МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ

«БРЕСТСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

Кафедра интеллектуальных информационных технологий

Отчет по лабораторной работе №4

По дисциплине «Современные платформы программирования»

Специальность ПО-8

Выполнил:

Серко А.С.

студент группы ПО-8

Проверил:

ст. преп. кафедры ИИТ,

«\_\_»\_\_\_\_\_\_\_\_\_\_\_2024 г.

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

**Вариант 19**

Реализовать указанный класс, включив в него вспомогательный внутренний класс или классы.

Реализовать 2-3 метода (на выбор). Продемонстрировать использование реализованных классов.

**Задание 1**

1) Создать класс Notepad (записная книжка) с внутренним классом или классами, с помощью

объектов которого могут храниться несколько записей на одну дату.

Выполнение

**Код программы**

Program.cs

using Task1;

Notepad myNotepad = new Notepad();

myNotepad.AddNote(DateTime.Now, "Note 1");

myNotepad.AddNote(DateTime.Now, "Note 2");

myNotepad.AddNote(DateTime.Now.AddDays(1), "Note 3");

Console.WriteLine("Today Notes:");

List<string> todaysNotes = myNotepad.FindNotes(DateTime.Now);

foreach (var note in todaysNotes)

{

Console.WriteLine("- " + note);

}

Console.WriteLine("All Notes:");

myNotepad.PrintAllNotes();

Notepad.cs

namespace Task1

{

public class Notepad

{

private class Note

{

public DateTime Date { get; }

public string Text { get; }

public Note(DateTime date, string text)

{

Date = date;

Text = text;

}

}

private Dictionary<DateTime, List<Note>> notes;

public Notepad()

{

notes = new Dictionary<DateTime, List<Note>>();

}

public void AddNote(DateTime date, string text)

{

if (!notes.ContainsKey(date))

{

notes[date] = new List<Note>();

}

notes[date].Add(new Note(date, text));

}

public List<string> FindNotes(DateTime date)

{

if (notes.ContainsKey(date))

{

List<string> foundNotes = new List<string>();

foreach (var note in notes[date])

{

foundNotes.Add(note.Text);

}

return foundNotes;

}

else

{

return new List<string>();

}

}

public void PrintAllNotes()

{

foreach (var entry in notes)

{

Console.WriteLine($"Date: {entry.Key.ToShortDateString()}");

foreach (var note in entry.Value)

{

Console.WriteLine($"- {note.Text}");

}

Console.WriteLine();

}

}

}

}

**Спецификация вывода**

**Пример**

Изображение выглядит как текст, Шрифт, снимок экрана, дизайн

Автоматически созданное описание

**Задание 2.**

7) Создать класс Страница, используя классы Строка, Слово.

Выполнение

**Код программы**

**Program.cs**

using Task2;

List<Line> linesOnPage = new List<Line>();

Page page = new Page(linesOnPage);

Console.WriteLine("Enter words for a line (separated by space), or type 'done' to finish:");

bool addingWords = true;

while (addingWords)

{

string input = Console.ReadLine();

if (input.ToLower() == "done")

{

addingWords = false;

}

else

{

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

string[] wordStrings = input.Split(' ');

foreach (string wordString in wordStrings)

{

wordsInLine.Add(new Word(wordString));

}

Line line = new Line(wordsInLine);

page.AddLine(line);

}

}

Console.WriteLine("Number of lines on the page: " + page.LineCount());

**Page.cs**

namespace Task2

{

internal class Page

{

private List<Line> lines;

public Page(List<Line> lines)

{

this.lines = lines;

}

public int LineCount()

{

return lines.Count;

}

public List<Line> GetLines()

{

return lines;

}

public void AddLine(Line line)

{

lines.Add(line);

}

}

}

**Line.cs**

namespace Task2

{

internal class Line

{

private List<Word> words;

public Line(List<Word> words)

{

this.words = words;

}

public int WordCount()

{

return words.Count;

}

public List<Word> GetWords()

{

return words;

}

public void AddWord(Word word)

{

words.Add(word);

}

}

}

**Word.cs**

namespace Task2

{

internal class Word

{

private string word;

public Word(string word)

{

this.word = word;

}

public int Length()

{

return word.Length;

}

public string GetWord()

{

return word;

}

}

}

**Спецификация вывода**

**Пример**

**Изображение выглядит как текст, снимок экрана, Шрифт, черный

Автоматически созданное описание**

**Задание 3**

Построить модель программной системы с применением отношений (обобщения, агрегации, ассо-

циации, реализации) между классами. Задать атрибуты и методы классов. Реализовать (если необ-

ходимо) дополнительные классы. Продемонстрировать работу разработанной системы.

Выполнение

**Код программы**

**Admin.cs**

namespace Task3

{

internal class Admin

{

public int Id { get; }

public string Name { get; }

public List<Sale> Sales { get; }

public List<Customer> Blacklist { get; }

public Admin(int id, string name)

{

Id = id;

Name = name;

Sales = new List<Sale>();

Blacklist = new List<Customer>();

}

public void AddProductInfo(Product product)

{

Console.WriteLine($"Product added: {product.GetProductInfo()}");

}

public void RegisterSale(Sale sale)

{

Sales.Add(sale);

Console.WriteLine($"Sale registered: {sale.GetSaleInfo()}");

}

public void AddToBlacklist(Customer customer)

{

Blacklist.Add(customer);

Console.WriteLine($"Customer {customer.Name} added to the blacklist.");

}

public bool checkOrderStatus(Order order, Customer customer)

{

if (order.Status != "Paid")

{

return false;

}

return true;

}

}

}

Blacklist.cs

namespace Task3

{

internal class Blacklist

{

public List<Customer> Customers { get; }

public Blacklist()

{

Customers = new List<Customer>();

}

public void AddToBlacklist(Customer customer)

{

Customers.Add(customer);

Console.WriteLine($"Customer {customer.Name} added to the blacklist.");

}

}

}

**Customer.cs**

namespace Task3

{

internal class Customer

{

public int Id { get; }

public string Name { get; }

public string Address { get; }

public string Email { get; }

public List<Order> Orders { get; }

public Customer(int id, string name, string address, string email)

{

Id = id;

Name = name;

Address = address;

Email = email;

Orders = new List<Order>();

}

public void PlaceOrder(Order order)

{

Orders.Add(order);

Console.WriteLine($"Order {order.Id} has been placed by customer {Name}.");

}

public void Pay(Order order)

{

order.Status = "Paid";

Console.WriteLine($"Order {order.Id} has been paid by customer {Name}.");

}

}

}

Order.cs

namespace Task3

{

internal class Order

{

public int Id { get; }

public List<Product> Products { get; }

public string Status { get; set; }

public double TotalAmount { get; set; }

public Order(int id)

{

Id = id;

Products = new List<Product>();

Status = "Pending";

TotalAmount = 0.0;

}

public void AddProduct(Product product)

{

Products.Add(product);

TotalAmount += product.Price;

}

}

}

Product.cs

namespace Task3

{

internal class Product

{

public int Id { get; }

public string Name { get; }

public double Price { get; }

public string Description { get; }

public Product(int id, string name, double price, string description)

{

Id = id;

Name = name;

Price = price;

Description = description;

}

public string GetProductInfo()

{

return $"ID: {Id}, Name: {Name}, Price: {Price}$, Description: {Description}";

}

}

}

Sale.cs

namespace Task3

{

internal class Sale

{

public int Id { get; }

public DateTime Date { get; }

public double Amount { get; }

public List<Product> ProductsSold { get; }

public Sale(int id, double amount, List<Product> products)

{

Id = id;

Date = DateTime.Now;

Amount = amount;

ProductsSold = products;

}

public string GetSaleInfo()

{

return $"ID: {Id}, Date: {Date}, Amount: {Amount}$";

}

}

}

Program.cs

class Program

{

static void Main(string[] args)

{

Product product1 = new Product(1, "Laptop", 999.99, "Chuwi Gemibook Pro");

Product product2 = new Product(2, "Phone", 499.99, "Xiaomi redmi Note 10");

Order order1 = new Order(1);

order1.AddProduct(product1);

order1.AddProduct(product2);

Customer customer1 = new Customer(1, "Alexandra", "ul. Moskocskaya 68", "alexa@gmail.com");

customer1.PlaceOrder(order1);

customer1.Pay(order1);

Sale sale1 = new Sale(1, order1.TotalAmount, order1.Products);

Admin admin = new Admin(1, "Admin");

order1.PrintProducts();

admin.RegisterSale(sale1);

if (!(admin.checkOrderStatus(order1, customer1)))

{

admin.AddToBlacklist(customer1);

}

}

}

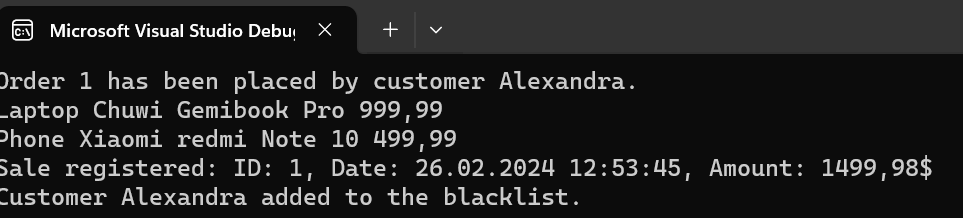
**Спецификация вывода**

**Если заказ был оплачен**

**Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание**

**Если не был оплачен**

****

**Вывод**

Приобрела практические навыки в области объектно-ориентированного проектирования.