian> GetLibrarians()

{

return Librarians;

}

public void SetReaders(List<Reader> readers)

{

Readers = readers;

}

public List<Reader> GetReaders()

{

return Readers;

}

public void SetBlackList(List<Reader> blackList)

{

BlackList = blackList;

}

public List<Reader> GetBlackList()

{

return BlackList;

}

public void SetOrders(List<Order> orders)

{

Orders = orders;

}

public List<Order> GetOrders()

{

return Orders;

}

public void AddBook(Book book)

{

Catalog.AddBook(book);

}

public void RemoveBook(Book book)

{

Catalog.RemoveBook(book);

}

public void AddAdministrator(Administrator administrator)

{

Administrators.Add(administrator);

}

public void RemoveAdministrator(Administrator administrator)

{

Administrators.Remove(administrator);

}

public void AddLibrarian(Librarian librarian)

{

Librarians.Add(librarian);

}

public void RemoveLibrarian(Librarian librarian)

{

Librarians.Remove(librarian);

}

public void AddReader(Reader reader)

{

Readers.Add(reader);

}

public void RemoveReader(Reader reader)

{

Readers.Remove(reader);

}

public void AddToBack(Reader reader)

{

BlackList.Add(reader);

}

public void RemoveFromBlackList(Reader reader)

{

BlackList.Remove(reader);

}

public void AddOrder(Order order)

{

Orders.Add(order);

}

public void RemoveOrder(Order order)

{

Orders.Remove(order);

}

public bool BookSearch(Book book)

{

return Catalog.BookSearch(book);

}

public string GenerateDeadline()

{

return Deadline;

}

public bool CheckDeadline(Order order)

{

return Deadline.Equals(order.GetDeadline());

}

public Order ProcessOrder(Reader reader, Book book)

{

Book orderedBook = Catalog.GiveBook(book);

if (orderedBook == null)

{

return null;

}

string deadline = GenerateDeadline();

Order order = new Order(orderedBook, reader, deadline);

bool processed = false;

while (!processed)

{

foreach (Librarian librarian in Librarians)

{

if (librarian.IsWorking())

{

librarian.AddOrder(order);

processed = true;

break;

}

}

}

Orders.Add(order);

return order;

}

public void Update()

{

bool updated = false;

while (!updated)

{

foreach (Administrator administrator in Administrators)

{

if (administrator.IsWorking())

{

administrator.Update();

updated = true;

break;

}

}

}

}

}

}

Order.cs

namespace lab4.\_3.\_5

{

public class Order

{

private Book Book = null;

private Reader Reader = null;

private string Deadline;

public Order(Book book, Reader reader, string deadline)

{

Book = book;

Reader = reader;

Deadline = deadline;

}

public void SetBook(Book book)

{

Book = book;

}

public Book GetBook()

{

return Book;

}

public void SetReader(Reader reader)

{

Reader = reader;

}

public Reader GetReader()

{

return Reader;

}

public void SetDeadline(string deadline)

{

Deadline = deadline;

}

public string GetDeadline()

{

return Deadline;

}

}

}

Person.cs

namespace lab4.\_3.\_5

{

public class Person

{

private int Age = 0;

private string Name;

public Person(int age, string name)

{

Age = age;

Name = name;

}

public void f\_set\_age(int age)

{

Age = age;

}

public int f\_get\_age()

{

return Age;

}

public void f\_set\_name(string name)

{

Name = name;

}

public string f\_get\_name()

{

return Name;

}

}

}

Reader.cs

using System.Collections.Generic;

namespace lab4.\_3.\_5

{

public class Reader : Person

{

private List<Order> Orders = new List<Order>();

public Reader(int Age, string Name) : base(Age, Name)

{

}

public Reader(int Age, string Name, List<Order> Orders) : base(Age, Name)

{

this.Orders = Orders;

}

public void SetOrders(List<Order> orders)

{

Orders = orders;

}

public List<Order> GetOrders()

{

return Orders;

}

public void AdrOrder(Order order)

{

Orders.Add(order);

}

public void RemoveOrder(Order order)

{

Orders.Remove(order);

}

public bool RequestBook(Library library, Book book)

{

if (library.BookSearch(book))

{

Order order = library.ProcessOrder(this, book);

if (order == null)

{

return false;

}

AdrOrder(order);

return true;

}

return false;

}

}

}

Worker.cs

namespace lab4.\_3.\_5

{

public interface Worker

{

public void Work();

public bool IsWorking();

public void f\_stop\_working();

}

}

WorkingPerson.cs

namespace lab4.\_3.\_5

{

public class WorkingPerson : Person, Worker

{

private bool isWorking = false;

public WorkingPerson(int Age, string Name) : base(Age, Name)

{

}

public void Work()

{

isWorking = true;

}

public bool IsWorking()

{

return isWorking;

}

public void f\_stop\_working()

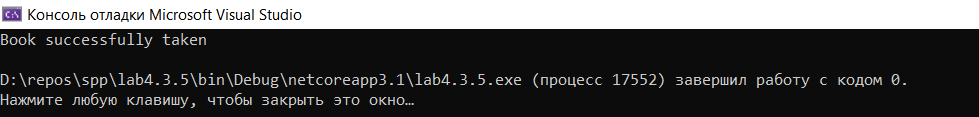
{

isWorking = false;

}

}

}



Вывод: в ходе выполнения лабораторной работы приобрел практические навыки в области объектно-ориентированного проектирования.