Министерство образования Республики Беларусь

Учреждение образования

«Брестский государственный технический университет»

Кафедра ИИТ

Лабораторная работа №4

По дисциплине: «CПП»

Выполнил:

Студент 3 курса

Группы ПО-3

Новикевич А.А.

Проверил:

Монтик Н.С.

2020

Лабораторная работа №4

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

**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 которогоможно хранить информацию обо всех должностях отдела и обо всех сотрудниках, когда-либо занимавших конкретную должность.

Код программы:

using System;

using System.Collections.Generic;

public class Department

{

private String name;

private int count;

private class Position

{

public String title;

public int salary;

public override string ToString()

{

return "Position: " + title + "\nSalary: " + salary + "$\n";

}

}

private class Employee

{

public String name;

public String surname;

public int age;

public String employeePosition;

public int employeeSalary;

public override string ToString()

{

return "Name" + name + "\nSurname: " + surname + "\nAge: " + age + "\nPosition: " + employeePosition + "\nSalary: " + employeeSalary + "\n";

}

}

List<Position> positions = new List<Position>();

public void AddPosition(String \_title, int \_salary)

{

Position pos = new Position();

pos.title = \_title;

pos.salary = \_salary;

positions.Add(pos);

}

public void show()

{

Console.WriteLine("ALL positions: ");

Console.WriteLine();

foreach(Position pos in positions)

{

Console.WriteLine($"{pos.ToString()}");

}

Console.WriteLine("ALL employees: ");

Console.WriteLine();

foreach (Employee employee in employees)

{

Console.WriteLine($"{employee.ToString()}");

}

}

List<Employee> employees = new List<Employee>();

public void AddEmployee (String \_name, String \_surname, int \_age, string \_employeePosition)

{

foreach(Position pos in positions)

{

if (pos.title == \_employeePosition)

{

Employee employee = new Employee();

employee.name = \_name;

employee.surname = \_surname;

employee.age = \_age;

employee.employeePosition = \_employeePosition;

employee.employeeSalary = pos.salary;

employees.Add(employee);

}

}

}

}

namespace ConsoleApp1

{

class Program

{

static void Main(string[] args)

{

Department department = new Department();

department.AddPosition("student", 2000);

department.AddPosition("seller", 5000);

department.AddEmployee("Artyom", "Novikevich", 20, "student");

department.AddEmployee("Gena", "Bukin", 43, "seller");

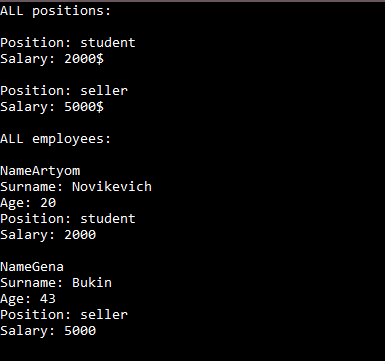
department.show();

}

}

}

Результат выполнения:



**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.

Создать класс Абзац, используя класс Слово.

Код программы:

using System;

using System.Collections.Generic;

class Word

{

private String word;

public Word(String word)

{

this.word = word;

}

public String getWord()

{

return word;

}

}

class Paragraph

{

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

public void addWord(Word word)

{

words.Add(word);

}

public void printALL()

{

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

Console.WriteLine(words[i].getWord() + " ");

}

}

namespace lab4\_2

{

class Program

{

static void Main(string[] args)

{

Word word1 = new Word("HELLO");

Word word2 = new Word("C#");

Word word3 = new Word("from");

Word word4 = new Word("React");

Paragraph paragraph1 = new Paragraph();

paragraph1.addWord(word1);

paragraph1.addWord(word2);

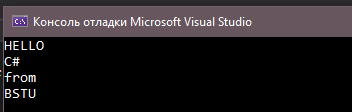
paragraph1.addWord(word3);

paragraph1.addWord(word4);

paragraph1.printALL();

}

}



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:

using System;

using System.Collections.Generic;

namespace ConsoleApp2

{

class Program

{

static void Main(string[] args)

{

String[] bookNames = { "Dead inside", "BSTU MAP", "Dead souls" };

Library library = new Library();

//create books

Book[] books = { new Book(bookNames[0], "Bulgakov", 12),

new Book(bookNames[1], "Gogol", 3),

new Book(bookNames[2], "Tolstoy", 1) };

//create readers

Reader[] readers = { new Reader("Gena Bukin", new DateTime(2003,03,13)),

new Reader("Artyom Nov", new DateTime(2012,06,22) ),

new Reader("Artemij Novik", new DateTime(2020,04,03)) ,

new Reader("Tema Novikecih", new DateTime(2001,11,23))};

//add books in arrayList

foreach (Book book in books)

{

library.addBook(book);

}

//add readers in arrayList

foreach (Reader reader in readers)

{

library.addReader(reader);

}

library.showAll();

Console.WriteLine("\nTry to add Bulgakov to Gena Bukin\n");

library.createOrder(readers[0], books[0]);

Console.WriteLine("\nTry to add Gogol to Gena Bukin\n");

library.createOrder(readers[0], books[1]);

Console.WriteLine("\nTry to add Tolstoi to Artyom Nov\n");

library.createOrder(readers[1], books[2]);

Console.WriteLine("\nTry to add Tolstoi to Artemij Novik\n");

library.createOrder(readers[2], books[2]);

Console.WriteLine("\nTry to add Gogol to Artemij Novik\n");

library.createOrder(readers[2], books[1]);

Console.WriteLine("\nTry to add Tolstoi to Tema Novikecih\n");

library.createOrder(readers[3], books[2]);

foreach (Reader reader in readers)

{

Console.WriteLine("Reader name: " + reader.getName()

+ "\nDate: " + reader.getDate() + "\n");

}

library.showBlackList();

}

}

}

interface Show

{

void showAll();

void createOrder(Reader reader, Book book);

void showBlackList();

void removeBook(Book book);

void removeOrder(Order order);

}

class Library:Show

{

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

private List<Reader> readers = new List<Reader>();

private List<Reader> blackList = new List<Reader>();

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

public void addBook(Book book)

{

books.Add(book);

}

public void createOrder(Reader reader, Book book)

{

if (reader.checkDate(DateTime.Now, reader.getDate()) > 20)

{

blackList.Add(reader);

return;

}

foreach (Book currentBook in books)

{

if (currentBook.getId() == book.getId() && currentBook.getNumber() > 0)

{

Order order = new Order(book.getId(), reader.getId(), book.getTitle());

orders.Add(order);

removeBook(currentBook);

reader.addBook(currentBook);

//removeOrder(order);

}

}

}

public void showBlackList()

{

Console.WriteLine("\nBlack list\n");

foreach (Reader reader in blackList)

{

Console.WriteLine("\nReader name: " + reader.getName() + "\nDate: " + reader.getDate());

}

}

public void showAll()

{

Console.WriteLine("\nAll books in the library:\n");

foreach (Book book in books)

{

Console.WriteLine("\nBooks title: " + book.getTitle()

+ "\nAuthor: " + book.getAuthor()

+ "\nCount: " + book.getNumber() + "\n");

}

}

public void removeBook(Book book)

{

book.setNumber(book.getNumber() - 1);

}

public void removeOrder(Order order)

{

orders.Remove(order);

}

public void addReader(Reader reader)

{

readers.Add(reader);

}

}

class Book

{

private int id;

private String title;

private String author;

private int number;

private static int booksCount = 1;

public Book(String title, String author, int number)

{

id = booksCount++;

setTitle(title);

setAuthor(author);

setNumber(number);

}

public int getId()

{

return id;

}

public int getNumber()

{

return number;

}

public void setNumber(int number)

{

this.number = number;

}

public String getAuthor()

{

return author;

}

public void setAuthor(String author)

{

this.author = author;

}

public String getTitle()

{

return title;

}

public void setTitle(String title)

{

this.title = title;

}

}

class Reader {

private int id;

private String name;

private DateTime bookDate = DateTime.Now;

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

private static int readersCount = 1;

public Reader(String name, DateTime date)

{

id = readersCount++;

setName(name);

setDate(date);

}

public int getId()

{

return id;

}

public String getName()

{

return name;

}

public void setName(String name)

{

this.name = name;

}

public void setDate(DateTime date)

{

bookDate = date;

}

public DateTime getDate()

{

return bookDate;

}

public long checkDate(DateTime date, DateTime bookDate)

{

TimeSpan difference = date - bookDate;

long days = (long)difference.TotalDays;

return days;

}

public void addBook(Book book)

{

books.Add(book);

}

public void showAll()

{

foreach (Book book in books){

if (books.Count == 0) { Console.WriteLine("There is not any books"); }

else

{

Console.WriteLine("Book name: " + book.getTitle() + "\n" +"Author: " + book.getAuthor() + "\n");

}

}

}

}

class Order

{

private int id;

private int bookId;

private int readerId;

private String bookTitle;

private static int ordersCount = 1;

public Order(int bookId, int readerId, String bookTitle)

{

id = ordersCount++;

setBookId(bookId);

setReaderId(readerId);

setBookTitle(bookTitle);

}

public int getReaderId()

{

return readerId;

}

public void setReaderId(int readerId)

{

this.readerId = readerId;

}

public int getBookId()

{

return bookId;

}

public void setBookId(int bookId)

{

this.bookId = bookId;

}

public void setBookTitle(String bookTitle)

{

this.bookTitle = bookTitle;

}

public String getBookTitle()

{

return bookTitle;

}

}

