МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ

«БРЕСТСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ» ФАКУЛЬТЕТ ЭЛЕКТРОННО-ИНФОРМАЦИОННЫХ СИСТЕМ

Кафедра интеллектуальных информационных технологий

Отчет по лабораторной работе №6

Специальность ПО-5(о)

Выполнил

Д. С. Бриштен,

студент группы ПО-5

Проверил

А. А. Крощенко

ст. преп. каф. ИИТ

Брест 2021

Цель работы: приобрести навыки применения паттернов проектирования при решении практических задач с использованием языка Java.

Вариант 6

Задание 1. Музыкальный магазин. Должно обеспечиваться одновременное обслуживание нескольких покупателей. Магазин должен предоставлять широкий выбор товаров различных музыкальных направлений.

Выполнение:

Код программы

// ABSTRACT FACTORY

/\*

    Позволяет создавать семейства независимых продуктов.

    Общий интерфейс для отдельных продуктов.

\*/

public class MusikStore {

    public static void main(String[] args) {

        Client Ivan = new Client(new BrestFactory());

        Ivan.Hit();

        Ivan.Run();

        Client Vasya = new Client(new MinskFactory());

        Vasya.Hit();

        Vasya.Run();

    }

}

abstract class Percussion

{

    public abstract void Hit();

}

abstract class Folk

{

    public abstract void Move();

}

class Dish extends Percussion

{

    @Override

    public void Hit()

    {

        System.out.println("Dish");

    }

}

class Drum extends Percussion

{

    @Override

    public void Hit()

    {

        System.out.println("Drum");

    }

}

class Garmon extends Folk

{

    @Override

    public void Move()

    {

        System.out.println("Garmon");

    }

}

//

class Bayan extends Folk

{

    @Override

    public void Move()

    {

        System.out.println("Bayan");

    }

}

abstract class ClientFactory

{

    public abstract Folk CreateFolkTool();

    public abstract Percussion CreatePecussionTool();

}

class BrestFactory extends ClientFactory

{

    @Override

    public Folk CreateFolkTool()

    {

        return new Garmon();

    }

    @Override

    public Percussion CreatePecussionTool()

    {

        return new Drum();

    }

}

class MinskFactory extends ClientFactory

{

    @Override

    public Folk CreateFolkTool()

    {

        return new Bayan();

    }

    @Override

    public Percussion CreatePecussionTool()

    {

        return new Dish();

    }

}

// клиент

class Client

{

    static int num = 0;

    private Percussion perTool;

    private Folk filkTool;

    public Client(ClientFactory factory)

    {

        num += 1;

        System.out.println("\n=========== Client #" + num + " ============");

        perTool = factory.CreatePecussionTool();

        filkTool = factory.CreateFolkTool();

    }

    public void Run()

    {

        filkTool.Move();

    }

    public void Hit()

    {

        perTool.Hit();

    }

}

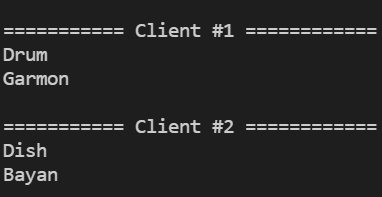
Спецификация вывода:

Используя переопределённый метод Move()

<num> - номер клиента

<sysout> - музыкальный инструмент и текст

Пример:



Задание 2. Учетная запись покупателя книжного интернет-магазина. Предусмотреть различные уровни учетки в зависимости от активности покупателя. Дополнительные уровни добавляют функциональные возможности и открывают доступ к уникальным предложениям.

Выполнение:

Код программы

// STRATEGY

/\*

    В зависимости от ситуации мы можем легко заменить один используемый алгоритм другим.

    Обеспечивается выбор из нескольких вариантов алгоритмов,

    которые можно легко менять в зависимости от условий.

\*/

public class Accounting {

    public static void main(String[] args) {

        Buyer Volodya = new Buyer("Volodya", "High", new HighLevel());

        Volodya.info();

        Volodya.Opportunities();

        System.out.println("Dropped a book");

        Volodya.setLevel(new AverageLevel());

        Volodya.Opportunities();

        System.out.println("Tore up a book");

        Volodya.setLevel(new LowLevel());

        Volodya.Opportunities();

    }

}

interface ILevel

{

    void Opportunities();

}

class LowLevel implements ILevel

{

    public void Opportunities()

    {

        System.out.println("Reader can buy a book");

    }

}

class AverageLevel implements ILevel

{

    public void Opportunities()

    {

        System.out.println("Reader can buy a book at 30% discount");

    }

}

class HighLevel implements ILevel

{

    public void Opportunities()

    {

        System.out.println("Reader can get the secong book as a gift after buying the first one");

    }

}

class Buyer

{

    private ILevel Level;

    protected String name;

    protected String activity;

    public Buyer()

    {

        this.name = "None";

        this.activity = "None";

        this.Level = null;

    }

    public Buyer(String name, String activity, ILevel level)

    {

        this.name = name;

        this.activity = activity;

        Level = level;

    }

    public void info()

    {

        System.out.println("The buyer is " + this.name);

    }

    public void setLevel(ILevel Level) { this.Level = Level; }

    public void Opportunities()

    {

        Level.Opportunities();

    }

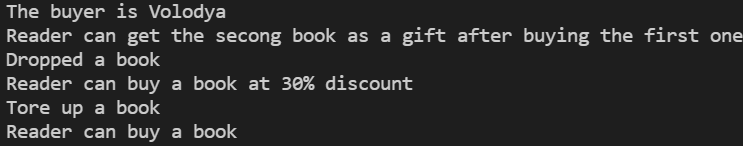
}

Спецификация вывода:

Используя метод info(): <name> - имя покупателя

Метод opportunities(): <информация о возможностях>

Пример:



Задание 3. Проект Принтер. Предусмотреть выполнение операций (печать, загрузка бумаги, извлечение зажатой бумаги, заправка картриджа), режимы – ожидание, печать документа, зажатие бумаги, отказ – при отсутствии бумаги или краски, атрибуты – модель, количество листов в лотке, % краски в картридже, вероятность зажатия.

Выполнение:

Код программы

import java.util.Scanner;

//  COMMAND

/\*

    Необходимо обеспечить выполнение очереди запросов, а также их возможную отмену.

\*/

public class Printer {

    public static void main(String[] args) {

        Scanner in = new Scanner(System.in);

        System.out.print("Enter number of sheets: ");

        int NS = in.nextInt();

        in.nextLine();

        System.out.print("Enter paint percentage: ");

        int PP = in.nextInt();

        in.nextLine();

        Pult pult = new Pult();

        Print print = new Print("Bosh", NS, PP);

        pult.SetCommand(new PrinterOnCommand(print));

        if (NS == 0 || PP == 0) { pult.PressOff(); in.close(); return; }

        print.infoPrinter();

        pult.PressOn();

        pult.SetAction(new RefillCommand(print));

        pult.Execute();

        print.waiting();

        pult.SetAction(new LoadOnCommand(print));

        pult.Execute();

        print.clampPaper();

        pult.SetAction(new PrintOnCommand(print));

        pult.Execute();

        print.printDocument();

        pult.SetAction(new ExtractionOnCommand(print));

        pult.Execute();

        print.waiting();

        pult.SetCommand(new PrinterOnCommand(print));

        pult.PressOff();

        in.close();

        return;

    }

}

interface Regimes

{

    void waiting();

    void printDocument();

    void clampPaper();

}

interface StartOperation

{

    void Start();

}

interface Command

{

    void Execute();

    void Undo();

}

// Receiver - Получатель

class Print implements Regimes

{

    private String model;

    private int numSheet, paintPercentage, pinchingProbability;

    private int max = 100, min = 0;

    Print(String model, int numSheet, int paintPercentage)

    {

        this.model = model;

        this.numSheet = numSheet;

        this.paintPercentage = paintPercentage;

        this.pinchingProbability = (int)(Math.random()\*((max-min)+1))+min;

    }

    public void On() { System.out.println("Printer On!"); }

    public void Off() { System.out.println("Printer Off!"); }

    public void Printing() { System.out.println("Printing..."); }

    public void loadPaper() { System.out.println("Paper loading..."); }

    public void extractionPaper() { System.out.println("Paper extraction...") ;}

    public void refillCartridge() { System.out.println("Cartridge refill...") ;}

    public void infoPrinter()

    {

        System.out.println("\nModel \'" + model + "\', number of sheets = " + numSheet +

        ", percentage of paint = " + paintPercentage + "% and pinching probability = " +

        pinchingProbability + "%");

    }

    public void waiting()

    {

        try {

            System.out.println("Waiting...");

            Thread.sleep(10000);

            System.out.println("Ready!");

        } catch (InterruptedException e) {

            e.printStackTrace();

        }

    }

    public void printDocument()

    {

        try {

            System.out.println("Printing...");

            Thread.sleep(15000);

            System.out.println("Ready!");

        } catch (InterruptedException e) {

            e.printStackTrace();

        }

    }

    public void clampPaper()

    {

        try {

            System.out.println("Clamping...");

            Thread.sleep(5000);

            System.out.println("Ready!");

        } catch (InterruptedException e) {

            e.printStackTrace();

        }

    }

}

// ========================================================

class PrinterOnCommand implements Command

{

    Print print;

    public PrinterOnCommand(Print printSet) { print = printSet; }

    public void Execute() { print.On(); }

    public void Undo() { print.Off(); }

}

class PrintOnCommand implements StartOperation

{

    Print print;

    public PrintOnCommand(Print printSet) { print = printSet; }

    public void Start() { print.Printing(); }

}

class LoadOnCommand implements StartOperation

{

    Print print;

    public LoadOnCommand(Print printSet) { print = printSet; }

    public void Start() { print.loadPaper(); }

}

class ExtractionOnCommand implements StartOperation

{

    Print print;

    public ExtractionOnCommand(Print printSet) { print = printSet; }

    public void Start() { print.extractionPaper(); }

}

class RefillCommand implements StartOperation

{

    Print print;

    public RefillCommand(Print printSet) { print = printSet; }

    public void Start() { print.refillCartridge(); }

}

// ==============================================================

class Pult

{

    Command command;

    StartOperation start;

    public Pult() { }

    public void SetCommand(Command com) { command = com; }

    public void SetAction(StartOperation operation) { start = operation; }

    public void PressOn() { command.Execute(); }

    public void PressOff() { command.Undo(); }

    public void Execute() { start.Start(); }

}

Спецификация ввода:

<NS> - количество листов

<PP> - процент оставшейся краски

Спецификация вывода:

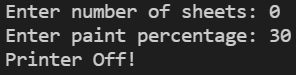
Метод info(): описание принтера

Метод execute(): осуществляет выполнение установленного действия

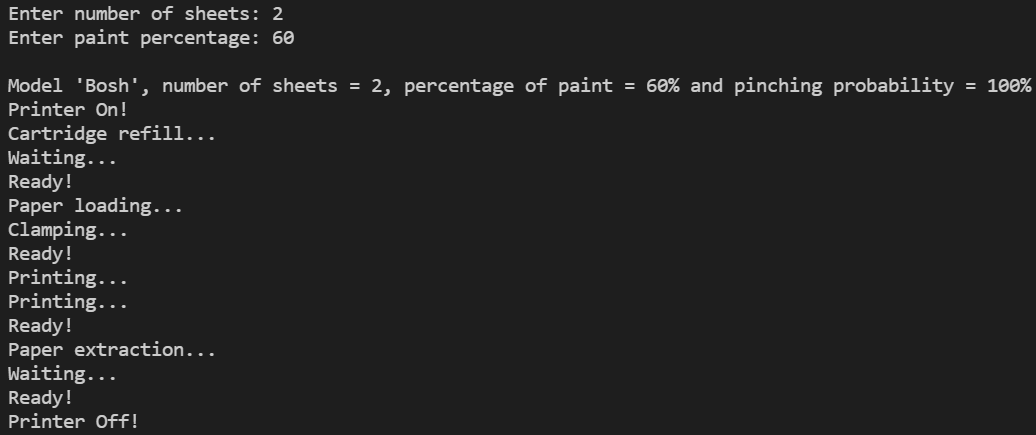
Методы интерфейса Regemes: обеспечивают задержку исполнения кода

Пример:

Случай, когда листов в принтере не оказалось – он выключается



Если всё в порядке, то действия выполнятся следующим образом



Вывод: Ознакомился с паттернами проектирования при решении практических задач в языке программирования Java.