**МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ**

**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ**

**ВЫСШЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ**

**«ЮЖНЫЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»**

**ИНЖЕНЕРНО-ТЕХНОЛОГИЧЕСКАЯ АКАДЕМИЯ**

**Институт компьютерных технологий и информационной безопасности**

**Кафедра математического обеспечения и применения ЭВМ**

**Отчет**

По лабораторной работе № 4

На тему: «Разработка объектно-ориентированной модели реальной системы»

По курсу: «Архитектура, технологии и ИС разработки ПО»

Вариант № 8

Выполнили:

студенты гр. КТбо3-6

Гришак В. С.

Проверил:

доцент кафедры МОП ЭВМ

Лутай В. Н.

Таганрог 2022

**Вариант 8**

Библиотека (Library) Включает книги (художественные и технические), находящиеся в 2-х соответствующих отделах, библиотекаря, абонентов. Пояснения: основной объект – книга. Она представлена карточкой, в которой указаны ее выходные данные, отдел (художественный /технический), место нахождения (стеллаж, ячейка), количество экземпляров на текущий момент. Каждый экземпляр книги снабжается вкладышем, в котором библиотекарем отмечается дата выдачи, срок и номер читательского билета абонента библиотеки. При возврате или утере книги абонентом делаются соответствующие записи. При несвоевременном возврате абоненту посылается напоминание. Необходимо написать методы выдачи книг и их возврата, учета книг как выданных, так и находящихся в библиотеке, причем раздельно по отделам. Если книга утеряна, то она списывается (количество экземпляров уменьшается), а абоненту выписывается штраф.

1. **Проанализировать задание.**

Из задания можно выделить следующие сущности:

1. Книга
2. Книжный шкаф
3. Отдел – совокупность книжных шкафов
4. Библиотекарь
5. Читатель
6. Библиотека

Следующие действия:

1. Запросить книгу у Библиотекаря
2. Вернуть книгу Библиотекарю
3. Найти книгу по названию
4. Взять книгу с определённого места
5. Поставить книгу на место
6. **Диаграммы вариантов использования**

Сценарий 1: «Запрос книги»

Основной поток событий:

- Читатель получает список Названий книг, затем выбирает номер интересующей его книги и сообщает его Библиотекарю. Библиотекарь выполняет прецендент «Взять книгу» и делает в книге соответствующие пометки. Читатель получает книгу.

Альтернативные последовательности:

- Если Библиотекарь не нашёл книгу, то даёт Читателю ничто.

Сценарий 2: «Вернуть книгу»

Основной поток событий:

- Читатель даёт Книгу Библиотекарю. Библиотекарь делает соответствующие пометки. Далее библиотекарь выполняет прецендент «Поставить книгу».

Альтернативные последовательности:

- Читатель возвращает ничто. Библиотекарь выписывает штраф. Библиотекарь пересчитывает количество экземпляров.

Сценарий 3: «Оповестить о задержке»

Основной поток событий:

- Библиотекарь ведёт список читателей, бравших список. Как только у читателя выходит срок, то Библиотекарь оповещает Читателя, до тех пор, пока Читатель не придёт в Библиотеку.

Альтернативные последовательности:

Отсутствуют.

Сценарий 4: «Взять книгу»

Основной поток событий:

- Библиотекарь ищет по номеру книги в списке положение в списке книги, которая ещё в наличии. Далее по соответствующему положению (отдел, шкаф, полка, ячейка) берётся Книга и в списке положений делается заметка об отсутствии этой книги. Передача вызвавшему преценденту.

Альтернативные последовательности:

- Если такая книга не нашлась в наличии, то не делается никаких пометок. Передача вызвавшему преценденту.

Сценарий 5: «Поставить книгу»

Основной поток событий:

- Библиотекарь ищет положение Книги по имеющимся данным во вкладыше номеру книги в списке положение в списке книги, которая ещё в наличии. Далее по соответствующему положению (отдел, шкаф, полка, ячейка) Книга ставиться в соответствующую ячейку, полку, шкаф и делается пометка, что книга с таким положение доступна. Передача вызвавшему преценденту.

Альтернативные последовательности:

Отсутствует

1. **Диаграмма классов.**

В последствии была разработана диаграмма классов рисунок 1.

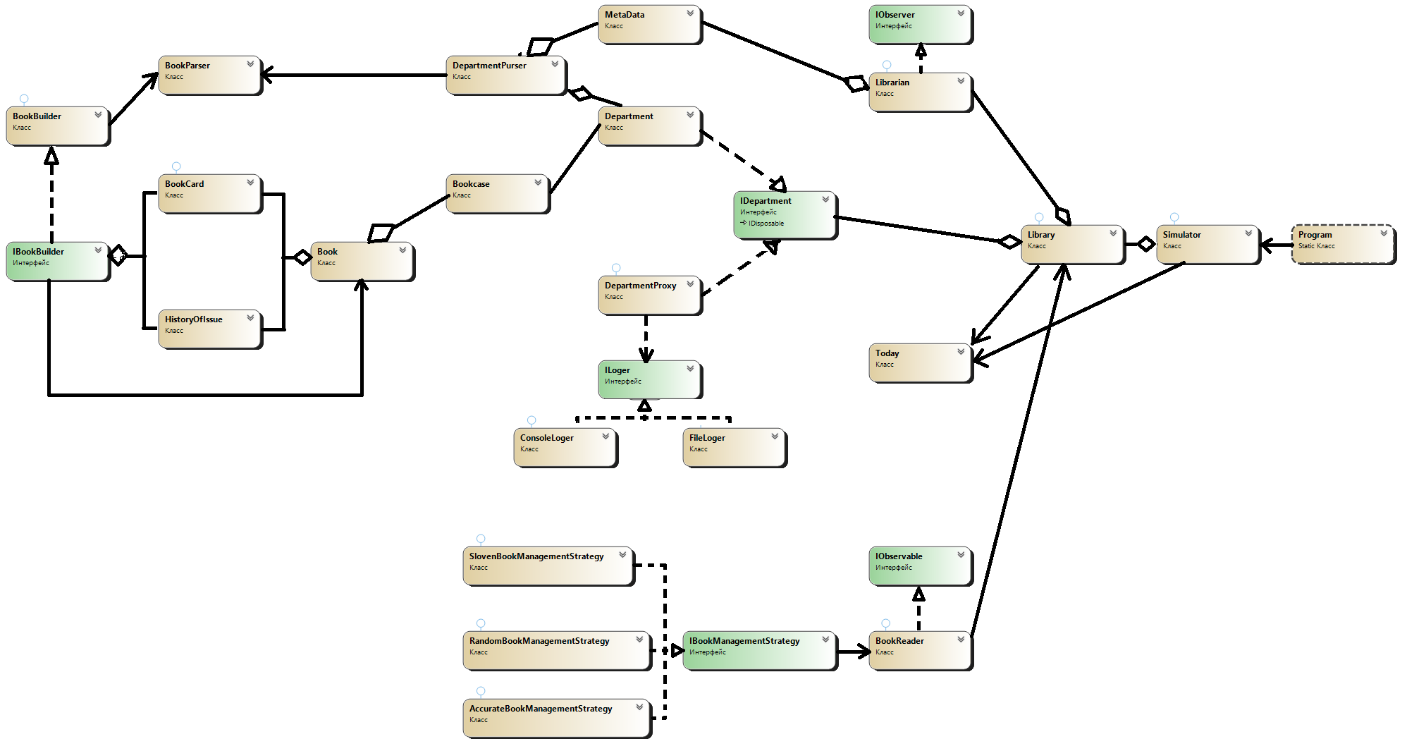


Рисунок 1 – Диаграмма классов

1. **Диаграмма Прецедентов**

Изображение выглядит как текст, ночное небо

Автоматически созданное описание

Рисунок 2 – Диаграмма прецедентов

1. **Диаграмма Последовательности**

Благодаря разработанной диаграмме последовательности, была разработана система. При её разработке были применены следующие паттерны: Стратегия, Строитель, Заместитель, Фасад, Наблюдатель.

**ПРИЛОЖЕНИЕ**

Form1.cs:

using Laba4Pattern.Readeres;

using library\_laba4;

using library\_laba4.Logers;

using System;

using System.Collections.Generic;

using System.Threading;

using System.Windows.Forms;

namespace Laba4Pattern

{

public partial class Form1 : Form

{

Thread backgroundThread;

Simulator simulator;

public Form1()

{

InitializeComponent();

}

private void InitComponent(object sender, EventArgs e)

{

InitSimulator();

start.Enabled = true;

stop.Enabled = true;

init.Enabled = false;

}

private void Run()

{

simulator.Simulate();

}

private void Start(object sender, EventArgs e)

{

if (backgroundThread == null)

{

backgroundThread = new Thread(new ThreadStart(Run));

backgroundThread.Start();

}

}

private void Stop(object sender, EventArgs e)

{

simulator.Stop();

backgroundThread = null;

}

private void InitSimulator()

{

DepartmentPurser artisticPurser = new DepartmentPurser("C:\\Users\\ALEX\\source\\repos\\Laba4Pattern\\Laba4Pattern\\archive\\artistic.txt");

DepartmentPurser technicalPurser = new DepartmentPurser("C:\\Users\\ALEX\\source\\repos\\Laba4Pattern\\Laba4Pattern\\archive\\technical.txt");

ILoger loger = new ConsoleLoger(textBox1);

DepartmentProxy artisticDepartment = new DepartmentProxy(artisticPurser.Parse(), loger);

DepartmentProxy technicalDepartment = new DepartmentProxy(technicalPurser.Parse(), loger);

Librarian librarian = new Librarian(artisticPurser.GetMetaData(), technicalPurser.GetMetaData(),

artisticDepartment, technicalDepartment);

Library library = new Library(artisticDepartment, technicalDepartment, librarian, loger);

List<BookReader> bookReaders = Generate(library, 5);

simulator = new Simulator(loger, library, bookReaders);

}

private static List<BookReader> Generate(Library library, int size)

{

List<BookReader> bookReaders = new List<BookReader>(size);

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

{

IBookManagementStrategy bookManagementStrategy = null;

switch (i % 3)

{

case 0:

bookManagementStrategy = new RandomBookManagementStrategy();

break;

case 1:

bookManagementStrategy = new AccurateBookManagementStrategy();

break;

case 2:

bookManagementStrategy = new SlovenBookManagementStrategy();

break;

default:

break;

}

bookReaders.Add(new BookReader(bookManagementStrategy));

library.AddBookReader(bookReaders[i]);

}

return bookReaders;

}

private void Form1\_FormClosing(Object sender, FormClosingEventArgs e)

{

if (e.CloseReason == CloseReason.UserClosing)

{

if (MessageBox.Show("You want to close Form ?", "Warning", MessageBoxButtons.YesNo)

== DialogResult.Yes)

{

simulator?.Stop();

backgroundThread?.Abort();

// backgroundThread?.Join();

simulator?.Dispose();

Application.Exit();

}

else

{

e.Cancel = true;

}

}

}

}

}

Form1.Designer.cs:

namespace Laba4Pattern

{

partial class Form1

{

/// <summary>

/// Обязательная переменная конструктора.

/// </summary>

private System.ComponentModel.IContainer components = null;

/// <summary>

/// Освободить все используемые ресурсы.

/// </summary>

/// <param name="disposing">истинно, если управляемый ресурс должен быть удален; иначе ложно.</param>

protected override void Dispose(bool disposing)

{

if (disposing && (components != null))

{

components.Dispose();

}

base.Dispose(disposing);

}

#region Код, автоматически созданный конструктором форм Windows

/// <summary>

/// Требуемый метод для поддержки конструктора — не изменяйте

/// содержимое этого метода с помощью редактора кода.

/// </summary>

private void InitializeComponent()

{

System.ComponentModel.ComponentResourceManager resources = new System.ComponentModel.ComponentResourceManager(typeof(Form1));

this.tableLayoutPanel1 = new System.Windows.Forms.TableLayoutPanel();

this.textBox1 = new System.Windows.Forms.TextBox();

this.panel1 = new System.Windows.Forms.Panel();

this.stop = new System.Windows.Forms.Button();

this.start = new System.Windows.Forms.Button();

this.init = new System.Windows.Forms.Button();

this.pictureBox1 = new System.Windows.Forms.PictureBox();

this.tableLayoutPanel1.SuspendLayout();

this.panel1.SuspendLayout();

((System.ComponentModel.ISupportInitialize)(this.pictureBox1)).BeginInit();

this.SuspendLayout();

//

// tableLayoutPanel1

//

this.tableLayoutPanel1.ColumnCount = 2;

this.tableLayoutPanel1.ColumnStyles.Add(new System.Windows.Forms.ColumnStyle(System.Windows.Forms.SizeType.Absolute, 300F));

this.tableLayoutPanel1.ColumnStyles.Add(new System.Windows.Forms.ColumnStyle());

this.tableLayoutPanel1.Controls.Add(this.textBox1, 1, 0);

this.tableLayoutPanel1.Controls.Add(this.panel1, 0, 0);

this.tableLayoutPanel1.Dock = System.Windows.Forms.DockStyle.Fill;

this.tableLayoutPanel1.Location = new System.Drawing.Point(0, 0);

this.tableLayoutPanel1.Name = "tableLayoutPanel1";

this.tableLayoutPanel1.RowCount = 1;

this.tableLayoutPanel1.RowStyles.Add(new System.Windows.Forms.RowStyle(System.Windows.Forms.SizeType.Percent, 100F));

this.tableLayoutPanel1.Size = new System.Drawing.Size(800, 450);

this.tableLayoutPanel1.TabIndex = 0;

//

// textBox1

//

this.textBox1.BackColor = System.Drawing.SystemColors.Info;

this.textBox1.Dock = System.Windows.Forms.DockStyle.Fill;

this.textBox1.Font = new System.Drawing.Font("Microsoft Sans Serif", 14F);

this.textBox1.Location = new System.Drawing.Point(303, 3);

this.textBox1.Multiline = true;

this.textBox1.Name = "textBox1";

this.textBox1.ReadOnly = true;

this.textBox1.ScrollBars = System.Windows.Forms.ScrollBars.Vertical;

this.textBox1.Size = new System.Drawing.Size(494, 444);

this.textBox1.TabIndex = 0;

//

// panel1

//

this.panel1.Controls.Add(this.pictureBox1);

this.panel1.Controls.Add(this.init);

this.panel1.Controls.Add(this.stop);

this.panel1.Controls.Add(this.start);

this.panel1.Dock = System.Windows.Forms.DockStyle.Fill;

this.panel1.Location = new System.Drawing.Point(3, 3);

this.panel1.Name = "panel1";

this.panel1.Size = new System.Drawing.Size(294, 444);

this.panel1.TabIndex = 1;

//

// stop

//

this.stop.AutoSize = true;

this.stop.Cursor = System.Windows.Forms.Cursors.Hand;

this.stop.Enabled = false;

this.stop.FlatStyle = System.Windows.Forms.FlatStyle.Flat;

this.stop.Font = new System.Drawing.Font("Microsoft Sans Serif", 14F);

this.stop.ImageAlign = System.Drawing.ContentAlignment.MiddleLeft;

this.stop.Location = new System.Drawing.Point(15, 355);

this.stop.Name = "stop";

this.stop.Size = new System.Drawing.Size(128, 36);

this.stop.TabIndex = 1;

this.stop.Text = "остановить";

this.stop.UseVisualStyleBackColor = true;

this.stop.Click += new System.EventHandler(this.Stop);

//

// start

//

this.start.AutoSize = true;

this.start.Cursor = System.Windows.Forms.Cursors.Hand;

this.start.Enabled = false;

this.start.FlatStyle = System.Windows.Forms.FlatStyle.Flat;

this.start.Font = new System.Drawing.Font("Microsoft Sans Serif", 14F);

this.start.ImageAlign = System.Drawing.ContentAlignment.MiddleLeft;

this.start.Location = new System.Drawing.Point(15, 301);

this.start.Name = "start";

this.start.Size = new System.Drawing.Size(114, 36);

this.start.TabIndex = 0;

this.start.Text = "запустить";

this.start.UseVisualStyleBackColor = true;

this.start.Click += new System.EventHandler(this.Start);

//

// init

//

this.init.AutoSize = true;

this.init.Cursor = System.Windows.Forms.Cursors.Hand;

this.init.FlatStyle = System.Windows.Forms.FlatStyle.Flat;

this.init.Font = new System.Drawing.Font("Microsoft Sans Serif", 14F);

this.init.ImageAlign = System.Drawing.ContentAlignment.MiddleLeft;

this.init.Location = new System.Drawing.Point(15, 248);

this.init.Name = "init";

this.init.Size = new System.Drawing.Size(152, 36);

this.init.TabIndex = 2;

this.init.Text = "инициировать";

this.init.UseVisualStyleBackColor = true;

this.init.Click += new System.EventHandler(this.InitComponent);

//

// pictureBox1

//

this.pictureBox1.Image = ((System.Drawing.Image)(resources.GetObject("pictureBox1.Image")));

this.pictureBox1.Location = new System.Drawing.Point(3, 43);

this.pictureBox1.Name = "pictureBox1";

this.pictureBox1.Size = new System.Drawing.Size(275, 180);

this.pictureBox1.SizeMode = System.Windows.Forms.PictureBoxSizeMode.Zoom;

this.pictureBox1.TabIndex = 3;

this.pictureBox1.TabStop = false;

//

// Form1

//

this.AutoScaleDimensions = new System.Drawing.SizeF(6F, 13F);

this.AutoScaleMode = System.Windows.Forms.AutoScaleMode.Font;

this.ClientSize = new System.Drawing.Size(800, 450);

this.Controls.Add(this.tableLayoutPanel1);

this.MinimumSize = new System.Drawing.Size(600, 400);

this.Name = "Form1";

this.Text = "библиотека";

this.WindowState = System.Windows.Forms.FormWindowState.Maximized;

this.FormClosing += new System.Windows.Forms.FormClosingEventHandler(this.Form1\_FormClosing);

this.tableLayoutPanel1.ResumeLayout(false);

this.tableLayoutPanel1.PerformLayout();

this.panel1.ResumeLayout(false);

this.panel1.PerformLayout();

((System.ComponentModel.ISupportInitialize)(this.pictureBox1)).EndInit();

this.ResumeLayout(false);

}

#endregion

private System.Windows.Forms.TableLayoutPanel tableLayoutPanel1;

private System.Windows.Forms.TextBox textBox1;

private System.Windows.Forms.Panel panel1;

private System.Windows.Forms.Button stop;

private System.Windows.Forms.Button start;

private System.Windows.Forms.Button init;

private System.Windows.Forms.PictureBox pictureBox1;

}

}

Program.cs:

using System;

using System.Windows.Forms;

namespace Laba4Pattern

{

internal static class Program

{

/// <summary>

/// Главная точка входа для приложения.

/// </summary>

[STAThread]

static void Main()

{

Application.EnableVisualStyles();

Application.SetCompatibleTextRenderingDefault(false);

Application.Run(new Form1());

}

}

}

Simulator.cs:

using library\_laba4;

using library\_laba4.Logers;

using System;

using System.Collections.Generic;

using System.Threading;

namespace Laba4Pattern

{

public class Simulator : IDisposable

{

ILoger loger;

Library library;

bool isLoop = true;

List<BookReader> bookReaders;

public Simulator(ILoger loger, Library library, List<BookReader> bookReaders)

{

this.loger = loger;

this.library = library;

this.bookReaders = bookReaders;

}

public void Simulate()

{

isLoop = true;

while (isLoop)

{

for (int j = 0; j < bookReaders.Count; j++)

{

bookReaders[j].SimulateDay();

}

loger.Log($"=== {Today.Data.ToShortDateString()} ===");

library.SimulateDay();

Thread.Sleep(1000);

Today.NextDay();

}

}

public void Stop()

{

isLoop = false;

}

public void Dispose()

{

library.CloseLibrary();

library.CloseLibrary();

}

}

}

Today.cs:

using System;

namespace library\_laba4

{

public class Today

{

public static DateTime Data { private set; get; } = DateTime.Now;

public static void NextDay()

{

Data = Data.AddDays(1);

}

}

}

Book.cs:

using System;

namespace library\_laba4

{

public class Book

{

private readonly HistoryOfIssue \_historyOfIssue;

public readonly BookCard BookCard;

public Book(BookCard bookCard, HistoryOfIssue historyOfIssue)

{

\_historyOfIssue = historyOfIssue;

BookCard = bookCard;

}

public void SetPosition(int bookсaseNumber, int shelfNumber, int cellNumber)

{

BookCard.SetPosition(bookсaseNumber, shelfNumber, cellNumber);

}

public string Name => BookCard.Name;

public string Author => BookCard.Author;

public int Department => BookCard.Department;

public (int, int, int) GetPosition() => BookCard.GetPosition();

public void IssueRecord(DateTime dateOfIssue, int days, Guid userID)

=> \_historyOfIssue.IssueRecord(dateOfIssue, days, userID);

public void ReturnRecord(DateTime dateOfIssue, Guid userID)

=> \_historyOfIssue.ReturnRecord(dateOfIssue, userID);

public string GetHistory() => \_historyOfIssue.GetHistory();

public override string ToString() => BookCard.ToString();

public string GetFullInformation()

=> $"Основная информация\n{BookCard.GetFullInformation()}История\n{GetHistory()}";

}

}

BookBuilder.cs:

using System;

using System.Collections.Generic;

namespace library\_laba4

{

public class BookBuilder : IBookBuilder

{

private string \_bookName;

private string \_author;

private int \_department;

private int \_bookсaseNumber;

private int \_shelfNumber;

private int \_cellNumber;

private int \_instanceNumber;

private IList<(DateTime, int, Guid)> \_issueRecord;

private IList<(DateTime, Guid)> \_returnRecord;

public BookBuilder()

{

\_bookName = "";

\_author = "";

\_department = 0;

\_bookсaseNumber = 0;

\_shelfNumber = 0;

\_cellNumber = 0;

\_issueRecord = new List<(DateTime, int, Guid)>();

\_returnRecord = new List<(DateTime, Guid)>();

}

public BookBuilder SetBookName(string bookName)

{

\_bookName = bookName;

return this;

}

public BookBuilder SetAuthor(string author)

{

\_author = author;

return this;

}

public BookBuilder SetBookDepartment(int department)

{

\_department = department;

return this;

}

public BookBuilder SetBookсaseNumber(int bookсaseNumber)

{

\_bookсaseNumber = bookсaseNumber;

return this;

}

public BookBuilder SetShelfNumber(int shelfNumber)

{

\_shelfNumber = shelfNumber;

return this;

}

public BookBuilder SetCellNumber(int cellNumber)

{

\_cellNumber = cellNumber;

return this;

}

public BookBuilder SetInstanceNumber(int instanceNumber)

{

\_instanceNumber = instanceNumber;

return this;

}

public BookBuilder AddIssueRecord(DateTime dateOfIssue, int days, Guid userID)

{

\_issueRecord.Add((dateOfIssue, days, userID));

return this;

}

public BookBuilder AddReturnRecord(DateTime dateOfIssue, Guid userID)

{

\_returnRecord.Add((dateOfIssue, userID));

return this;

}

private BookCard BuildBookCard()

{

BookCard bookCard = new BookCard(\_department, \_bookName, \_author);

bookCard.BookсaseNumber = \_bookсaseNumber;

bookCard.ShelfNumber = \_shelfNumber;

bookCard.СellNumber = \_cellNumber;

bookCard.InstanceNumber = \_instanceNumber;

return bookCard;

}

private HistoryOfIssue BuildHistoryOfIssue()

{

HistoryOfIssue historyOfIssue = new HistoryOfIssue();

int i = 0;

for (; i < \_returnRecord.Count; i++)

{

historyOfIssue.IssueRecord(\_issueRecord[i]);

historyOfIssue.ReturnRecord(\_returnRecord[i]);

}

if (i == \_issueRecord.Count - 1)

{

historyOfIssue.IssueRecord(\_issueRecord[i]);

}

return historyOfIssue;

}

public Book Build()

{

BookCard bookCard = BuildBookCard();

HistoryOfIssue historyOfIssue = BuildHistoryOfIssue();

return new Book(bookCard, historyOfIssue);

}

}

}

BookCard.cs:

using System;

namespace library\_laba4

{

public class BookCard : IEquatable<BookCard>

{

public BookCard(int department, string name, string author)

{

Department = department;

Name = name;

Author = author;

}

public string Name { private set; get; }

public string Author { private set; get; }

public int Department { private set; get; }

public int BookсaseNumber { set; get; }

public int ShelfNumber { set; get; }

public int СellNumber { set; get; }

public int InstanceNumber { set; get; }

public bool Equals(BookCard other)

{

if (other != null)

{

return Department == other.Department

&& Name == other.Name

&& Author == other.Author;

}

return false;

}

public (int, int, int) GetPosition() => (BookсaseNumber, ShelfNumber, СellNumber);

public void SetPosition(int bookсaseNumber, int shelfNumber, int cellNumber)

{

BookсaseNumber = bookсaseNumber;

ShelfNumber = shelfNumber;

СellNumber = cellNumber;

}

public string GetFullInformation()

=> $"{ToString()}\nПоложение: № отдела: {Department}," +

$"№ шкафа: {BookсaseNumber}, № полки: {ShelfNumber}, № ячейки: {СellNumber}\n" +

$"Количество экземпляров: {InstanceNumber}\n";

public override string ToString()

{

return $"Название: {Name}, Автор: {Author}";

}

}

}

BookParser.cs:

using System;

using System.Text.RegularExpressions;

namespace library\_laba4

{

public class BookParser

{

// паттерны

private static readonly Regex nameAndAthorPatern = new Regex(@"Название: ([A-Z А-Я Ё a-z а-я ё 0-9 \-]+), Автор: ([A-Z А-Я Ё a-z а-я ё 0-9 \-]+)");

private static readonly Regex positionPatern = new Regex(@"Положение: № отдела: ([0-9]+),№ шкафа: ([0-9]+), № полки: ([0-9]+), № ячейки: ([0-9]+)");

private static readonly Regex numberPatern = new Regex(@"Количество экземпляров: ([0-9]+)");

private static readonly Regex issuePatern = new Regex(@"Дата выдачи: (\d{2}.\d{2}.\d{4}) Количество дней: (\d+) Номер Читательского билета: ([0-9 a-z]{8}-[0-9 a-z]{4}-[0-9 a-z]{4}-[0-9 a-z]{4}-[0-9 a-z]{12})");

private static readonly Regex returnPatern = new Regex(@"Дата возврата: (\d{2}.\d{2}.\d{4}) Номер Читательского билета: ([0-9 a-z]{8}-[0-9 a-z]{4}-[0-9 a-z]{4}-[0-9 a-z]{4}-[0-9 a-z]{12})");

// поля

private readonly IBookBuilder \_builder;

public BookParser(string src, IBookBuilder builder)

{

\_builder = builder;

ParseBookCard(nameAndAthorPatern.Match(src),

positionPatern.Match(src),

numberPatern.Match(src));

ParseHistory(issuePatern.Matches(src),

returnPatern.Matches(src));

}

public Book Parse()

{

return \_builder.Build();

}

private void ParseBookCard(Match nameAndAthor, Match position, Match number)

{

\_builder.SetBookName(nameAndAthor.Groups[1].Value)

.SetAuthor(nameAndAthor.Groups[2].Value)

.SetBookDepartment(int.Parse(position.Groups[1].Value))

.SetBookсaseNumber(int.Parse(position.Groups[2].Value))

.SetShelfNumber(int.Parse(position.Groups[3].Value))

.SetCellNumber(int.Parse(position.Groups[4].Value))

.SetInstanceNumber(int.Parse(number.Groups[1].Value));

}

private void ParseHistory(MatchCollection issues, MatchCollection returns)

{

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

{

\_builder.AddIssueRecord(DateTime.Parse(issues[i].Groups[1].Value),

int.Parse(issues[i].Groups[2].Value),

Guid.Parse(issues[i].Groups[3].Value)

);

}

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

{

\_builder.AddReturnRecord(DateTime.Parse(returns[i].Groups[1].Value),

Guid.Parse(returns[i].Groups[2].Value)

);

}

}

}

}

HistoryOfIssue.cs:

using System;

using System.Collections.Generic;

using System.Text;

namespace library\_laba4

{

public class HistoryOfIssue

{

private IList<(DateTime, int, Guid)> \_issueRecord;

private IList<(DateTime, Guid)> \_returnRecord;

public HistoryOfIssue()

{

\_issueRecord = new List<(DateTime, int, Guid)>();

\_returnRecord = new List<(DateTime, Guid)>(); ;

}

public void IssueRecord(DateTime dateOfIssue, int days, Guid userID)

{

\_issueRecord.Add((dateOfIssue, days, userID));

}

public void IssueRecord((DateTime, int, Guid) record)

{

\_issueRecord.Add(record);

}

public void ReturnRecord(DateTime dateOfIssue, Guid userID)

{

\_returnRecord.Add((dateOfIssue, userID));

}

public void ReturnRecord((DateTime, Guid) record)

{

\_returnRecord.Add(record);

}

public string GetHistory()

{

StringBuilder builder = new StringBuilder();

int i = 0;

for (; i < \_returnRecord.Count; i++)

{

AppendIssue(builder, i);

AppendRecord(builder, i);

}

if (i == \_issueRecord.Count - 1)

{

AppendIssue(builder, i);

}

return builder.ToString();

}

private void AppendIssue(StringBuilder builder, int i)

{

builder.Append("Дата выдачи: ")

.Append(\_issueRecord[i].Item1.ToShortDateString())

.Append(" Количество дней: ")

.Append(\_issueRecord[i].Item2)

.Append(" Номер Читательского билета: ")

.Append(\_issueRecord[i].Item3)

.AppendLine();

}

private void AppendRecord(StringBuilder builder, int i)

{

builder.Append("Дата возврата: ")

.Append(\_returnRecord[i].Item1.ToShortDateString())

.Append(" Номер Читательского билета: ")

.Append(\_returnRecord[i].Item2)

.AppendLine();

}

public override string ToString()

{

return GetHistory();

}

}

}

IBookBuilder.cs:

using System;

namespace library\_laba4

{

public interface IBookBuilder

{

BookBuilder AddIssueRecord(DateTime dateOfIssue, int days, Guid userID);

BookBuilder AddReturnRecord(DateTime dateOfIssue, Guid userID);

BookBuilder SetAuthor(string author);

BookBuilder SetBookDepartment(int department);

BookBuilder SetBookName(string bookName);

BookBuilder SetBookсaseNumber(int bookсaseNumber);

BookBuilder SetCellNumber(int cellNumber);

BookBuilder SetInstanceNumber(int instanceNumber);

BookBuilder SetShelfNumber(int shelfNumber);

Book Build();

}

}

Bookсase.cs:

using System.Collections.Generic;

namespace library\_laba4

{

public class Bookсase

{

private Book[][] \_books;

public Bookсase(int hieght, int width)

{

\_books = new Book[hieght][];

for (int i = 0; i < \_books.Length; i++)

{

\_books[i] = new Book[width];

}

}

public Book GetBook(int i, int j)

{

var temp = \_books[i][j];

\_books[i][j] = null;

return temp;

}

public Book ShowBook(int i, int j)

{

return \_books[i][j];

}

public IList<Book> GetBooks()

{

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

for (int i = 0; i < \_books.Length; i++)

{

for (int j = 0; j < \_books[i].Length; j++)

{

Book bookCell = GetBook(i, j);

if (bookCell != null)

{

books.Add(bookCell);

}

}

}

return books;

}

public IList<Book> ShowBooks()

{

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

for (int i = 0; i < \_books.Length; i++)

{

for (int j = 0; j < \_books[i].Length; j++)

{

Book bookCell = \_books[i][j];

if (bookCell != null)

{

books.Add(bookCell);

}

}

}

return books;

}

public void SetBook(Book book, int shelf, int cell)

{

\_books[shelf][cell] = book;

}

public (int, int) GetSize()

{

return (\_books.Length, \_books[0].Length);

}

}

}

Department.cs:

using System.Text;

namespace library\_laba4

{

public class Department : IDepartment

{

public string Name { get; private set; }

private readonly Bookсase[] \_bookсases;

public Department(string name, int bookсaseNum, int bookсaseHieght, int bookсaseWidth)

{

Name = name;

\_bookсases = new Bookсase[bookсaseNum];

for (int i = 0; i < \_bookсases.Length; i++)

{

\_bookсases[i] = new Bookсase(bookсaseHieght, bookсaseWidth);

}

}

public void SetBook(Book book)

{

int deport, shelf, cell;

(deport, shelf, cell) = book.GetPosition();

\_bookсases[deport].SetBook(book, shelf, cell);

}

public Book GetBook(int deport, int shelf, int cell)

{

return \_bookсases[deport].GetBook(shelf, cell);

}

public Book ShowBook(int deport, int shelf, int cell)

{

return \_bookсases[deport].ShowBook(shelf, cell);

}

public string GetFullInformation()

{

StringBuilder builder = new StringBuilder();

builder.AppendLine("Основная информация об отделе");

builder.AppendLine($"Название: {Name}");

var a = \_bookсases[0].GetSize();

builder.AppendLine($"Размер шкафов: {a.Item1} {a.Item2}");

int i = 1;

foreach (var bookcase in \_bookсases)

{

foreach (var book in bookcase.ShowBooks())

{

builder.AppendLine($"===== Книга {i++} =====");

builder.Append(book.GetFullInformation());

builder.AppendLine(" ");

}

}

return builder.ToString();

}

public void Dispose()

{

}

}

}

DepartmentProxy.cs:

using library\_laba4.Logers;

using System;

namespace library\_laba4

{

public class DepartmentProxy : IDepartment, IDisposable

{

private readonly Department \_department;

private readonly ILoger \_loger;

public DepartmentProxy(Department department, ILoger loger)

{

\_loger = loger;

\_department = department;

\_loger.Log(\_department.GetFullInformation());

}

public string Name => \_department.Name;

public void Dispose()

{

\_loger.Log(\_department.GetFullInformation());

}

public Book GetBook(int deport, int shelf, int cell)

{

var book = \_department.GetBook(deport, shelf, cell);

\_loger.Log($"Выдана книга {book?.ToString() ?? "-"}");

return book;

}

public Book ShowBook(int deport, int shelf, int cell)

{

return \_department.ShowBook(deport, shelf, cell);

}

public string GetFullInformation()

{

\_loger.Log(\_department.GetFullInformation());

return \_department.GetFullInformation();

}

public void SetBook(Book book)

{

\_loger.Log($"Получена книга {book}");

\_department.SetBook(book);

}

}

}

DepartmentPurser.cs:

using System;

using System.Collections.Generic;

using System.IO;

using System.Text.RegularExpressions;

namespace library\_laba4

{

public class DepartmentPurser

{

private static readonly Regex namePatern = new Regex(@"Название: ([A-Z А-Я Ё a-z а-я ё 0-9 \-]+)");

private static readonly Regex sizeBookcasePatern = new Regex(@"Размер шкафов: ([0-9]+) ([0-9]+)");

private MetaData \_metaData;

private IDictionary<string, int> \_booksNames = new Dictionary<string, int>();

private IDictionary<int, IList<((int, int, int), bool)>> \_booksPositions = new Dictionary<int, IList<((int, int, int), bool)>>();

private Department \_department;

public DepartmentPurser(string filePath)

{

StreamReader fs = new StreamReader(filePath);

fs.ReadLine();

string name = namePatern.Match(fs.ReadLine() ?? "Название: -").Groups[1].Value;

var sizeBookcase = sizeBookcasePatern.Match(fs.ReadLine() ?? "Размер шкафов: 10 10").Groups;

int hieght = int.Parse(sizeBookcase[1].Value);

int width = int.Parse(sizeBookcase[2].Value);

var booksSource = fs.ReadToEnd().Split(' ');

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

int maxBookcaseNum = 1;

foreach (var bookSource in booksSource)

{

Book book = new BookParser(bookSource, new BookBuilder()).Parse();

books.Add(book);

maxBookcaseNum = Math.Max(maxBookcaseNum, book.GetPosition().Item1 + 1);

AddBookToConteiners(book);

}

fs.Close();

\_department = new Department(name, maxBookcaseNum, hieght, width);

\_metaData = new MetaData(\_booksNames, \_booksPositions);

foreach (var book in books)

{

\_department.SetBook(book);

}

}

private void AddBookToConteiners(Book book)

{

if (!\_booksNames.ContainsKey(book.Name))

{

\_booksNames.Add(book.Name, \_booksNames.Count);

IList<((int, int, int), bool)> node = new List<((int, int, int), bool)>();

node.Add((book.GetPosition(), true));

\_booksPositions.Add(\_booksNames.Count - 1, node);

}

else

{

\_booksPositions[\_booksNames[book.Name]].Add((book.GetPosition(), true));

}

}

public Department Parse()

{

return \_department;

}

public MetaData GetMetaData()

{

return \_metaData;

}

}

}

IDepartment.cs:

using System;

namespace library\_laba4

{

public interface IDepartment : IDisposable

{

string Name { get; }

Book GetBook(int deport, int shelf, int cell);

Book ShowBook(int deport, int shelf, int cell);

string GetFullInformation();

void SetBook(Book book);

}

}

MetaData.cs:

using System.Collections.Generic;

namespace library\_laba4

{

public class MetaData

{

public IDictionary<string, int> BooksNames { get; private set; }

public IDictionary<int, IList<((int, int, int), bool)>> BooksPositions { get; private set; }

public MetaData(IDictionary<string, int> booksNames

, IDictionary<int, IList<((int, int, int), bool)>> booksPositions)

{

BooksNames = booksNames;

BooksPositions = booksPositions;

}

public IList<((int, int, int), bool)> GetPositionsByName(string name)

{

return BooksPositions[BooksNames[name]];

}

}

}

IObservable.cs:

namespace library\_laba4

{

public interface IObservable

{

void AddObserver(IObserver o, int days);

void RemoveObserver(IObserver o);

void NotifyObservers();

void NotifyAllObservers();

}

}

IObserver.cs:

namespace library\_laba4

{

public interface IObserver

{

void Update();

}

}

Librarian.cs:

using System;

using System.Collections.Generic;

using System.Linq;

namespace library\_laba4

{

public class Librarian : IObservable

{

private MetaData \_artisticData;

private MetaData \_technicalData;

private IDepartment \_artistic;

private IDepartment \_technical;

List<IObserver> \_notAnswered;

List<List<IObserver>> \_debtors;

IDictionary<Guid, BookCard> \_takenBooks;

public Librarian(MetaData artisticData, MetaData technicalData,

IDepartment artistic, IDepartment technical)

{

\_artistic = artistic;

\_technical = technical;

\_artisticData = artisticData;

\_technicalData = technicalData;

\_notAnswered = new List<IObserver>();

\_debtors = new List<List<IObserver>>();

\_takenBooks = new Dictionary<Guid, BookCard>();

}

public IList<ICollection<string>> GetBookesNames()

{

return new List<ICollection<string>>()

{

\_artisticData.BooksNames.Keys,

\_technicalData.BooksNames.Keys

};

}

public void GetBook(int department, int bookNumber, BookReader bookReader, DateTime date, int days)

{

Book book = GetBook(department, bookNumber);

if (book != null)

{

bookReader.SetBook(book, days);

book?.IssueRecord(date, days, bookReader.Guid);

AddObserver(bookReader, days);

\_takenBooks.Add(bookReader.Guid, book.BookCard);

}

}

private Book GetBook(int department, int bookNumber)

{

Book result = null;

if (department == 0)

{

result = FindBook(\_artistic, \_artisticData, bookNumber);

}

else if (department == 1)

{

result = FindBook(\_technical, \_technicalData, bookNumber);

}

return result;

}

private Book FindBook(IDepartment department, MetaData metaData, int bookNumber)

{

(int, int, int) postion = GetPosition(metaData, bookNumber);

try

{

return department.GetBook(postion.Item1, postion.Item2, postion.Item3);

}

catch (Exception)

{

return null;

}

}

public (int, int, int) GetPosition(MetaData metaData, int bookNumber)

{

var postions = metaData.BooksPositions[bookNumber];

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

{

if (postions[i].Item2)

{

postions[i] = (postions[i].Item1, false);

return postions[i].Item1;

}

}

return (-1, -1, -1);

}

public Book SetBook(BookReader bookReader, DateTime date)

{

Book book = bookReader.GetBook();

if (book != null)

{

int department = book.Department;

if (department == 0)

{

SetBook(book, \_artistic, \_artisticData);

}

else if (department == 1)

{

SetBook(book, \_technical, \_technicalData);

}

book.ReturnRecord(date, bookReader.Guid);

RemoveObserver(bookReader);

\_takenBooks.Remove(bookReader.Guid);

}

return book;

}

public BookCard LostBook(BookReader bookReader)

{

BookCard bookCard = \_takenBooks[bookReader.Guid];

bookReader.AddDebt(1000);

RemoveObserver(bookReader);

\_takenBooks.Remove(bookReader.Guid);

var temp = \_takenBooks.Where(cardPare => cardPare.Value.Equals(bookCard));

foreach (var card in temp)

{

--card.Value.InstanceNumber;

}

if (bookCard.Department == 0)

{

var positions = \_artisticData.GetPositionsByName(bookCard.Name);

positions.Remove((bookCard.GetPosition(), false));

foreach (var pos in positions)

{

if (pos.Item2)

{

\_artistic.ShowBook(pos.Item1.Item1, pos.Item1.Item2, pos.Item1.Item3)

.BookCard.InstanceNumber--;

}

}

}

else if (bookCard.Department == 1)

{

}

return bookCard;

}

private void SetBook(Book book, IDepartment department, MetaData metaDate)

{

var bookPosition = book.GetPosition();

department.SetBook(book);

int bookID = metaDate.BooksNames[book.Name];

for (int i = 0; i < metaDate.BooksPositions[bookID].Count; i++)

{

if (metaDate.BooksPositions[bookID][i].Item1 == bookPosition)

{

metaDate.BooksPositions[bookID][i] = (bookPosition, true);

}

}

}

public void AddObserver(IObserver o, int days)

{

int size = \_debtors.Count;

if (size - 1 <= days)

{

\_debtors.Capacity = days + 1;

for (int i = size; i < \_debtors.Capacity; i++)

{

\_debtors.Add(new List<IObserver>());

}

}

\_debtors[days].Add(o);

}

public void RemoveObserver(IObserver o)

{

\_notAnswered.Remove(o);

foreach (var dayline in \_debtors)

{

dayline.Remove(o);

}

}

public void NotifyObservers()

{

if (\_debtors.Count > 0)

{

\_notAnswered.AddRange(\_debtors[0]);

\_debtors.RemoveAt(0);

}

foreach (BookReader v in \_notAnswered)

{

v.Update();

}

}

public void NotifyAllObservers()

{

if (\_debtors.Count > 0)

{

int size = \_debtors.Count;

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

{

\_notAnswered.AddRange(\_debtors[0]);

\_debtors.RemoveAt(0);

}

}

foreach (BookReader v in \_notAnswered)

{

v.Update();

}

}

}

}

Library.cs:

using library\_laba4.Logers;

using System;

using System.Collections.Generic;

namespace library\_laba4

{

public class Library : IDisposable

{

public const int ISSUE\_PERIOD = 14;

IDepartment \_artistic;

IDepartment \_technical;

Librarian \_librarian;

ILoger \_loger;

IList<BookReader> \_bookreaders = new List<BookReader>();

Queue<BookReader> \_queue = new Queue<BookReader>();

public Library(IDepartment artistic, IDepartment technical, Librarian librarian, ILoger loger)

{

\_artistic = artistic;

\_technical = technical;

\_librarian = librarian;

\_loger = loger;

}

public void AddBookReader(BookReader bookReader)

{

bookReader.Loger = \_loger;

bookReader.Library = this;

\_bookreaders.Add(bookReader);

}

public void AddBookReader(IList<BookReader> bookReaders)

{

foreach (var bookReader in bookReaders)

{

AddBookReader(bookReader);

}

}

public void Dispose()

{

\_artistic.Dispose();

\_technical.Dispose();

}

public void LetIn(BookReader bookReader)

{

if (!\_bookreaders.Contains(bookReader))

{

AddBookReader(bookReader);

}

\_queue.Enqueue(bookReader);

}

public void SimulateDay()

{

\_librarian.NotifyObservers();

while (\_queue.Count != 0)

{

var bookReader = \_queue.Dequeue();

switch (bookReader.MyIntention)

{

case BookReader.Intention.returnBook:

ReturnBook(bookReader);

break;

case BookReader.Intention.requestBook:

GiveBook(bookReader);

break;

}

}

}

public void CloseLibrary()

{

\_librarian.NotifyAllObservers();

while (\_queue.Count != 0)

{

var bookReader = \_queue.Dequeue();

switch (bookReader.MyIntention)

{

case BookReader.Intention.returnBook:

ReturnBook(bookReader);

break;

}

}

}

private void GiveBook(BookReader bookReader)

{

DateTime date = Today.Data;

int department, bookNumber;

(department, bookNumber) = bookReader.RequestBook(\_librarian.GetBookesNames());

\_librarian.GetBook(department, bookNumber, bookReader, date, ISSUE\_PERIOD);

\_loger.Log(bookReader.ShowBook()?.GetFullInformation() ?? "Книги нет в наличии");

}

private void ReturnBook(BookReader bookReader)

{

DateTime date = Today.Data;

var book = \_librarian.SetBook(bookReader, date);

if (book == null)

{

BookCard bookCard = \_librarian.LostBook(bookReader);

\_loger.Log(bookCard.ToString());

\_loger.Log($"Актуальное число книг {--bookCard.InstanceNumber}");

}

else

{

\_loger.Log(book.GetFullInformation());

}

\_loger.Log();

}

}

}

ConsoleLoger.cs:

using System;

using System.Windows.Forms;

namespace library\_laba4.Logers

{

public class ConsoleLoger : ILoger

{

private System.Windows.Forms.TextBox textBox1;

public ConsoleLoger(TextBox textBox1)

{

this.textBox1 = textBox1;

}

public void Log(string message)

{

try

{

if (textBox1?.InvokeRequired ?? false)

textBox1?.Invoke(new Action(() =>

{

textBox1.AppendText(message.Replace("\n", Environment.NewLine));

textBox1.AppendText(Environment.NewLine);

}));

else

textBox1?.AppendText(message.Replace("\n", Environment.NewLine));

}

catch (Exception e)

{

textBox1 = null;

}

}

public void Log()

{

try {

if (textBox1?.InvokeRequired ?? false)

textBox1?.Invoke(new Action(() =>

{

textBox1.AppendText(Environment.NewLine);

}));

else

textBox1?.AppendText(Environment.NewLine);

}

catch (Exception e)

{

textBox1 = null;

}

}

public void Close()

{

textBox1 = null;

}

}

}

FileLoger.cs:

using System;

using System.IO;

namespace library\_laba4.Logers

{

public class FileLoger : ILoger, IDisposable

{

private StreamWriter \_fileStream = new StreamWriter("log.txt", append: true);

public void Log(string message)

{

\_fileStream.WriteLine(message);

}

public void Log()

{

\_fileStream.WriteLine();

}

public void Close()

{

\_fileStream.Close();

}

public void Dispose()

{

\_fileStream.Dispose();

}

}

}

ILoger.cs:

namespace library\_laba4.Logers

{

public interface ILoger

{

void Log(string message);

void Log();

void Close();

}

}

AccurateBookManagementStrategy.cs:

namespace Laba4Pattern.Readeres

{

internal class AccurateBookManagementStrategy : IBookManagementStrategy

{

public bool IntentionLostBook()

{

return false;

}

public bool IntentionRequestBook()

{

return true;

}

public bool IntentionReturnBook(int days)

{

return days < 2;

}

}

}

BookReader.cs:

using Laba4Pattern.Readeres;

using library\_laba4.Logers;

using System;

using System.Collections.Generic;

using System.Linq;

namespace library\_laba4

{

public class BookReader : IObserver

{

public enum Intention

{

returnBook,

requestBook,

none

}

private static readonly Random RANDOM = new Random();

public BookReader(IBookManagementStrategy bookManagementStrategy)

{

\_bookManagementStrategy = bookManagementStrategy;

}

private Book \_book;

private int \_days = 0;

public bool ContainBook => \_book != null;

public Guid Guid { get; private set; } = Guid.NewGuid();

public ILoger Loger { get; set; }

public Library Library { get; set; }

public Intention MyIntention { get; set; }

private IBookManagementStrategy \_bookManagementStrategy;

private int \_debt;

public int GetDebt()

{

return \_debt;

}

public void AddDebt(int value)

{

\_debt += value;

Loger?.Log($"Читатель {Guid} получил штраф {value} общая задолжность составляет {\_debt}");

}

public Book GetBook()

{

var t = \_book;

\_book = null;

MyIntention = Intention.none;

\_days = 0;

if (\_bookManagementStrategy.IntentionLostBook())

{

Loger?.Log($"Читатель {Guid} потерял книгу");

}

return t;

}

public void SetBook(Book value, int day)

{

\_book = value;

\_days = day;

MyIntention = Intention.none;

}

public Book ShowBook()

{

return \_book;

}

public void SimulateDay()

{

if (ContainBook)

{

if (\_bookManagementStrategy.IntentionReturnBook(\_days))

{

MyIntention = Intention.returnBook;

GoToLibrary(Library);

}

\_days--;

}

else

{

if (\_bookManagementStrategy.IntentionRequestBook())

{

MyIntention = Intention.requestBook;

GoToLibrary(Library);

}

}

}

public (int, int) RequestBook(IList<ICollection<string>> booksNames)

{

int department = RANDOM.Next(booksNames.Count);

int bookNumber = RANDOM.Next(booksNames[department].Count);

Loger?.Log($"Запрос книги {booksNames[department].ElementAt(bookNumber)}"

+ " Читателем " + Guid);

return (department, bookNumber);

}

public void GoToLibrary(Library library)

{

library?.LetIn(this);

}

public void Update()

{

Loger?.Log($"Читатель {Guid} оповещён о задожности");

MyIntention = Intention.returnBook;

GoToLibrary(Library);

}

}

}

IBookManagementStrategy.cs:

namespace Laba4Pattern.Readeres

{

public interface IBookManagementStrategy

{

bool IntentionReturnBook(int days);

bool IntentionRequestBook();

bool IntentionLostBook();

}

}

RandomBookManagementStrategy.cs:

using System;

namespace Laba4Pattern.Readeres

{

public class RandomBookManagementStrategy : IBookManagementStrategy

{

private static readonly Random RANDOM = new Random();

public bool IntentionLostBook()

{

double probability = 0.99;

double r = RANDOM.NextDouble();

return probability <= r;

}

public bool IntentionRequestBook()

{

double probability = 0.6;

double r = RANDOM.NextDouble();

return probability <= r;

}

public bool IntentionReturnBook(int days)

{

if (days < 0)

{

double probability = 0.9;

double r = RANDOM.NextDouble();

return probability <= r;

}

else if (days < 5)

{

double probability = (1.0 / (0.8 \* days + 1.1));

double r = RANDOM.NextDouble();

return probability <= r;

}

return false;

}

}

}

SlovenBookManagementStrategy.cs:

using System;

namespace Laba4Pattern.Readeres

{

public class SlovenBookManagementStrategy : IBookManagementStrategy

{

private static readonly Random RANDOM = new Random();

public bool IntentionLostBook()

{

return RANDOM.NextDouble() > 0.5;

}

public bool IntentionRequestBook()

{

return RANDOM.NextDouble() > 0.5;

}

public bool IntentionReturnBook(int days)

{

return false;

}

}

}

artistic.txt:

Основная информация об отделе

Название: Художественная

Размер шкафов: 10 20

===== Книга 1 =====

Основная информация

Название: Великий Гусляр, Автор: Кир Булычёв,

Положение: № отдела: 0,№ шкафа: 0, № полки: 0, № ячейки: 0

Количество экземпляров: 2

История

Дата выдачи: 04.05.2000 Количество дней: 10 Номер Читательского билета: 2e016384-9c28-4da3-bed0-97dadfc94891

Дата возврата: 14.05.2000 Номер Читательского билета: 2e016384-9c28-4da3-bed0-97dadfc94891

===== Книга 2 =====

Основная информация

Название: Великий Гусляр, Автор: Кир Булычёв,

Положение: № отдела: 0,№ шкафа: 0, № полки: 0, № ячейки: 1

Количество экземпляров: 2

История

Дата выдачи: 04.05.2000 Количество дней: 5 Номер Читательского билета: 2e016384-9c28-4da3-bed0-97dadfc94891

Дата возврата: 09.05.2000 Номер Читательского билета: 2e016384-9c28-4da3-bed0-97dadfc94891

===== Книга 3 =====

Основная информация

Название: Ночной дозор, Автор: Сергей Лукьяненко,

Положение: № отдела: 0,№ шкафа: 3, № полки: 0, № ячейки: 1

Количество экземпляров: 2

История

===== Книга 4 =====

Основная информация

Название: Ночной дозор, Автор: Сергей Лукьяненко,

Положение: № отдела: 0,№ шкафа: 3, № полки: 0, № ячейки: 2

Количество экземпляров: 2

История

===== Книга 5 =====

Основная информация

Название: Ночной дозор, Автор: Сергей Лукьяненко,

Положение: № отдела: 0,№ шкафа: 3, № полки: 1, № ячейки: 1

Количество экземпляров: 2

История

technical.txt:

Основная информация об отделе

Название: Технический

Размер шкафов: 10 20

===== Книга 1 =====

Основная информация

Название: Паттерны проектирования, Автор: Банда четырёх,

Положение: № отдела: 1,№ шкафа: 0, № полки: 0, № ячейки: 0

Количество экземпляров: 3

История

Дата выдачи: 04.05.2000 Количество дней: 5 Номер Читательского билета: 2e016384-9c28-4da3-bed0-97dadfc94891

Дата возврата: 09.05.2000 Номер Читательского билета: 2e016384-9c28-4da3-bed0-97dadfc94891

===== Книга 2 =====

Основная информация

Название: Паттерны проектирования, Автор: Банда четырёх,

Положение: № отдела: 1,№ шкафа: 0, № полки: 0, № ячейки: 1

Количество экземпляров: 3

История

Дата выдачи: 04.05.2000 Количество дней: 5 Номер Читательского билета: 2e016384-9c28-4da3-bed0-97dadfc94891

Дата возврата: 09.05.2000 Номер Читательского билета: 2e016384-9c28-4da3-bed0-97dadfc94891

===== Книга 3 =====

Основная информация

Название: Паттерны проектирования, Автор: Банда четырёх,

Положение: № отдела: 1,№ шкафа: 0, № полки: 0, № ячейки: 2

Количество экземпляров: 3

История

Дата выдачи: 04.05.2000 Количество дней: 5 Номер Читательского билета: 2e016384-9c28-4da3-bed0-97dadfc94891

Дата возврата: 09.05.2000 Номер Читательского билета: 2e016384-9c28-4da3-bed0-97dadfc94891

===== Книга 4 =====

Основная информация

Название: Совершенный код, Автор: Стив Макконнелл,

Положение: № отдела: 1,№ шкафа: 0, № полки: 0, № ячейки: 3

Количество экземпляров: 2

История

Дата выдачи: 04.09.2002 Количество дней: 5 Номер Читательского билета: 2e016384-9c28-4da3-bed0-97dadfc94891

Дата возврата: 09.09.2002 Номер Читательского билета: 2e016384-9c28-4da3-bed0-97dadfc94891

===== Книга 5 =====

Основная информация

Название: Совершенный код, Автор: Стив Макконнелл,

Положение: № отдела: 1,№ шкафа: 0, № полки: 1, № ячейки: 0

Количество экземпляров: 2

История

Дата выдачи: 04.09.2002 Количество дней: 3 Номер Читательского билета: 2e016384-9c28-4da3-bed0-97dadfc94891

Дата возврата: 07.09.2002 Номер Читательского билета: 2e016384-9c28-4da3-bed0-97dadfc94891

===== Книга 6 =====

Основная информация

Название: Java 8 Полное руководство, Автор: Герберт Шилдт,

Положение: № отдела: 1,№ шкафа: 1, № полки: 5, № ячейки: 0

Количество экземпляров: 4

История

Дата выдачи: 04.09.2002 Количество дней: 5 Номер Читательского билета: 2e016384-9c28-4da3-bed0-97dadfc94891

Дата возврата: 09.09.2002 Номер Читательского билета: 2e016384-9c28-4da3-bed0-97dadfc94891

===== Книга 7 =====

Основная информация

Название: Java 8 Полное руководство, Автор: Герберт Шилдт,

Положение: № отдела: 1,№ шкафа: 1, № полки: 5, № ячейки: 1

Количество экземпляров: 4

История

Дата выдачи: 04.09.2002 Количество дней: 3 Номер Читательского билета: 2e016384-9c28-4da3-bed0-97dadfc94891

Дата возврата: 07.09.2002 Номер Читательского билета: 2e016384-9c28-4da3-bed0-97dadfc94891

===== Книга 8 =====

Основная информация

Название: Java 8 Полное руководство, Автор: Герберт Шилдт,

Положение: № отдела: 1,№ шкафа: 1, № полки: 5, № ячейки: 2

Количество экземпляров: 4

История

Дата выдачи: 04.09.2002 Количество дней: 5 Номер Читательского билета: 2e016384-9c28-4da3-bed0-97dadfc94891

Дата возврата: 09.09.2002 Номер Читательского билета: 2e016384-9c28-4da3-bed0-97dadfc94891

===== Книга 9 =====

Основная информация

Название: Java 8 Полное руководство, Автор: Герберт Шилдт,

Положение: № отдела: 1,№ шкафа: 1, № полки: 5, № ячейки: 3

Количество экземпляров: 4

История

Дата выдачи: 04.09.2002 Количество дней: 3 Номер Читательского билета: 2e016384-9c28-4da3-bed0-97dadfc94891

Дата возврата: 07.09.2002 Номер Читательского билета: 2e016384-9c28-4da3-bed0-97dadfc94891