МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ
РОССИЙСКОЙ ФЕДЕРАЦИИ
ГОУ НИЖЕГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
ИМ. Р.Е. АЛЕКСЕЕВА

ИНСТИТУТ ЭКОНОМИКА И УПРАВЛЕНИЕ

Курсовая работа.

Дисциплина "Технология программирования"

Выполнил:
Краличев Игорь Евгеньевич
студент группы: 21- САИ
Проверил:
Жевнерчук Дмитрий Валерьевич

Нижний Новгород
2022 г.

Содержание

Введение

В объектно-ориентированном программировании (ООП) обычно используется гораздо больше модулей, каждый из которых обеспечивает конкретные функции и может быть изолирован или даже полностью отделен от всех остальных. Такое модульное программирование обеспечивает гораздо большую гибкость и возможности для многократного использования кода.

Также достоинством объектно-ориентированного подхода от традиционного проектирования относят возможность использования итеративно-поступательного цикла создания программного обеспечения и перенос акцента проектирования с разработки алгоритмов функционирования системы на построения системы абстракций и их взаимодействия.

Цель курсовой работы. – разработать требуемую модель документа, научиться работать с ООП и паттернами программирования, а также с языком программирования Java.

1. Постановка задачи

Разработайте объектно-ориентированную модель, которая обеспечивает хранение информации о доступных билетах в кинотеатры, театры, выставки, мастер-классы и т. д. Реализуйте программу, которая позволяет:

1. Приобретать билеты на мероприятия.

2. В процессе приобретения формировать кратковременную бронь.

3. Посещать мероприятия по абонементу.

2. Анализ предметной области

А) Предметная область программы состоит из следующих сущностей:

- Главный класс - обеспечивает обработку данных о количестве свободных билетов и стоимости купленных билетов.
- Классы с данными о мероприятиях- указывается вместимость зала и цена за один билет.

Классы с данными о мероприятиях будут отправляться в главный класс для обработки.

Б) текстовое описание декомпозиции или интеграции сущностей, делегирование функций одними сущностями другим.

Программа будет брать значения атрибутов вместимости зала и стоимости билетов для дальнейшего использования в главном классе. Данное решение позволит в будущем добавлять новые мероприятия.

В) текстовое описание ассоциаций (агрегаций, композиций) и обобщений.

Агрегация (агрегирование по ссылке) — это ассоциация типа «целое-часть». Агрегация в UML представляется в виде прямой с ромбом на конце. Ромб на связи указывает, какой класс является агрегирующим (т.е. «состоящим из»); класс с противоположного конца — агрегированным (т.е. те самые «части»).

Композиция (агрегирование по значению) — это такая агрегация, где объекты-части не могут существовать сами по себе и уничтожаются при уничтожении объекта агрегирующего класса. Композиция изображается так же, как ассоциация, только ромбик закрашен.

Обобщение и наследование позволяют выявить аналогии между различными классами объектов, определяют многоуровневую классификацию объектов.

С помощью композиции будет связан класс, в котором происходит сбор значений вместимости зала и цен на билеты в классах, описывающий мероприятия.

3. Разработка объектно-ориентированной модели предметной области

3.1 Описание структуры классов.

JavaBean — это объект Java, который удовлетворяет определенным соглашениям программирования:

- Класс должен иметь конструктор без параметров, с модификатором доступа public. Такой конструктор позволяет инструментам создать объект без дополнительных сложностей с параметрами.
- Свойства класса должны быть доступны через get, set и другие методы (так называемые методы доступа), которые должны подчиняться стандартному соглашению об именах. Это легко позволяет инструментам автоматически определять и обновлять содержание bean'ов. Многие инструменты даже имеют специализированные редакторы для различных типов свойств.
- Класс должен быть сериализуем. Это даёт возможность надёжно сохранять, хранить и восстанавливать состояние bean независимым от платформы и виртуальной машины способом.
- Класс должен иметь переопределенные методы equals(), hashCode() и toString().

Класс POJO - это plain old Java object, простой Java-объект, не ограниченный какими-либо запретами, специфичными для того или иного фреймворка и пригодный для использования в любой среде.

Многие классы в будущем приложении расширяются из интерфейсов, а также удовлетворяют соглашениям JavaBean, следовательно, структура классов – JavaBean.

3.2 *Описание подсистемы управления созданием объектов, описание применения порождающих шаблонов проектирования.*

В качестве порождающего шаблона выбран паттерн Строитель- он обеспечит более удобный способ написания текстовой информации для отображения пользователю.

3.3 *Описание применения поведенческих шаблонов проектирования.*

В качестве поведенческого паттерна подходит Стратегия – он определяет семейство схожих алгоритмов и помещает каждый из них в собственный класс, после чего алгоритмы можно взаимозаменять прямо во время исполнения программы. В разрабатываемой программе такими алгоритмами являются сбор данных о вместимости зала и стоимости билетов на мероприятия.

3.4 *Разработка объектно-ориентированной модели предметной области (с построением диаграммы классов). Обоснование слабой межмодульной связанности и расширяемости системы.*

Cinema, Exhibition, Theatre, MasterClass – Классы, в которых хранятся данные о вместимости зала и стоимости билета.

Helper – Класс, собирающий данные о вместимости зала и цене билета.

App – Класс, который выводит и хранит данные о изменении вместимости зала.

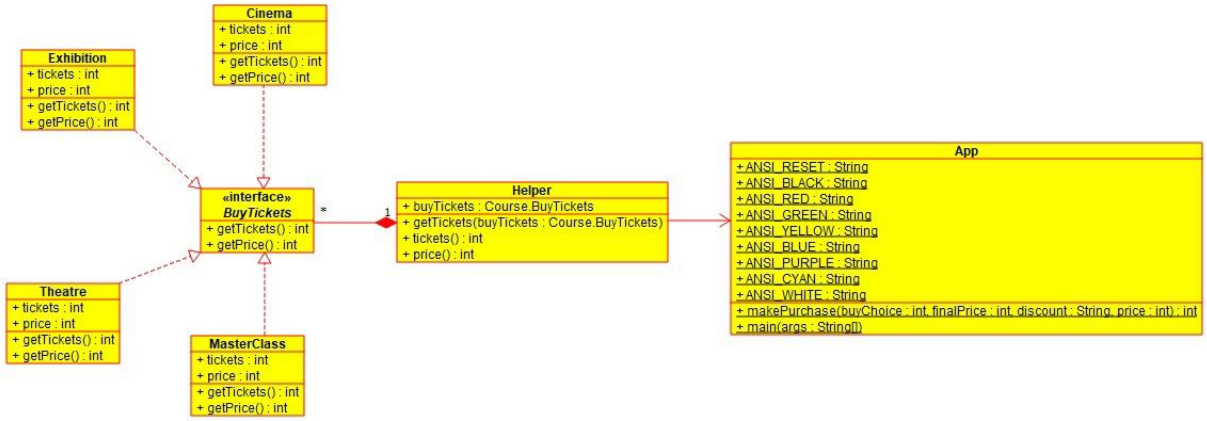


Рисунок 1 Диаграмма классов

Система может быть расширяема по другим мероприятиям. Связность модуля определяется как мера независимости его частей. Чем выше связность модуля, тем больше отдельные части модуля зависят друг от друга и тем лучше результат проектирования. Слабая связанность означает, что модули должны быть максимально независимы друг от друга, таким образом изменения в одном модуле не влияют на другие модули.

4. Реализация объектно-ориентированной библиотеки и приложения:

4.1 Аналитический обзор языков программирования (привести краткую характеристику 2-3 варианта). Описать преимущества и недостатки каждого инструмента, обосновать выбор языка программирования для решения задачи по варианту.

В качестве методологии программирования было выбрано объектно-ориентированное программирование. Самыми популярными и востребованными языками ООП являются: Java, Python, C++, Kotlin, JavaScript, Ruby.

Java — язык программирования общего назначения. Это объектно-ориентированный язык. Вся структура строится вокруг объектов, классов, инстансов и прочих формальных сущностей, принятых в сообществе программистов за стандарт разработки ООП. Код, написанный на этом языке, запускается в виртуальной машине JVM и без проблем инициализируются в любой системе, где поддерживается соответствующая виртуальная машина.

Плюсы:

1. Независимый код. Любая платформа, которая поддерживает виртуальную машину Java, воспроизведет ваш код.
2. Надежный код. Строгая статистическая типизация дает главное преимущество — надежность вашего кода.
3. Высокая функциональность. На Java можно написать практически все: от простого приложения на смартфон до программ по машинному обучению для беспилотных автомобилей.
4. Синтаксис средней сложности. Данный язык поддается изучению новичкам, которые раньше вообще не имели дела с программированием.
5. Java для Андроид. Андроид — самая популярная ОС для смартфонов, а Java — самый популярный язык для приложений на Андроид, соответственно, изучив Java, будет очень широкое поле для деятельности.

Минусы:

1. Более низкая производительность. За счет свой специфики Java во многих случаях работает медленнее, чем другие языки, такие как: C, C#, C++, Python.
2. Потребляет память. Опять же, за счет своей специфики работы данный язык требует больше памяти, чем многие сторонние языки.
3. Платность. Буквально с 2019 года для коммерческо-юридических проектов язык Java стал платным, но для частного использования он абсолютно бесплатен.

Ruby — динамический, рефлективный, интерпретируемый высокоуровневый язык программирования. Язык обладает независимой от операционной системы реализацией многопоточности, сильной динамической типизацией, сборщиком мусора и многими другими возможностями. По особенностям синтаксиса он близок к языкам Perl и Eiffel, по объектно-ориентированному подходу — к Smalltalk. Также некоторые черты языка взяты из Python, Lisp, Dylan и CLU.

Плюсы:

1. открытая разработка;
2. работает на многих платформах;
3. может внедряться в **HTML**-разметку;
4. относится к языкам программирования сверхвысокого уровня, то есть обладает высоким уровнем абстракции и предметным подходом в реализации алгоритмов;
5. реализует концептуально чистую объектно-ориентированную парадигму;
6. предоставляет продвинутые методы манипуляции строками и текстом;
7. легко интегрирует в свои программы высокопроизводительные серверы баз данных;
8. простой и чистый синтаксис значительно облегчает программистам первые шаги в обучении этому языку;
9. имеется простой программный интерфейс для создания многопоточных приложений;
10. имеет продвинутые средства для работы с массивами;
11. возможности языка можно расширить при помощи библиотек, написанных на **C** или **Ruby**;
12. зарезервированные слова могут являться идентификаторами, если это не создаёт неоднозначности для парсера;
13. дополнительные возможности для обеспечения безопасности;
14. встроенный отладчик.

Минусы:

1. обучение языку выше начального уровня может оказаться непростым;
2. информационных ресурсов, посвящённых **Ruby**, явно недостаточно;
3. Ruby менее производителен по сравнению со многими другими языками, применяемыми в веб-разработке;
4. Ruby относительно медленно разрабатывается и развивается.

Kotlin — это объектно-ориентированный язык со статической типизацией. Объектно-ориентированными называют языки, в которых все операции происходят с объектами — блоками кода, куда можно «складывать» несколько значений. Объектом может быть любая сущность с определённым набором характеристик. Статическая типизация означает, что типы переменных задаются разработчиком до выполнения программы. Если объявить целочисленную переменную, текст в неё поместить уже не получится — компилятор сразу укажет на ошибку. Это замедляет разработку на начальном этапе, но добавляет надёжности.

Плюсы:

1. совместимость с Java – оба языка можно использовать в одном проекте. И что самое приятное – можно внедрять Котлин в уже существующие проекты на JS без сложностей и негативных последствий. Даже наоборот, это один из способов усовершенствовать работу. Но совместимость поддерживается еще не со всеми Java-фреймворками и библиотеками, это вопрос времени;
2. чистота кода – никаких избыточных кусков, все компактно, красиво и выразительно. В среднем программный код на Котлине выходит на 40% короче в сравнении с другими языками;
3. безопасность – предусмотрена автоматическая проверка на этапе компиляции. Особенности языка позволяют сразу обнаружить ссылку на null в константах, полях и переменных. Null safety считается одним из главных преимуществ.
4. возможность сделать компактный код при помощи таких функций как .apply или .with.

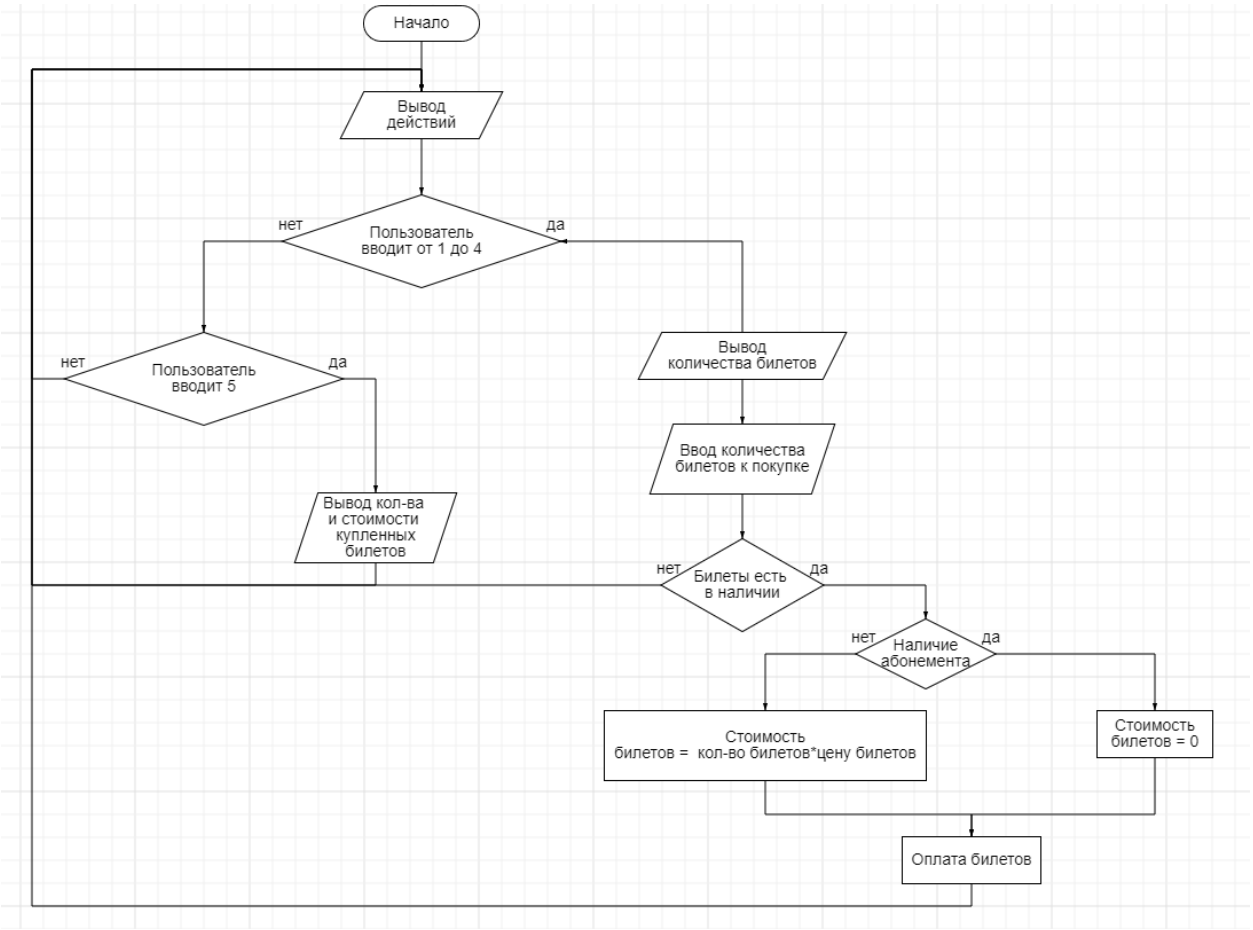
Минусы:

1. низкая скорость компиляции в сравнении с Java;
2. медленное развитие – обновления и новые библиотеки выпускаются редко;
3. долго исправляются баги – задачи закрываются по мере критичности, малочисленная команда просто не успевает исправлять десятки тысяч недоработок.

По итогу основным языком разработки был выбран язык Java. Компилятор Java производит байт-коды, т.е. модули приложения имеют архитектурно-независимый формат, который может быть проинтерпретирован на множестве разнообразных платформ. Это уже не исходные тексты, но еще не платформно-зависимые машинные коды. Для решения задачи курсовой работы он подходит больше, т.к. имеет ряд преимуществ. Он обеспечивает хорошую организацию кода и его легкую поддержку, более удобен для организации структуры программ, достаточно простой синтаксис и возможность параллельной разработки.

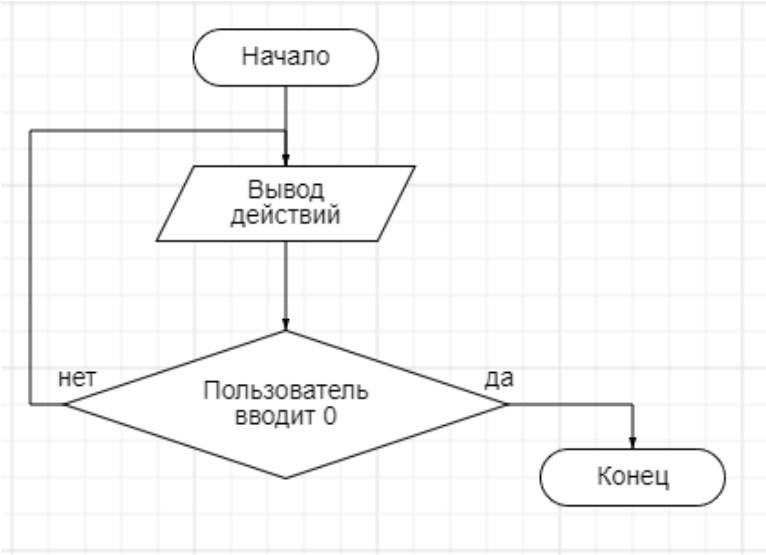
4.2 *Описание алгоритмов, реализуемых программно. Описание реализации методов.*

Блок-схема алгоритма покупки билета и вывода купленных билетов. Пользователь вводит номер действия. Если пользователь выбрал покупку билета, то вводит необходимое количество билетов, далее вводит сведения о наличии абонемента и оплачивает покупку. Если пользователь выбрал вывод купленных билетов, то ему предоставляется информация о количестве купленных билетов и их стоимости.



Блок-схема 1. Покупка билета и вывод купленных билетов

Блок-схема завершения работы, позволяющая пользователю работать с программой до тех пор, пока он самостоятельно не прекратит работу, введя цифру 0.



Блок-схема 2. Завершение работы программы

4.3 *Описание тестового примера: входные, выходные и промежуточные данные для каждого теста, привести скриншоты тестов, из которых очевидна корректность работы программы* .

```
Вас приветствует программа покупки билетов!
Для завершения работы программы введите 0.
Для продолжения выберите мероприятие:
1:Кино
2:Спектакль
3:Мастер-класс
4:Выставка
Для просмотра билетов нажмите 5.
```

Рисунок 2. Начало работы программы

Пользователь выбирает Кино:

```
Для продолжения выберите мероприятие:
1:Кино
2:Спектакль
3:Мастер-класс
4:Выставка
Для просмотра билетов нажмите 5.

1
Билетов в наличии на Киносеанс:200 шт.
Стоимость билета:150 руб.
Какое кол-во билетов желаете приобрести?
```

Рисунок 3. Выбор мероприятия

Пользователь выбрал 20 билетов и вводит данные о наличии абонемента

```
Билетов в наличии на Киносеанс:200 шт.
Стоимость билета:150 руб.
Какое кол-во билетов желаете приобрести?
20
У Вас есть абонемент на мероприятие?(y-Да/n-Нет)
y
Оплата билета...
Программа продолжит выполнение через 3 сек.

Билет успешно оплачен. Заказ можно посмотреть по коду 5.
Выберите следующее действие или выключите программу.
```

Рисунок 4. Успешная покупка билета по абонементу

```
Билетов в наличии на Киносеанс:180 шт.
Стоимость билета:150 руб.
Какое кол-во билетов желаете приобрести?
20
У Вас есть абонемент на мероприятие?(y-Да/n-Нет)
n
Оплата билета...
Программа продолжит выполнение через 3 сек.

Билет успешно оплачен. Заказ можно посмотреть по коду 5.
Выберите следующее действие или выключите программу.
```

Рисунок 5. Успешная покупка билета без абонемента

Пользователь просматривает купленные билеты и их стоимость:

```
Для просмотра билетов нажмите 5.

5
Вы приобрели билетов на Киносеанс:40 шт.
Общая сумма:3000 руб.

Вы приобрели билетов на Спектакль:0 шт.
Общая сумма:0 руб.

Вы приобрели билетов на Мастер-класс:0 шт.
Общая сумма:0 руб.

Вы приобрели билетов на Выставку:0 шт.
Общая сумма:0 руб.
```

Рисунок 6. Вывод купленных билетов и их стоимость

Завершение работы программы:

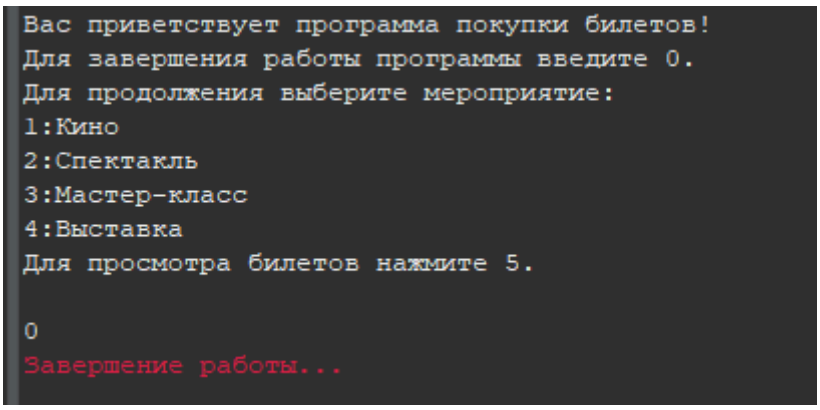


Рисунок 7. Завершение работы

Заключение

Разработанное консольное приложение даёт возможность пользователю совершать покупку билета на мероприятие. Преимуществом данной программы является простота использования и интуитивно понятный интерфейс. Недостатком является невозможность дистанционной оплаты билета, так как предполагается развёртывание данного приложения на рабочем месте кассира. В перспективе дальнейшей разработки реализовать клиент-серверную связь для дистанционной покупки и бронировании билета. Также реализовать добавление мероприятий не только разработчиками, но и владельцами заведений, использовав интуитивно понятную структуру добавления мероприятия.

Список литературы

1. Паттерны проектирования в Java / https://javarush.com/groups/posts/2267-patternih-proektirovanija-v-java#Поведенческие_шаблоны

2. JavaBeans / https://ru.wikipedia.org/wiki/JavaBeans

3. POJO / https://www.baeldung.com/java-pojo-class

3. Языки объектно-ориентированного программирования / https://bestprogrammer.ru/programmirovanie-i-razrabotka/yazyki-obektno-orientirovannogo-programmirovaniya

4. Язык программирования Ruby / https://ru.hexlet.io/blog/posts/yazyk-programmirovaniya-ruby-osobennosti-perspektivy-rynok-truda

5. Kotlin – что нужно знать разработчику о новом языке / https://wayup.in/ru/blog/kotlin

6. Плюсы и минусы Java / https://www.cischool.ru/plyusy-i-minusy-java/

7. Плюсы и минусы Ruby / https://www.internet-technologies.ru/articles/php-ruby-python-harakteristika-yazykov-programmirovaniya.html

Приложения

Приложение 1 «Программный код»

Код класса «Cinema»:

```
package Course;

import java.util.Scanner;

public class Cinema implements BuyTickets{
	public int tickets = 200;
	public int price = 150;

	// Вывод количества билетов
	@Override
	public int getTickets() {
		return this.tickets;
	}

	// Вывод стоимости билета
	@Override
	public int getPrice() {
		return this.price;
	}
}
```

Код класса «Exhibition»:

```
package Course;

import java.util.Scanner;

public class Exhibition implements BuyTickets{
	public int tickets = 200;
	public int price = 250;

	// Вывод количества билетов
	@Override
	public int getTickets() {
		return this.tickets;
	}

	// Вывод стоимости билета
	@Override
	public int getPrice() {
		return this.price;
	}
}
```

Код класса «Theatre»:

```
package Course;
```

```
import java.util.Scanner;

public class Theatre implements BuyTickets{
	public int tickets = 200;
	public int price = 300;

	// Вывод количества билетов
	@Override
	public int getTickets() {
		return this.tickets;
	}

	// Вывод стоимости билета
	@Override
	public int getPrice() {
		return this.price;
	}
}
```

Код класса «MasterClass»:

```
package Course;

import java.util.Scanner;

public class MasterClass implements BuyTickets{
	public int tickets = 200;
	public int price = 500;

	// Вывод количества билетов
	@Override
	public int getTickets() {
		return this.tickets;
	}

	// Вывод стоимости билета
	@Override
	public int getPrice() {
		return this.price;
	}
}
```

Код интерфейса «BuyTickets»:

```
package Course;

public interface BuyTickets {
	public int getTickets();
	public int getPrice();
}
```

Код класса «Helper»:

```
package Course;

public class Helper {
	BuyTickets buyTickets;

	public void getTickets(BuyTickets buyTickets) {
		this.buyTickets=buyTickets;
	}
	public int tickets() {
		return buyTickets.getTickets();
	}
	public int price() {
		return buyTickets.getPrice();
	}
}
```

Код главного класса «App»:

```
package Course;


import java.util.Scanner;

import java.util.concurrent.TimeUnit;


public class App {

	//Цвета символов в консоли

	public static final String ANSI_RESET = "\u001B[0m";

	public static final String ANSI_BLACK = "\u001B[30m";

	public static final String ANSI_RED = "\u001B[31m";

	public static final String ANSI_GREEN = "\u001B[32m";

	public static final String ANSI_YELLOW = "\u001B[33m";

	public static final String ANSI_BLUE = "\u001B[34m";

	public static final String ANSI_PURPLE = "\u001B[35m";

	public static final String ANSI_CYAN = "\u001B[36m";

	public static final String ANSI_WHITE = "\u001B[37m";


	// Метод подсчёта стоимости билетов после ввода наличия/отсутствия абонемента

	public static int makePurchase(int buyChoice, int finalPrice, String discount, int price) {
```

```
        if (discount.equalsIgnoreCase("n")) {
                finalPrice = finalPrice + buyChoice * price;
        } else if (discount.equalsIgnoreCase("y")) {
                finalPrice = 0;
        }
        return finalPrice;
}

public static void main(String[] args) {
        // Приветствие и предложение действий

        // Объявление переменных
        int choice = 1; // Выбор мероприятия
        int buyChoice; // Выбор количества билетов
        int buyStatisticCinema = 0; // Статистика покупки билетов
        int buyStatisticTheatre = 0;
        int buyStatisticMC = 0;
        int buyStatisticExhibition = 0;
        String discount; //Наличие абонемента
        int finalPriceCinema = 0; // Окончательная цена
        int finalPriceTheatre = 0;
        int finalPriceMC = 0;
        int finalPriceExhibition = 0;

        // Количество мест в зале и стоимость билетов по мероприятиям
        // Поведенческий паттерн "стратегия"
        Helper helper = new Helper();
        // Киносеанс
        helper.getTickets(new Cinema());
        int buyCinema = helper.tickets();
        int priceCinema = helper.price();
```

```
// Выставка
helper.getTickets(new Exhibition());
int buyExhibition = helper.tickets();
int priceExhibition = helper.price();
// Спектакль
helper.getTickets(new Theatre());
int buyTheatre = helper.tickets();
int priceTheatre = helper.price();
// Мастер-класс
helper.getTickets(new MasterClass());
int buyMC = helper.tickets();
int priceMC = helper.price();

// Паттерн проектирования "строитель"
// Приветствие
StringBuilder builderHello = new StringBuilder();
builderHello.append("Вас приветствует программа покупки билетов!\n");
builderHello.append("Для завершения работы программы введите 0.\n");
builderHello.append("Для продолжения выберите мероприятие:\n");
builderHello.append("1:Кино\n");
builderHello.append("2:Спектакль\n");
builderHello.append("3:Мастер-класс\n");
builderHello.append("4:Выставка\n");
builderHello.append("Для просмотра билетов нажмите 5.\n");
builderHello.append(ANSI_RESET);

// Успешная оплата
StringBuilder builderSuccess = new StringBuilder();
builderSuccess.append(ANSI_GREEN + "Билет успешно оплачен. Заказ можно посмотреть по коду 5.\n" + ANSI_RESET);
builderSuccess.append("Выберите следующее действие или выключите программу.\n");
```

```
// Выбор действия и проверка правильности ввода
try (Scanner in = new Scanner(System.in)) {
    // Цикл выборки
    while (choice != 0) {
        System.out.println(builderHello.toString());
        choice = in.nextInt();
        // Выборка выбора
        switch (choice) {
        // Покупка билета в кино
        case 1:
            System.out.println("Билетов в наличии на Киносеанс:" + buyCinema + " шт.\n" + "Стоимость билета:"
                    + priceCinema + " руб.\n" + "Какое кол-во билетов желаете приобрести?");
            buyChoice = in.nextInt();
            in.nextLine();
            if (buyChoice < 0) {
                System.err.println(
                        "Число билетов не может быть отрицательным!\n" + "Выберите другое мероприятие.\n");
            } else if (buyChoice < buyCinema) {
                buyCinema = buyCinema - buyChoice;
                System.out.println("У Вас есть абонемент на мероприятие?(y-Да/n-Нет)");
                discount = in.nextLine();
                if (discount.equalsIgnoreCase("y") == false && discount.equalsIgnoreCase("n") == false) {
                    buyCinema = buyCinema + buyChoice;
                    System.err.println("Ответ неверный! Повторите попытку покупки ещё раз!\n");
                    break;
                } else {
```

```
                                        finalPriceCinema = makePurchase(buy-
Choice, finalPriceCinema, discount, priceCinema);
                                        buyStatisticCinema = buyStatisticCinema +
buyChoice;
                                        System.out.println("Оплата билета...\n" +
"Программа продолжит выполнение через 3 сек.\n");
                                        // Остановка выполнения на 3 секунды
(нужно для разделения оплаты и продолжения
                                        // работы)
                                        TimeUnit.SECONDS.sleep(3);
                                        System.out.println(builderSuc-
cess.toString());
                                }
                        } else {
                                System.err.println(
                                        "На данное мероприятие нет
столько билетов!\n" + "Выберите другое мероприятие.\n");
                        }
                        break;


                // Покупка билета в театр
                case 2:
                        System.out.println("Билетов в наличии на Спектакль:" +
buyTheatre + " шт.\n" + "Стоимость билета:"
                                        + priceTheatre + " руб.\n" + "Какое кол-во
билетов желаете приобрести?");
                        buyChoice = in.nextInt();
                        in.nextLine();
                        if (buyChoice < buyTheatre) {
                                buyTheatre = buyTheatre - buyChoice;
                                System.out.println("У Вас есть абонемент на ме-
роприятие?(Y-Да/N-Нет)");
                                discount = in.nextLine();
```

```
                                        if (discount.equalsIgnoreCase("y") == false && dis-
count.equalsIgnoreCase("n") == false) {
                                            buyTheatre = buyTheatre + buyChoice;
                                            System.err.println("Ответ неверный! По-
вторите попытку покупки ещё раз!\n");
                                            break;
                                        }
                                        finalPriceTheatre = makePurchase(buyChoice, final-
PriceTheatre, discount, priceTheatre);
                                        buyStatisticTheatre = buyStatisticTheatre + buy-
Choice;
                                        System.out.println("Оплата билета...\n" +
"Программа продолжит выполнение через 3 сек.\n");
                                        // Остановка выполнения на 3 секунды (нужно
для разделения оплаты и продолжения
                                        // работы)
                                        TimeUnit.SECONDS.sleep(3);
                                        System.out.println(builderSuccess.toString());
                                    } else {
                                        System.err.println(
                                                "На данное мероприятие нет
столько билетов!\n" + "Выберите другое мероприятие.\n");
                                    }
                                    break;

                                // Покупка билета на мастер-класс
                                case 3:
                                    System.out.println("Билетов в наличии на Мастер-класс:"
+ buyMC + " шт.\n" + "Стоимость билета:"
                                            + priceMC + " руб.\n" + "Какое кол-во би-
летов желаете приобрести?");
                                    buyChoice = in.nextInt();
                                    in.nextLine();
                                    if (buyChoice < buyMC) {
```

```
                                        buyMC = buyMC - buyChoice;
                                        System.out.println("У Вас есть абонемент на мероприятие?(Y-Да/N-Нет)");
                                        discount = in.nextLine();
                                        if (discount.equalsIgnoreCase("y") == false && discount.equalsIgnoreCase("n") == false) {
                                                buyMC = buyMC + buyChoice;
                                                System.err.println("Ответ неверный! Повторите попытку покупки ещё раз!\n");
                                                break;
                                        }
                                        finalPriceMC = makePurchase(buyChoice, finalPriceMC, discount, priceMC);
                                        buyStatisticMC = buyStatisticMC + buyChoice;
                                        System.out.println("Оплата билета...\n" + "Программа продолжит выполнение через 3 сек.\n");
                                        // Остановка выполнения на 3 секунды (нужно для разделения оплаты и продолжения
                                        // работы)
                                        TimeUnit.SECONDS.sleep(3);
                                        System.out.println(builderSuccess.toString());
                                } else {
                                        System.err.println(
                                                        "На данное мероприятие нет столько билетов!\n" + "Выберите другое мероприятие.\n");
                                }
                                break;

                        // Покупка билета на выставку
                        case 4:
                                System.out.println("Билетов в наличии на Выставку:" + buyExhibition + " шт.\n" + "Стоимость билета:"
                                                + priceExhibition + " руб.\n" + "Какое кол-во билетов желаете приобрести?");
```

```
                    buyChoice = in.nextInt();
                    in.nextLine();
                    if (buyChoice < buyMC) {
                        buyExhibition = buyExhibition - buyChoice;
                        System.out.println("У Вас есть абонемент на мероприятие?(Y-Да/N-Нет)");
                        discount = in.nextLine();
                        if (discount.equalsIgnoreCase("y") == false && discount.equalsIgnoreCase("n") == false) {
                            buyExhibition = buyExhibition + buyChoice;
                            System.err.println("Ответ неверный! Повторите попытку покупки ещё раз!\n");
                            break;
                        }
                        finalPriceExhibition = makePurchase(buyChoice, finalPriceExhibition, discount, priceExhibition);
                        buyStatisticExhibition = buyStatisticExhibition + buyChoice;
                        System.out.println("Оплата билета...\n" + "Программа продолжит выполнение через 3 сек.\n");
                        // Остановка выполнения на 3 секунды (нужно для разделения оплаты и продолжения
                        // работы)
                        TimeUnit.SECONDS.sleep(3);
                        System.out.println(builderSuccess.toString());
                    } else {
                        System.err.println(
                                "На данное мероприятие нет столько билетов!\n" + "Выберите другое мероприятие.\n");
                    }
                    break;

                case 5:
                    // Сбор статистики
```

```
                    StringBuilder builderStatistics = new StringBuilder();
                    // Билеты в кино
                    builderStatistics.append("Вы приобрели билетов на Кино-
сеанс:" + buyStatisticCinema + " шт.\n");
                    builderStatistics.append("Общая сумма:" + finalPriceCin-
ema + " руб.\n\n");
                    // Билеты в театр
                    builderStatistics.append("Вы приобрели билетов на Спек-
такль:" + buyStatisticTheatre + " шт.\n");
                    builderStatistics.append("Общая сумма:" + finalPriceThea-
tre + " руб.\n\n");
                    // Билеты на Мастер-класс
                    builderStatistics.append("Вы приобрели билетов на Ма-
стер-класс:" + buyStatisticMC + " шт.\n");
                    builderStatistics.append("Общая сумма:" + finalPriceMC +
" руб.\n\n");
                    // Билеты на выставку
                    builderStatistics.append("Вы приобрели билетов на Вы-
ставку:" + buyStatisticExhibition + " шт.\n");
                    builderStatistics.append("Общая сумма:" + finalPriceExhi-
bition + " руб.\n\n");

                    // Вывод статистики
                    System.out.println(builderStatistics.toString());
                    break;

                    // Выход из системы
                case 0:
                    System.err.println("Завершение работы...");
                    System.exit(0);
                }

                // Проверка на наличие действия
                if (choice < 0 || choice > 5) {
```

```
                                System.err.println("Выберите цифру от 0 до 5!");
                            }
                        }
                    }

                    // Обработка исключения ввода числа
                    catch (Exception e) {
                        System.err.println("Вводить можно только цифры.\n" + "Перезапустите программу для продолжения.");

                    }
                }
            }
```

Приложение 2 «Руководство пользователя»

1) Пользователь запускает приложение и выбирает мероприятие (ввод цифры от 1 до 4):

```
Вас приветствует программа покупки билетов!
Для завершения работы программы введите 0.
Для продолжения выберите мероприятие:
1:Кино
2:Спектакль
3:Мастер-класс
4:Выставка
Для просмотра билетов нажмите 5.
```

Рисунок 8. Начало работы программы

2) Пользователь выбирает нужное количество билетов для покупки:

```
Для продолжения выберите мероприятие:
1:Кино
2:Спектакль
3:Мастер-класс
4:Выставка
Для просмотра билетов нажмите 5.

1
Билетов в наличии на Киносеанс:200 шт.
Стоимость билета:150 руб.
Какое кол-во билетов желаете приобрести?
```

Рисунок 9. Выбор мероприятия

3) Пользователь вводит информацию о наличии абонемента (y-Да/n-Нет) и оплачивает покупку кассиру.

```
Билетов в наличии на Киносеанс:200 шт.
Стоимость билета:150 руб.
Какое кол-во билетов желаете приобрести?
20
У Вас есть абонемент на мероприятие?(y-Да/n-Нет)
y
Оплата билета...
Программа продолжит выполнение через 3 сек.

Билет успешно оплачен. Заказ можно посмотреть по коду 5.
Выберите следующее действие или выключите программу.
```

Рисунок 10. Успешная покупка билета по абонементу

4) Пользователь вводит цифру 5 для просмотра купленных билетов и получает следующие данные:

```
Для просмотра билетов нажмите 5.

5
Вы приобрели билетов на Киносеанс:40 шт.
Общая сумма:3000 руб.

Вы приобрели билетов на Спектакль:0 шт.
Общая сумма:0 руб.

Вы приобрели билетов на Мастер-класс:0 шт.
Общая сумма:0 руб.

Вы приобрели билетов на Выставку:0 шт.
Общая сумма:0 руб.
```

Рисунок 11. Вывод купленных билетов и их стоимость

5) Пользователь вводит цифру 0 для завершения работы программы

```
Вас приветствует программа покупки билетов!
Для завершения работы программы введите 0.
Для продолжения выберите мероприятие:
1:Кино
2:Спектакль
3:Мастер-класс
4:Выставка
Для просмотра билетов нажмите 5.

0
Завершение работы...
```

Рисунок 12. Завершение работы