Звіт

Автор: Пумня О., КІТ101.8б

Дата: 07.10.2019

Лабораторна робота №7

ОБ’ЄКТНО-ОРІЄНТОВАНА ДЕКОМПОЗИЦІЯ

Мета. Використання об’єктно-орієнтованого підходу для розробки об’єкта предметної (прикладної) галузі.

Вимоги:

1. Використовуючи об’єктно-орієнтований аналіз, реалізувати класи для представлення сутностей відповідно прикладної задачі - domain-об’єктів.
2. Забезпечити та продемонструвати коректне введення та відображення кирилиці.
3. Продемонструвати можливість управління масивом domain-об’єктів.

ЗАВДАННЯ ДО РОБОТИ

Використовуючи об’єктно-орієнтований аналіз, реалізувати класи для представлення сутностей відповідно прикладної задачі.

Прикладна задача. Планувальник. Захід: дата, час початку і тривалість; місце проведення; опис; учасники (кількість не обмежена).

ОПИС ПРОГРАМИ

Опис змінних

Scanner scan;	// зчитування даних з клавіатури
int size;	// кількість заходів
SchedulerEvent[] events;	// масив заходів

Ієрархія та структура класів

class Pumnya07 – точка входу в програму.

class SchedulerEvent – клас-планувальник (відображає 1 захід).

ТЕКСТ ПРОГРАМИ

Текст файлу Pumnya05.java

```
package labs.pumnya07;

import java.io.IOException;
import java.text.ParseException;
import java.util.Scanner;
```

```
public final class Pumnya07 {
    private Pumnya07() {
    }
    /**
     * Точка входа.
     * @param args - аргументы функции
     * @throws ParseException если не удалось
     * спарсить дату или время
     * @throws IOException - при
     * некорректном считывании
     */
    public static void main(final String[] args)
            throws IOException, ParseException {
        Scanner scan = new Scanner(System.in);
        System.out.print("Сколько мероприятий"
                       + " добавить? ");
        int size = scan.nextInt();
        scan.nextLine();
        SchedulerEvent[] events = new SchedulerEvent[size];
        for (int i = 0; i < events.length; i++) {
            System.out.format("МЕРОПРИЯТИЕ %d:%n", i + 1);
            events[i] = SchedulerEvent.generate();
        }
        for (int i = 0; i < events.length; i++) {
            System.out.format("%nМЕРОПРИЯТИЕ %d:%n", i + 1);
            System.out.println(events[i].printInfo());
        }
    }
}
```

Текст файлу Pumnya05.java

```
package labs.pumnya07;

import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.text.ParseException;
import java.text.SimpleDateFormat;
import java.util.ArrayList;
import java.util.List;
import java.util.Scanner;
import java.util.regex.Matcher;
import java.util.regex.Pattern;

public class SchedulerEvent {
    /** Хранение даты мероприятия. */
    private String date;
    /** Хранение времени начала мероприятия. */
    private String time;
    /** Хранение длительности мероприятия. */
    private int duration;
    /** Хранение места проведения. */
    private String eventVenue;
    /** Хранение описания мероприятия. */
    private String description;
    /** Хранение участников мероприятия. */
    private List<String> participants;
    /**
     * Получение даты.
     * @return дату
     */
    public String getDate() {
        return this.date;
    }
    /**
     * Установка даты.
     * @param d - дата, которую нужно установить
     */
    public void setDate(final String d) {
        if (this.checkDate(d)) {
            this.date = d;
        }
    }
    /**
     * Проверка даты.
     * @param d - дата
```

```
 * @return - true, если дата подходит формату
 */
private boolean checkDate(final String d) {
    final String df = "dd.MM.yyyy";
    try {
        SimpleDateFormat sdf = new SimpleDateFormat(df);
        sdf.setLenient(false);
        sdf.parse(d);
        return true;
    } catch (ParseException e) {
        return false;
    }
}
/**
 * Получение времени.
 * @return время
 */
public String getTime() {
    return this.time;
}
/**
 * Установка времени.
 * @param t - время, которое нужно установить
 */
public void setTime(final String t) {
    if (checkTime(t)) {
        this.time = t;
    }
}
/**
 * Проверка времени.
 * @param t - время
 * @return - true, если время подходит формату
 */
private boolean checkTime(final String t) {
    final String tf = "HH:mm";
    try {
        SimpleDateFormat sdf = new SimpleDateFormat(tf);
        sdf.setLenient(false);
        sdf.parse(t);
        return true;
    } catch (ParseException e) {
        return false;
    }
}
/**
 * Получение длительности мероприятия.
 * @return длительность
 */
public int getDuration() {
    return this.duration;
}
/**
 * Установка длительности.
 * @param dur - длительность,
 *            которую нужно установить
 */
public void setDuration(final int dur) {
    this.duration = dur;
}
/**
 * Получение места проведения.
 * @return место проведения
 */
public String getVenue() {
    return this.eventVenue;
}

/**
 * Установка места проведения.
 * @param venue - место проведения
 */
public void setVenue(final String venue) {
    if (this.checkVenue(venue)) {
        this.eventVenue = venue;
    }
}
```

```
/**
 * Проверка на корректность
 * ввода места проведения.
 * @param venue - место проведения
 * @return true, если строка
 * удовлетворяет регулярному выражению
 */
public boolean checkVenue(final String venue) {
    Pattern pattern = Pattern.compile(
            "^[А-Я][а-я]+(\\s[А-Я][а-я]+)?");
    Matcher matcher = pattern.matcher(venue);
    return matcher.matches();
}
/**
 * Получение описания мероприятия.
 * @return описание
 */
public String getDescription() {
    return this.description;
}
/**
 * Установка описания мероприятия.
 * @param desc - описание
 */
public void setDescription(final String desc) {
    if (this.checkDesc(desc)) {
        this.description = desc;
    }
}
/**
 * Проверка на корректность
 * ввода описания.
 * @param desc - описание
 * @return true, если строка
 * удовлетворяет регулярному выражению
 */
private boolean checkDesc(final String desc) {
    Pattern pattern = Pattern.compile(
            "^[А-Я][а-я]+(\\s?[А-Я]?[а-я]+)+\\.$");
    Matcher matcher = pattern.matcher(desc);
    return matcher.matches();
}
/**
 * Получение участников мероприятия.
 * @return массив участников
 */
public List<String> getParticipants() {
    return this.participants;
}
/**
 * Сеттер для участников.
 * @param part - набор участников
 */
public void setParticipants(final List<String> part) {
    this.participants = part;
}
/**
 * Установка участников мероприятия.
 * @param partAmount - количество участников
 * @return список участников
 * @throws IOException - при
 * некорректном считывании
 */
public List<String> fillParticipants(final int partAmount)
        throws IOException {
    BufferedReader reader = new BufferedReader(
                            new InputStreamReader(System.in));
    System.out.format("Введите имена"
            + " %s участников.%n", partAmount);
    String name;
    this.participants = new ArrayList<>();
    for (int i = 0; i < partAmount; i++) {
        System.out.format("Участник №%d: ", i + 1);
        name = reader.readLine();
        if (checkName(name)) {
            this.participants.add(name);
        } else {
```

```
            i--;
        }
    }
    return this.participants;
}
/**
 * Проверка на корректность
 * ввода имени участника.
 * @param name - имя
 * @return true, если строка
 * удовлетворяет регулярному выражению
 */
private boolean checkName(final String name) {
    Pattern pattern = Pattern.compile(
            "^[А-Я][а-я]+(\\s[А-Я][а-я]+)?");
    Matcher matcher = pattern.matcher(name);
    return matcher.matches();
}
/**
 * Создание мероприятия.
 * @return новое мероприятие
 * @throws ParseException если не удалось
 * спарсить дату или время.
 * @throws IOException - при
 * некорректном считывании
 */
public static SchedulerEvent generate() throws ParseException, IOException {
    Scanner in = new Scanner(System.in);
    SchedulerEvent se = new SchedulerEvent();
    System.out.print("Введите дату "
                    + "мероприятия (дд.мм.гггг): ");
    se.setDate(in.nextLine());
    System.out.print("Введите время начала"
            + " мероприятия (чч:мм): ");
    se.setTime(in.nextLine());
    System.out.print("Введите длительность"
                    + " мероприятия (в минутах): ");
    se.setDuration(in.nextInt());
    in.nextLine();
    System.out.print("Введите место проведения: ");
    se.setVenue(in.nextLine());
    System.out.print("Введите описание"
                    + " мероприятия: ");
    se.setDescription(in.nextLine());
    System.out.print("Введите количество"
            + " участников: ");
    int amount = in.nextInt();
    in.nextLine();
    List<String> part = se.fillParticipants(amount);
    se.setParticipants(part);
    return se;
}

/**
 * Вывод информации о мероприятии.
 * @return строку с информацией
 */
public String printInfo() {
    StringBuilder builder = new StringBuilder();
    builder.append("Дата: ").append(this.getDate()).append("\n");
    builder.append("Время начала: ").append(
            this.getTime()).append("\n");
    builder.append("Длительность: ").append(
            this.getDuration()).append(" минут\n");
    builder.append("Место проведения: ").append(
            this.getVenue()).append("\n");
    builder.append("Описание: ").append(
            this.getDescription()).append("\n");
    builder.append("Участники: ");
    for (String name : this.getParticipants()) {
        builder.append(name).append(" ");
    }
    return builder.toString();
}
/**
 * Переопределение toString().
 */
```

```
    @Override
    public String toString() {
        return "Дата: " + this.getDate() + "\n"
                + "Время начала: " + this.getTime() + "\n"
                + "Длительность: " + this.getDuration()
                + " минут\n"
                + "Место проведения: " + this.getVenue() + "\n"
                + "Описание: " + this.getDescription() + "\n";

    }
}
```

ВАРІАНТИ ВИКОРИСТАННЯ

```
Сколько мероприятий добавить? 1
МЕРОПРИЯТИЕ 1:
Введите дату мероприятия (дд.мм.гггг): 10.11.2019
Введите время начала мероприятия (чч:мм): 13:15
Введите длительность мероприятия (в минутах): 40
Введите место проведения: Вечерний корпус
Введите описание мероприятия: Обсуждение дальнейших действий.
Введите количество участников: 3
Введите имена 3 участников.
Участник №1: Александр
Участник №2: Сергей
Участник №3: Иван

МЕРОПРИЯТИЕ 1:
Дата: 10.11.2019
Время начала: 13:15
Длительность: 40 минут
Место проведения: Вечерний корпус
Описание: Обсуждение дальнейших действий.
Участники: Александр Сергей Иван

Process finished with exit code 0
```

Рисунок 1 – Результат роботи програми

Програма може бути використана для планування заходів.

ВИСНОВКИ

При виконанні лабораторної роботи набуто практичних навичок щодо розробки класів для заданих прикладних областей.