|  |
| --- |
| Изображение выглядит как текст, коллекция картинок, фарфор  Автоматически созданное описание  МИНОБРНАУКИ РОССИИ |
| Федеральное государственное бюджетное образовательное учреждение  высшего образования  **«МИРЭА – Российский технологический университет»**  **РТУ МИРЭА** |
| Институт кибербезопасности и цифровых технологий |
| Кафедра КБ-2 «Прикладные информационные технологии» |

**ЗАЧТЕНО**

\_\_\_\_\_\_\_\_\_\_ /Лесько С.А./

«\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_ 20\_\_\_\_г.

**ОТЧЕТ ПО ПРАКТИЧЕСКОЙ РАБОТЕ №1  
по дисциплине  
«Технологии и методы программирования»**

***(наименование дисциплины)***

**Выполнил**

Студент 3 курса: Жидков Г.С.

Группы: БИСО-01-20

Специальности: 10.05.04

Москва 2023

**Оглавление**

[Постановка задачи 3](#_Toc132631078)

[Диаграмма вариантов использования 4](#_Toc132631079)

[Диаграмма классов 5](#_Toc132631080)

[Приложение 7](#_Toc132631081)

Постановка задачи

1. При помощи программы PlantUML либо любого редактора построить UML-диаграмму вариантов использования, диаграмму классов проектируемой информационной системы в соответствии с вариантом задания, а также диаграмму последовательности для наиболее часто используемых прецедентов . При построении диаграммы классов нужно добиться достаточной детализации информационной системы. Убедитесь в том, что использовали отношения dependency, aggregation/c¬omposition, generalization, описали размещение классов по пакетам проекта.

2. Подготовить отчет с включением диаграмм. Загрузить в GIT. Прикрепить ссылку

**Вариант 10.** Система продажи билетов на футбол

Обзор: система продажи билетов позволяет покупать и сдавать билеты и абонементы на матчи, проходящие на одном стадионе с нумерованными местами через несколько одновременно работающих касс.

Диаграмма вариантов использования

На Рисунках 1 и 2 представлена Use Case диаграмма для наиболее часто используемых прецедентов. Код диаграмм представлен в Приложении.

Изображение выглядит как диаграмма

Автоматически созданное описание

Рисунок 1 – Диаграмма использования для покупки билета

Изображение выглядит как диаграмма

Автоматически созданное описание

Рисунок 2 – Диаграмма использования для возврата билета

*Описание.*

Покупатель может купить билет, что включает в себя выбор места и ввод персональных данных. Кассир принимает оплату картой или наличными и вносит данные о продаже билета в базу.

Когда человеку надо сдать билет, то у него есть возможность описать причину возврата. Кассир же возвращает денежные средства, используя реквизиты карты. После этого он вносит данные о возврате и выставляет билет обратно на продажу.

Диаграмма классов

На Рисунке 3 представлена диаграмма классов для системы продажи билетов на футбол. Код диаграммы представлен в Приложении.

Изображение выглядит как диаграмма

Автоматически созданное описание

Рисунок 4 – Диаграмма классов

*Описание.*

Человек - является обобщением двух классов. Имеет имя и фамилию человека. Можно их ввести или получить.

Болельщик – является расширением класса «Человек». Может оплачивать билеты или абонементы. Имеет определённый бюджет и персональные данные.

Кассир – является расширением класса «Человек». Имеет собственный атрибут – обработка заказа (получается от клиента). Может выдать билет.

Касса – имеет имя кассира, работающего на ней, конкретного клиента и заказ. Изменяет информацию о наличии заказа, заполняет данные и выдает чек, проверяет оплату.

Билет/Абонемент – имеет код. Может вернуть количество матчей, на которые можно сходить.

Матч – содержит дату и время, когда пройдёт, команды, которые будут участвовать и спонсора. Проводит игру.

Приложение

Ссылка на репозиторий GitHub:<https://github.com/st-georgy/TMP>

**Исходный код диаграммы использования покупки билета:**

@startuml

actor Покупатель

actor Кассир

usecase "Покупка билета" as UC1

usecase "Выбор места" as UC2

usecase "Ввод персональных данных" as UC3

usecase "Прием оплаты" as UC4

usecase "Оплата картой" as UC5

usecase "Оплата наличными" as UC6

usecase "Внос данных о продаже в базу" as UC7

Покупатель --> UC1

UC1 ..> UC2 : include

UC1 ..> UC3 : include

UC1 --> Кассир

Кассир --> UC4

UC4 ..> UC5 : extends

UC4 ..> UC6 : extends

Кассир --> UC7

@enduml

**Исходный код диаграммы использования возврата билета:**

@startuml

actor Покупатель

actor Кассир

usecase "Сдача билета" as UC1

usecase "Описание причины" as UC2

usecase "Возврат денежных средств" as UC3

usecase "Предоставление реквизитов карты" as UC4

usecase "Выставление билета на продажу" as UC5

usecase "Внос данных о возврате в базу" as UC6

Покупатель --> UC1

UC1 ..> UC2 : extends

UC1 --> Кассир

Кассир --> UC3

UC3 ..> UC4 : uses

Кассир --> UC5

Кассир --> UC6

@enduml

**Исходный код диаграммы классов**

@startuml

left to right direction

class Матч {

+date : Timestamp

+teams : Text

+sponsor : String

+getGame() : Game

}

class Болельщик {

- phoneNumber : Bigint

- balance : Float

- email : String

+ Pay() : Float

}

class БилетАбонемент {

+ code : Bigint

+ getActivateNum() : Int

}

class Кассир {

- orderProcessing : Order

+ getTicket() : Boolean

}

class Касса {

- cost : Float

- employeeName : String

- currentClient : String

- orderProcessing : Order

+ getPayment() : Boolean

+ changeInfo(orderProcessing) : void

+ issueCheque(employeeName) : String

}

class Человек {

firstName : String

lastName : String

+ setName(firstname, lastname) : void

+ getName() : String

}

Человек --> Болельщик : extends

Человек --> Кассир : extends

Болельщик --> Кассир : Покупает билет

Кассир --> Касса : Проводит оплату

Кассир --> БилетАбонемент : Выдает

Болельщик --> Матч : Смотрит

Матч --> БилетАбонемент

@enduml