

Предметная область - Фитнес-центр

Общее описание системы

Нужно уметь:

- вести учёт клиентов,
- осуществлять онлайн-запись на тренировки,
- контролировать абонементы,
- управлять расписанием,
- обрабатывать платежи,
- формировать отчёты,
- управлять сотрудниками и тренерами.

Акторы

1. Клиент

Посещает фитнес-центр, покупает абонементы, записывается на занятия.

Основные действия:

- просматривать расписание;
- записываться на тренировки;
- отменять запись;
- покупать абонементы;
- просматривать историю посещений;
- оплачивать услуги;
- просматривать доступные тренажёры и оборудование.

2. Тренер

Проводит персональные тренировки и групповые занятия.

Основные действия:

- просматривать расписание своих занятий;
- отмечать посещаемость;
- создавать планы тренировок;
- вносить отчёты о занятиях;
- отправлять рекомендации клиентам.

3. Администратор

Работает с клиентами в клубе.

Основные действия:

- регистрировать новых клиентов;
 - оформлять абонементы;
 - проверять доступ по абонементу;
 - производить возвраты и корректировки;
 - записывать клиентов на тренировки вручную;
 - подтверждать оплату наличными.
-

4. Менеджер

Управляет услугами и персоналом.

Основные действия:

- формировать отчёты (по посещаемости, продажам, тренерам);
 - создавать расписание;
 - редактировать список услуг;
 - управлять тренерами (назначение на занятия);
 - контролировать использование залов;
 - управлять скидками и акциями.
-

5. Платёжная система

Принимает онлайн-оплату.

Функциональные требования к системе

Для клиента:

1. Просмотр расписания занятий.
2. Поиск тренировок по типу, тренеру, дате.
3. Запись на тренировку.
4. Отмена записи.
5. Покупка абонемента.
6. Оплата услуг (через внешнюю платёжную систему).
7. Просмотр истории посещений.

8. Просмотр информации о тренерах.
9. Получение рекомендаций от тренера.

Для тренера:

1. Просмотр расписания своих занятий.
2. Отметка посещаемости.
3. Ведение плана тренировок.
4. Отправка рекомендаций клиентам.
5. Создание отчётов по занятиям.

Для администратора:

1. Регистрация клиента.
2. Оформление абонемента.
3. Проверка доступа (валидности абонемента).
4. Ручная запись клиента на занятие.
5. Подтверждение оплаты наличными.
6. Возвраты и корректировки.

Для менеджера клуба:

1. Формирование отчётов (посещаемость, продажи, рейтинг тренеров).
2. Создание и редактирование расписания.
3. Управление тренерами: назначение на занятия.
4. Управление услугами.
5. Управление акциями и скидками.

Сущности для диаграммы классов

Основные классы:

- Человек (общий класс)
- Клиент
- Тренер
- Администратор
- Менеджер
- Абонемент
- Тренировка
- Запись
- План тренировки
- Занятие
- Зал / Локация

- **Оплата**
- **Отчёт**
- **Услуга**

Перечисления:

- Тип тренировки
- Статус записи
- Тип абонемента
- Способ оплаты

Промпты

A1

Описание предметной области: {описание}

Задание:

«Сгенерируй PlantUML код для диаграммы вариантов использования системы управления фитнес-центром FitPlus. Укажи основных действующих лиц и перечисли основные случаи использования.»

```
@startuml
!theme plain

left to right direction

actor Клиент
actor Тренер
actor Администратор
actor Менеджер
actor "Платёжная система" as PaymentSystem

rectangle "Система управления фитнес-центром FitPlus" {
    usecase "Просмотр расписания занятий" as UC1
    usecase "Поиск тренировок" as UC2
    usecase "Запись на тренировку" as UC3
    usecase "Отмена записи" as UC4
    usecase "Покупка абонемента" as UC5
    usecase "Оплата услуг" as UC6
    usecase "Просмотр истории посещений" as UC7
    usecase "Просмотр информации о тренерах" as UC8
    usecase "Получение рекомендаций" as UC9

    usecase "Просмотр расписания своих занятий" as UC10
    usecase "Отметка посещаемости" as UC11
    usecase "Ведение плана тренировок" as UC12
}
```

```
usecase "Отправка рекомендаций клиентам" as UC13
usecase "Создание отчётов по занятиям" as UC14

usecase "Регистрация клиента" as UC15
usecase "Оформление абонемента" as UC16
usecase "Проверка доступа" as UC17
usecase "Ручная запись на занятие" as UC18
usecase "Подтверждение оплаты наличными" as UC19
usecase "Возвраты и корректировки" as UC20

usecase "Формирование отчётов" as UC21
usecase "Создание и редактирование расписания" as UC22
usecase "Управление тренерами" as UC23
usecase "Управление услугами" as UC24
usecase "Управление акциями и скидками" as UC25
}
```

Клиент --> UC1

Клиент --> UC2

Клиент --> UC3

Клиент --> UC4

Клиент --> UC5

Клиент --> UC6

Клиент --> UC7

Клиент --> UC8

Клиент --> UC9

Тренер --> UC10

Тренер --> UC11

Тренер --> UC12

Тренер --> UC13

Тренер --> UC14

Администратор --> UC15

Администратор --> UC16

Администратор --> UC17

Администратор --> UC18

Администратор --> UC19

Администратор --> UC20

Менеджер --> UC21

Менеджер --> UC22

Менеджер --> UC23

Менеджер --> UC24

Менеджер --> UC25

PaymentSystem --> UC6

UC3 .> UC1 : include

UC4 .> UC1 : include

```
UC5 .> UC6 : include  
UC18 .> UC1 : include  
UC22 .> UC1 : include  
UC23 .> UC10 : include
```

```
@enduml
```

A2

Описание предметной области: {описание}

Задание:

«Сгенерируй PlantUML код для диаграммы вариантов использования системы управления фитнес-центром.

Действующие лица: Клиент, Тренер, Администратор, Менеджер, Платёжная система.

Укажи основные варианты использования: просмотр расписания, запись на занятие, покупка абонемента, отмена записи, регистрация клиента, создание расписания, ведение плана тренировок, формирование отчётов, онлайн-оплата и т.д.»

```
@startuml  
left to right direction  
  
actor Клиент  
actor Тренер  
actor Администратор  
actor Менеджер  
actor "Платёжная система" as PaymentSystem  
  
rectangle "Система управления фитнес-центром" {  
    usecase "Просмотр расписания" as UC1  
    usecase "Поиск тренировок" as UC2  
    usecase "Просмотр информации о тренерах" as UC3  
    usecase "Просмотр истории посещений" as UC4  
    usecase "Запись на тренировку" as UC5  
    usecase "Отмена записи" as UC6  
    usecase "Покупка абонемента" as UC7  
    usecase "Онлайн-оплата" as UC8  
    usecase "Получение рекомендаций" as UC9  
  
    usecase "Просмотр расписания занятий" as UC10  
    usecase "Отметка посещаемости" as UC11  
    usecase "Ведение плана тренировок" as UC12  
    usecase "Создание отчётов по занятиям" as UC13  
    usecase "Отправка рекомендаций" as UC14  
  
    usecase "Регистрация клиента" as UC15}
```

```
usecase "Оформление абонемента" as UC16
usecase "Проверка доступа" as UC17
usecase "Ручная запись на занятие" as UC18
usecase "Подтверждение оплаты наличными" as UC19
usecase "Возвраты и корректировки" as UC20

usecase "Формирование отчётов" as UC21
usecase "Создание и редактирование расписания" as UC22
usecase "Управление тренерами" as UC23
usecase "Управление услугами" as UC24
usecase "Управление акциями и скидками" as UC25
}
```

Клиент --> UC1

Клиент --> UC2

Клиент --> UC3

Клиент --> UC4

Клиент --> UC5

Клиент --> UC6

Клиент --> UC7

Клиент --> UC8

Клиент --> UC9

Тренер --> UC10

Тренер --> UC11

Тренер --> UC12

Тренер --> UC13

Тренер --> UC14

Администратор --> UC15

Администратор --> UC16

Администратор --> UC17

Администратор --> UC18

Администратор --> UC19

Администратор --> UC20

Менеджер --> UC21

Менеджер --> UC22

Менеджер --> UC23

Менеджер --> UC24

Менеджер --> UC25

UC8 --> PaymentSystem

' Связи между use case

UC5 .> UC1 : include

UC6 .> UC1 : include

UC7 .> UC8 : include

UC18 .> UC15 : include

UC16 .> UC15 : include

```
UC17 .> UC16 : include  
UC21 .> UC13 : include  
UC22 .> UC1 : include  
UC23 .> UC10 : include  
UC14 .> UC9 : include
```

```
@enduml
```

A3

Описание предметной области: {описание}

Задание:

«Сгенерируй PlantUML код для диаграммы вариантов использования системы управления фитнес-центром.

Для каждого действующего лица укажи связанные варианты использования.

Для Клиента: поиск занятий, просмотр расписания, запись, отмена записи, покупка абонемента, оплата услуг, просмотр истории посещений.

Для Тренера: просмотр своего расписания, ведение плана тренировок, отметка посещаемости, рекомендации клиентам.

Для Администратора: регистрация клиента, оформление абонемента, проверка доступа, ручная запись, возвраты.

Для Менеджера: создание расписания, управление тренерами, управление услугами, отчёты.

Отобрази границу системы и связи между вариантами использования (include, extend).»

```
@startuml
```

```
left to right direction
```

```
actor Клиент  
actor Тренер  
actor Администратор  
actor Менеджер  
actor "Платёжная система" as PaymentSystem
```

```
rectangle "Система управления фитнес-центром" {  
    usecase "Поиск занятий" as UC1  
    usecase "Просмотр расписания" as UC2  
    usecase "Запись на тренировку" as UC3  
    usecase "Отмена записи" as UC4  
    usecase "Покупка абонемента" as UC5  
    usecase "Оплата услуг" as UC6  
    usecase "Просмотр истории посещений" as UC7  
    usecase "Просмотр информации о тренерах" as UC8  
  
    usecase "Просмотр своего расписания" as UC9
```

```
usecase "Ведение плана тренировок" as UC10
usecase "Отметка посещаемости" as UC11
usecase "Отправка рекомендаций клиентам" as UC12

usecase "Регистрация клиента" as UC13
usecase "Оформление абонемента" as UC14
usecase "Проверка доступа" as UC15
usecase "Ручная запись на занятие" as UC16
usecase "Подтверждение оплаты наличными" as UC17
usecase "Оформление возвратов" as UC18

usecase "Создание и редактирование расписания" as UC19
usecase "Управление тренерами" as UC20
usecase "Управление услугами" as UC21
usecase "Формирование отчетов" as UC22
usecase "Управление акциями и скидками" as UC23

usecase "Обработка онлайн-платежей" as UC24
}
```

Клиент --> UC1

Клиент --> UC2

Клиент --> UC3

Клиент --> UC4

Клиент --> UC5

Клиент --> UC6

Клиент --> UC7

Клиент --> UC8

Тренер --> UC9

Тренер --> UC10

Тренер --> UC11

Тренер --> UC12

Администратор --> UC13

Администратор --> UC14

Администратор --> UC15

Администратор --> UC16

Администратор --> UC17

Администратор --> UC18

Менеджер --> UC19

Менеджер --> UC20

Менеджер --> UC21

Менеджер --> UC22

Менеджер --> UC23

PaymentSystem --> UC24

' Связи include/extend

```
UC3 .> UC2 : include  
UC4 .> UC2 : include  
UC5 .> UC6 : include  
UC16 .> UC2 : include  
UC16 .> UC15 : include  
UC17 .> UC14 : extend  
UC22 .> UC7 : include  
UC22 .> UC14 : include  
UC6 .> UC24 : include
```

```
@enduml
```

A4

Описание предметной области: {описание}

Задание:

«Сгенерируй PlantUML код для диаграммы вариантов использования системы.

Отобрази всех актёров, включая наследование:

Человек → Клиент

Человек → Тренер

Человек → Администратор

Человек → Менеджер

Включи границу системы.

Используй include и extend:

– Запись на занятие → include → Проверка доступного абонемента

– Покупка абонемента → include → Оплата

– Отметка посещаемости → extend → Сообщение клиенту

– Создание расписания → include → Проверка доступности зала

Укажи все связи и корректные пунктирные стрелки.»

```
@startuml  
left to right direction  
  
actor "Платёжная система" as PaymentSystem  
  
package "Фитнес-центр" {  
    usecase "Просмотр расписания" as ViewSchedule  
    usecase "Поиск тренировок" as SearchWorkouts  
    usecase "Просмотр информации о тренерах" as ViewTrainers  
    usecase "Просмотр истории посещений" as ViewHistory  
    usecase "Просмотр доступных тренажёров" as ViewEquipment  
  
    usecase "Запись на тренировку" as BookWorkout  
    usecase "Проверка доступного абонемента" as CheckMembership  
    usecase "Отмена записи" as CancelBooking  
  
    usecase "Покупка абонемента" as BuyMembership}
```

```

usecase "Оплата услуг" as MakePayment

usecase "Получение рекомендаций" as GetRecommendations

usecase "Просмотр расписания занятий" as ViewTrainerSchedule
usecase "Отметка посещаемости" as MarkAttendance
usecase "Сообщение клиенту" as NotifyClient
usecase "Ведение плана тренировок" as ManageWorkoutPlans
usecase "Создание отчётов по занятиям" as CreateSessionReports
usecase "Отправка рекомендаций" as SendRecommendations

usecase "Регистрация клиента" as RegisterClient
usecase "Оформление абонемента" as IssueMembership
usecase "Проверка доступа" as CheckAccess
usecase "Ручная запись на занятие" as ManualBooking
usecase "Подтверждение оплаты наличными" as ConfirmCashPayment
usecase "Возвраты и корректировки" as ProcessRefunds

usecase "Формирование отчётов" as GenerateReports
usecase "Создание расписания" as CreateSchedule
usecase "Проверка доступности зала" as CheckRoomAvailability
usecase "Редактирование расписания" as EditSchedule
usecase "Управление тренерами" as ManageTrainers
usecase "Управление услугами" as ManageServices
usecase "Управление акциями и скидками" as ManagePromotions
}

actor "Человек" as Human
actor "Клиент" as Client
actor "Тренер" as Trainer
actor "Администратор" as Admin
actor "Менеджер" as Manager

Human <|-- Client
Human <|-- Trainer
Human <|-- Admin
Human <|-- Manager

Client --> ViewSchedule
Client --> SearchWorkouts
Client --> ViewTrainers
Client --> ViewHistory
Client --> ViewEquipment
Client --> BookWorkout
Client --> CancelBooking
Client --> BuyMembership
Client --> MakePayment
Client --> GetRecommendations

Trainer --> ViewTrainerSchedule

```

```
Trainer --> MarkAttendance  
Trainer --> ManageWorkoutPlans  
Trainer --> CreateSessionReports  
Trainer --> SendRecommendations
```

```
Admin --> RegisterClient  
Admin --> IssueMembership  
Admin --> CheckAccess  
Admin --> ManualBooking  
Admin --> ConfirmCashPayment  
Admin --> ProcessRefunds
```

```
Manager --> GenerateReports  
Manager --> CreateSchedule  
Manager --> EditSchedule  
Manager --> ManageTrainers  
Manager --> ManageServices  
Manager --> ManagePromotions
```

```
PaymentSystem --> MakePayment
```

```
BookWorkout ..> CheckMembership : include  
BuyMembership ..> MakePayment : include  
MarkAttendance ..> NotifyClient : extend  
CreateSchedule ..> CheckRoomAvailability : include
```

```
@enduml
```

B1

Описание предметной области: {описание}

Задание:

«Сгенерируй PlantUML код для диаграммы классов системы управления фитнес-центром. Укажи основные классы и связи между ними.»

```
@startuml  
!define ENUM enum  
  
skinparam groupInheritance 2  
skinparam nodesep 10  
skinparam ranksep 10  
  
' Перечисления  
ENUM ТипТренировки {  
    ГРУППОВАЯ  
    ПЕРСОНАЛЬНАЯ  
    КАРДИО
```

СИЛОВАЯ
ЙОГА

}

```
ENUM СтатусЗаписи {  
    ЗАПИСАН  
    ОТМЕНЕН  
    ПОСЕЩЕН  
    НЕ_ПОСЕЩЕН  
}
```

```
ENUM ТипАбонемента {  
    РАЗОВЫЙ  
    МЕСЯЧНЫЙ  
    КВАРТАЛЬНЫЙ  
    ГОДОВОЙ  
    БЕЗЛИМИТНЫЙ  
}
```

```
ENUM СтатусАбонемента {  
    АКТИВЕН  
    ПРОСРОЧЕН  
    ЗАБЛОКИРОВАН  
    ЗАВЕРШЕН  
}
```

```
ENUM СпособОплаты {  
    НАЛИЧНЫЕ  
    КАРТА  
    ONLINE  
}
```

```
ENUM СтатусОплаты {  
    ОЖИДАЕТ  
    ПОДТВЕРЖДЕНА  
    ОТМЕНЕНА  
    ВОЗВРАТ  
}
```

' Основные классы

```
class Человек {  
    # id: Long  
    # имя: String  
    # фамилия: String  
    # email: String  
    # телефон: String  
    # датаРегистрации: Date  
}
```

```
class Клиент {
```

```
    - номерКарты: String
    - датаРождения: Date
    - медицинскиеОсобенности: String
+   просмотретьРасписание()
+   записатьсяНаТренировку()
+   отменитьЗапись()
+   купитьАбонемент()
+   оплатитьУслугу()
}
```

```
class Тренер {
    - специализация: String
    - рейтинг: Double
    - опыт: Integer
    - сертификаты: List<String>
+   просмотретьРасписание()
+   отметитьПосещаемость()
+   создатьПланТренировки()
+   отправитьРекомендацию()
}
```

```
class Администратор {
    - номерСотрудника: String
    - смена: String
+   зарегистрироватьКлиента()
+   оформитьАбонемент()
+   проверитьДоступ()
+   записатьНаЗанятие()
+   подтвердитьОплату()
}
```

```
class Менеджер {
    - номерСотрудника: String
    - отдел: String
+   сформироватьОтчет()
+   создатьРасписание()
+   управлятьТренерами()
+   управлятьУслугами()
+   управлятьАкциями()
}
```

```
class Абонемент {
    - id: Long
    - типАбонемента: ТипАбонемента
    - статус: СтатусАбонемента
    - датаНачала: Date
    - датаОкончания: Date
    - количествоПосещений: Integer
    - использованоПосещений: Integer
    - цена: BigDecimal
}
```

```
+ проверитьВалидность(): Boolean  
+ использоватьПосещение()  
}
```

```
class Тренировка {  
    - id: Long  
    - название: String  
    - описание: String  
    - тип: ТипТренировки  
    - продолжительность: Integer  
    - уровеньСложности: String  
    - требования: String  
}
```

```
class Занятие {  
    - id: Long  
    - датаВремя: DateTime  
    - максимальноУчастников: Integer  
    - текущееУчастников: Integer  
    - цена: BigDecimal  
    + проверитьДоступность(): Boolean  
    + добавитьУчастника()  
    + удалитьУчастника()  
}
```

```
class Запись {  
    - id: Long  
    - датаЗаписи: DateTime  
    - статус: СтатусЗаписи  
    - комментарий: String  
    + отменить()  
    + подтвердитьПосещение()  
}
```

```
class ПланТренировки {  
    - id: Long  
    - название: String  
    - описание: String  
    - упражнения: List<String>  
    - рекомендации: String  
    - датаСоздания: Date  
    + обновитьПлан()  
}
```

```
class Зал {  
    - id: Long  
    - название: String  
    - описание: String  
    - вместимость: Integer  
    - оборудование: List<String>
```

```
+ проверитьДоступность(): Boolean
}

class Оплата {
    - id: Long
    - сумма: BigDecimal
    - способОплаты: СпособОплаты
    - статус: СтатусОплаты
    - датаОплаты: DateTime
    - валюта: String
    + подтвердитьОплату()
    + отменитьОплату()
    + выполнитьВозврат()
}

class Отчет {
    - id: Long
    - типОтчета: String
    - периодНачала: Date
    - периодОкончания: Date
    - данные: Map<String, Object>
    - датаСоздания: DateTime
    + сгенерировать()
    + экспорттировать()
}

class Услуга {
    - id: Long
    - название: String
    - описание: String
    - цена: BigDecimal
    - продолжительность: Integer
    - активна: Boolean
    + активировать()
    + деактивировать()
}

class Акция {
    - id: Long
    - название: String
    - описание: String
    - скидка: BigDecimal
    - датаНачала: Date
    - датаОкончания: Date
    - активна: Boolean
    + проверитьАктивность(): Boolean
    + применитьСкидку()
}

' Наследование
```

```

Человек <|-- Клиент
Человек <|-- Тренер
Человек <|-- Администратор
Человек <|-- Менеджер

' Связи
Клиент "1" *-- "0..*" Абонемент
Клиент "1" *-- "0..*" Запись
Клиент "1" *-- "0..*" Оплата

Тренер "1" *-- "0..*" ПланТренировки
Тренер "1" *-- "0..*" Занятие

Абонемент "1" *-- "0..*" Запись

Тренировка "1" *-- "0..*" Занятие
Занятие "1" *-- "1..*" Запись
Занятие "1" *-- "1" Зал

Запись "1" *-- "0..1" ПланТренировки

Менеджер "1" *-- "0..*" Отчет
Менеджер "1" *-- "0..*" Акция

Услуга "1" *-- "0..*" Тренировка
Услуга "1" *-- "0..*" Абонемент

Акция "0..*" -- "0..*" Услуга : применяется к >
Оплата "1" -- "1" Абонемент : оплачивает >
Оплата "1" -- "0..1" Запись : оплачивает >

@enduml

```

B2

Описание предметной области: {описание}
Задание:

«Сгенерируй PlantUML код для диаграммы классов системы.
Добавь классы: Человек, Клиент, Тренер, Администратор, Менеджер,
Тренировка, Запись, Абонемент, Оплата.
Укажи базовые атрибуты и связи между ними.»

```

@startuml
!define ENUM enum

skinparam groupInheritance 2

```

```
ENUM ТипТренировки {  
    ГРУППОВАЯ  
    ПЕРСОНАЛЬНАЯ  
    ЙОГА  
    КАРДИО  
    СИЛОВАЯ  
}
```

```
ENUM СтатусЗаписи {  
    ЗАПИСАН  
    ОЖИДАЕТ_ОПЛАТЫ  
    ОТМЕНЕН  
    ПОСЕЩЕН  
    НЕ_ПОСЕЩЕН  
}
```

```
ENUM ТипАбонемента {  
    РАЗОВЫЙ  
    МЕСЯЧНЫЙ  
    КВАРТАЛЬНЫЙ  
    ГОДОВОЙ  
    ПРОБНЫЙ  
}
```

```
ENUM СпособОплаты {  
    КАРТА  
    НАЛИЧНЫЕ  
    ПЕРЕВОД  
}
```

```
class Человек {  
    - id: Long  
    - имя: String  
    - фамилия: String  
    - email: String  
    - телефон: String  
    - датаРегистрации: Date  
}
```

```
class Клиент {  
    - номерКарты: String  
    - датаРождения: Date  
    - медицинскиеОсобенности: String  
    - getВозраст(): int  
}
```

```
class Тренер {  
    - специализация: String  
    - рейтинг: Double
```

```
- стаж: int
- лицензии: List<String>
- isАктивен: boolean
}

class Администратор {
    - смена: String
    - номерРабочегоМеста: String
}

class Менеджер {
    - отдел: String
    - уровеньДоступа: int
}

class Тренировка {
    - id: Long
    - название: String
    - тип: ТипТренировки
    - описание: String
    - продолжительность: int
    - уровеньСложности: String
    - максимальноеКоличествоУчастников: int
}

class Занятие {
    - id: Long
    - датаВремяНачала: DateTime
    - датаВремяОкончания: DateTime
    - зал: String
    - количествоЗаписей: int
    - isОтменено: boolean
}

class Запись {
    - id: Long
    - датаЗаписи: DateTime
    - статус: СтатусЗаписи
    - примечания: String
    - отменить(): void
    - подтвердить(): void
}

class Абонемент {
    - id: Long
    - тип: ТипАбонемента
    - датаНачала: Date
    - датаОкончания: Date
    - количествоЗанятий: int
    - использованоЗанятий: int
}
```

```
        - цена: BigDecimal  
        - isАктивен(): boolean  
        - оставшеесяЗанятий(): int  
    }
```

```
class Оплата {  
    - id: Long  
    - сумма: BigDecimal  
    - датаОплаты: DateTime  
    - способОплаты: СпособОплаты  
    - статус: String  
    - номерТранзакции: String  
    - подтвердить(): void  
    - вернуть(): void  
}
```

```
class ПланТренировки {  
    - id: Long  
    - название: String  
    - описание: String  
    - упражнения: List<String>  
    - рекомендации: String  
    - создатьОтчет(): String  
}
```

```
class Отчет {  
    - id: Long  
    - тип: String  
    - датаСоздания: Date  
    - данные: Map<String, Object>  
    - сгенерировать(): void  
    - экспортовать(): void  
}
```

```
class Услуга {  
    - id: Long  
    - название: String  
    - описание: String  
    - цена: BigDecimal  
    - продолжительность: int  
    - isАктивна: boolean  
}
```

```
' Наследование  
Человек <|-- Клиент  
Человек <|-- Тренер  
Человек <|-- Администратор  
Человек <|-- Менеджер
```

```
Тренировка <|-- Занятие
```

```

' Связи между классами
Клиент "1" -- "*" Запись
Клиент "1" -- "*" Абонемент
Клиент "1" -- "*" Оплата

Тренер "1" -- "*" Занятие
Тренер "1" -- "*" ПланТренировки

Занятие "1" -- "*" Запись
Занятие "1" -- "1" Тренировка

Абонемент "1" -- "*" Запись

Оплата "1" -- "1" Абонемент

Тренер "1" -- "*" ПланТренировки
ПланТренировки "1" -- "*" Клиент

Менеджер "1" -- "*" Отчет
Менеджер "1" -- "*" Услуга

Администратор "1" -- "*" Оплата

@enduml

```

B3

Описание предметной области: {описание}

Задание:

«Сгенерируй PlantUML для диаграммы классов.

Наследование: Человек → Клиент, Человек → Тренер, Человек → Администратор, Человек → Менеджер.

Добавь классы Тренировка, Занятие, Зал, План тренировки, Запись, Абонемент, Оплата, Отчёт.

Покажи типы связей (ассоциация, агрегация, композиция, ассоциация–класс) и кратности.»

```
@startuml
```

```
!define Abstract abstract
!define Enum enum
```

' Перечисления

```
Enum ТипТренировки {
    + ГРУППОВАЯ
    + ПЕРСОНАЛЬНАЯ
```

```
+ ЙОГА
+ СИЛОВАЯ
+ КАРДИО
}
```

```
Enum СтатусЗаписи {
    + ЗАПИСАН
    + ПРИШЕЛ
    + ОТМЕНЕН
    + ПРОПУЩЕН
}
```

```
Enum ТипАбонемента {
    + РАЗОВЫЙ
    + МЕСЯЧНЫЙ
    + КВАРТАЛЬНЫЙ
    + ГОДОВОЙ
}
```

```
Enum СпособОплаты {
    + НАЛИЧНЫЕ
    + КАРТА
    + ОНЛАЙН
}
```

```
Enum СтатусОплаты {
    + ОЖИДАЕТ
    + ОПЛАЧЕНО
    + ОШИБКА
    + ВОЗВРАТ
}
```

```
' Основные классы
Abstract Человек {
    # id: Long
    # имя: String
    # фамилия: String
    # email: String
    # телефон: String
    # датаРегистрации: Date
}
```

```
class Клиент {
    - датаРождения: Date
    - медицинскиеОсобенности: String
    + просмотретьРасписание()
    + записатьсяНаТренировку()
    + отменитьЗапись()
    + купитьАбонемент()
}
```

```
class Тренер {  
    - специализация: String  
    - рейтинг: Double  
    - опыт: Integer  
    + просмотретьРасписание()  
    + отметитьПосещаемость()  
    + создатьПланТренировки()  
    + отправитьРекомендацию()  
}
```

```
class Администратор {  
    - смена: String  
    + зарегистрироватьКлиента()  
    + оформитьАбонемент()  
    + проверитьДоступ()  
    + подтвердитьОплату()  
}
```

```
class Менеджер {  
    - отдел: String  
    + сгенерироватьОтчет()  
    + создатьРасписание()  
    + управлятьТренерами()  
    + управлятьАкциями()  
}
```

```
class Абонемент {  
    - id: Long  
    - тип: ТипАбонемента  
    - датаНачала: Date  
    - датаОкончания: Date  
    - количествоПосещений: Integer  
    - цена: BigDecimal  
    - активен: Boolean  
    + проверитьВалидность(): Boolean  
    + использоватьПосещение()  
}
```

```
class Тренировка {  
    - id: Long  
    - название: String  
    - описание: String  
    - тип: ТипТренировки  
    - длительность: Integer  
    - сложность: String  
}
```

```
class Занятие {  
    - id: Long
```

```
    - датаВремя: DateTime
    - максимальноУчастников: Integer
    - текущееКоличествоУчастников: Integer
    - отменено: Boolean
    + проверитьДоступность(): Boolean
    + отменить()
}
```

```
class Зал {
    - id: Long
    - название: String
    - описание: String
    - вместимость: Integer
    - оборудование: String
}
```

```
class ПланТренировки {
    - id: Long
    - название: String
    - описание: String
    - упражнения: String
    - рекомендации: String
    - датаСоздания: Date
}
```

```
class Запись {
    - id: Long
    - датаЗаписи: DateTime
    - статус: СтатусЗаписи
    - комментарий: String
    + отменить()
    + подтвердитьПосещение()
}
```

```
class Оплата {
    - id: Long
    - сумма: BigDecimal
    - датаОплаты: DateTime
    - способОплаты: СпособОплаты
    - статус: СтатусОплаты
    - внешнийId: String
    + обработатьОплату()
    + выполнитьВозврат()
}
```

```
Abstract Отчет {
    # id: Long
    # название: String
    # датаСоздания: Date
    # данные: String
```

```

+ сгенерировать()
}

class ОтчетПоПосещаемости {
    - периодНачала: Date
    - периодОкончания: Date
    - общееКоличествоПосещений: Integer
    - популярныеТренировки: List<String>
}

class ОтчетПоПродажам {
    - периодНачала: Date
    - периодОкончания: Date
    - общаяВыручка: BigDecimal
    - количествоПродаж: Integer
}

class ОтчетПоТренерам {
    - периодНачала: Date
    - периодОкончания: Date
    - рейтингиТренеров: Map<String, Double>
    - количествоЗанятий: Map<String, Integer>
}

class Услуга {
    - id: Long
    - название: String
    - описание: String
    - цена: BigDecimal
    - активна: Boolean
    + активировать()
    + деактивировать()
}

' Наследование
Человек <|-- Клиент
Человек <|-- Тренер
Человек <|-- Администратор
Человек <|-- Менеджер

Отчет <|-- ОтчетПоПосещаемости
Отчет <|-- ОтчетПоПродажам
Отчет <|-- ОтчетПоТренерам

' Связи
Клиент "1" -- "*" Абонемент : имеет >
Клиент "*" -- "*" Тренировка : записывается на >
Клиент "1" -- "*" Запись : имеет >

Тренер "1" -- "*" Тренировка : проводит >

```

```

Тренер "1" -- "*" ПланТренировки : создает >
Тренер "*" -- "*" Клиент : тренирует >

Абонемент "*" -- "1" Тренировка : дает доступ к >

Тренировка "1" -- "*" Занятие : включает >
Занятие "*" -- "1" Зал : проводится в >

Запись "1" -- "1" Занятие : на >
Запись "1" -- "1" Клиент : от >

(Клиент, Тренировка) .. Запись : ассоциация-класс

Оплата "1" -- "1" Абонемент : для >
Оплата "1" -- "1" Клиент : от >

Менеджер "1" -- "*" Отчет : создает >
Менеджер "1" -- "*" Тренер : управляет >

Администратор "1" -- "*" Оплата : подтверждает >

ПланТренировки "1" -- "*" Клиент : назначен >
ПланТренировки "1" -- "1" Тренер : создан >

@enduml

```

B4

Описание предметной области: {описание}

Задание:

«Сгенеририуй PlantUML для продвинутой диаграммы классов системы.
Добавь следующие классы с атрибутами и типами:

Человек: ФИО:string, ДатаРождения:date, Телефон:string
 Клиент: ДатаРегистрации:date, СтатусАбонемента:boolean
 Тренер: Специализация:string, Стаж:int
 Администратор: РабочаяСмена:string
 Менеджер: УровеньДоступа:int
 Абонемент: Тип:string, ДатаПокупки:date, СрокДействия:int
 Тренировка: Тип:enum, Длительность:int
 Занятие: Дата:date, Место:string
 ПланТренировки: Цель:string, Описание:string
 Запись: Статус:enum, ДатаСоздания:date
 Оплата: Сумма:int, Дата:date, Метод:enum
 Отчёт: Тип:string, ДатаСоздания:date
 Зал: Название:string, Вместимость:int

Добавь enum для статусов записи, типов тренировок и способов оплаты.

Добавь композиции (Например: Тренировка *-- ПланТренировки), агрегации, наследование и ассоциации с кратностями.»

```
@startuml
!define MAIN_ENTITY class
hide circle
skinparam classAttributeIconSize 0

' Перечисления
enum "СтатусЗаписи" {
    ACTIVE = "Активна"
    CANCELLED = "Отменена"
    COMPLETED = "Завершена"
    PENDING = "Ожидание"
}

enum "ТипТренировки" {
    PERSONAL = "Персональная"
    GROUP = "Групповая"
    YOGA = "Йога"
    CARDIO = "Кардио"
    STRENGTH = "Силовая"
    DANCE = "Танцы"
}

enum "СпособОплаты" {
    CASH = "Наличные"
    CARD = "Карта"
    ONLINE = "Онлайн"
    TRANSFER = "Перевод"
}

enum "ТипАбонемента" {
    SINGLE = "Разовый"
    MONTHLY = "Месячный"
    QUARTERLY = "Квартальный"
    YEARLY = "Годовой"
    UNLIMITED = "Безлимитный"
}

' Основные классы
class Человек {
    ФИО: string
    ДатаРождения: date
    Телефон: string
}

class Клиент {
    ДатаРегистрации: date
```

```
    СтатусАбонемента: boolean
}

class Тренер {
    Специализация: string
    Стаж: int
}

class Администратор {
    РабочаяСмена: string
}

class Менеджер {
    УровеньДоступа: int
}

class Абонемент {
    Тип: ТипАбонемента
    ДатаПокупки: date
    СрокДействия: int
    Активен: boolean
}

class Тренировка {
    Тип: ТипТренировки
    Длительность: int
    Сложность: string
    Описание: string
}

class Занятие {
    Дата: date
    ВремяНачала: time
    ВремяОкончания: time
    Статус: string
}

class Запись {
    Статус: СтатусЗаписи
    ДатаСоздания: date
    ДатаОтмены: date
}

class ПланТренировки {
    Цель: string
    Описание: string
    УровеньСложности: string
    Рекомендации: string
}
```

```
class Оплата {
    Сумма: decimal
    Дата: date
    Метод: СпособОплаты
    Статус: string
}

class Отчёт {
    Тип: string
    ДатаСоздания: date
    Период: string
    Данные: string
}

class Зал {
    Название: string
    Вместимость: int
    Оснащение: string
    Этаж: int
}

class Услуга {
    Название: string
    Описание: string
    Стоимость: decimal
    Длительность: int
    Активна: boolean
}

' Наследование
Человек <|-- Клиент
Человек <|-- Тренер
Человек <|-- Администратор
Человек <|-- Менеджер

' Композиции
Тренировка *--- ПланТренировки
Занятие *--- Запись

' Агрегации
Абонемент o-- Клиент
Тренировка o-- Тренер
Занятие o-- Зал
Услуга o-- Тренировка

' Ассоциации с кратностями
Клиент "1" -- "*" Запись : имеет >
Запись "1" -- "1" Занятие : на >
Занятие "1" -- "1" Тренер : проводит >
Занятие "1" -- "1" Тренировка : относится к >
```

Клиент "1" -- "*" Абонемент : покупает >
Клиент "1" -- "*" Оплата : совершает >
Администратор "1" -- "*" Оплата : обрабатывает >
Менеджер "1" -- "*" Отчёт : формирует >
Тренер "1" -- "*" ПланТренировки : создает >
Клиент "1" -- "*" ПланТренировки : назначается >
Занятие "1" -- "1" Зал : проходит в >
Менеджер "1" -- "*" Расписание : управляет >
Тренер "1" -- "*" Занятие : назначен на >

' Дополнительные ассоциации

```
class Расписание {  
    Период: string  
    Активно: boolean  
}
```

Расписание "1" -- "*" Занятие : содержит >
Услуга "1" -- "*" Тренировка : включает >
Менеджер "1" -- "*" Услуга : управляет >
Администратор "1" -- "*" Клиент : регистрирует >

' Note для пояснений

```
note top of Абонемент  
СрокДействия – в днях  
Активен – true если абонемент действует  
end note
```

```
note right of Запись  
ДатаОтмены – заполняется при отмене записи  
Статус – из перечисления СтатусЗаписи  
end note
```

```
note left of Оплата  
Сумма – в валюте системы  
Статус: SUCCESS/FAILED/PENDING  
end note
```

@enduml

Анализ

Группа А

A1

- **Диаграмма вариантов использования**

- Четко обозначены основные действующие лица и их взаимосвязи.
- Все сценарии использования соответствуют требованиям задания.

- Границы системы четко выделены.
- **Комментарии**
 - Код PlantUML выполнен правильно, но отсутствует подробное описание атрибутов и методов классов.
 - Рекомендуется дополнить детализацией атрибутов и методов для полного соответствия заданию.

A2

- **Диаграмма вариантов использования**
 - Аналогично А1, но меньше внимания уделено отношениям между вариантами использования.
 - Ключевые моменты отражены верно, но нет включения специфичных случаев использования, указанных в задании.
- **Комментарии**
 - Необходимо включить дополнительные случаи использования, указанные в задании.

A3

- **Диаграмма вариантов использования**
 - Максимально соответствует заданию по количеству и качеству отраженных случаев использования.
 - Правильное использование `include` и `extend`.
 - Наличие комментариев и разъяснений, облегчающих понимание диаграммы.
- **Комментарии**
 - Оптимальное решение для выполнения задания.

A4

- **Диаграмма вариантов использования**
 - Наиболее полная и детализированная версия.
 - Включает все требуемые случаи использования и взаимоотношения между ними.
 - Идеально подходит для демонстрации сложных взаимоотношений между компонентами системы.
- **Комментарии**
 - Преимущественное решение для демонстрации полной функциональности системы.

Группа В

B1

- **Диаграмма классов**
 - Полноценная реализация классов с указанием атрибутов и методов.
 - Правильно реализованы перечисления для улучшения качества кода.
 - Представлены все необходимые связи между классами.
- **Комментарии**
 - Подходит для детального представления внутренней структуры системы.

B2

- **Диаграмма классов**
 - Реализация базового набора классов с минимальными атрибутами и методами.
 - Достаточное количество связей представлено, но без излишней детализации.
- **Комментарии**
 - Лучше всего подходит для начального этапа проектирования.

B3

- **Диаграмма классов**
 - Глубокая детализация атрибутов и методов.
 - Широко используются наследование и полиморфизм.
 - Предусмотрено большое количество связей и зависимостей.
- **Комментарии**
 - Это оптимальное решение для понимания сложной структуры системы.

B4

- **Диаграмма классов**
 - Максимальная детализация классов и их атрибутов.
 - Проработана каждая деталь вплоть до отдельных полей.
- **Комментарии**
 - Самое подходящее решение для финального варианта проектной документации.

Итоговый выбор лучшего промпта

Лучший промпт из группы А: А4

- Причина: наиболее полно отражает функциональность системы, включая все требуемые компоненты и взаимосвязи.

Лучший промпт из группы В: В4

- Причина: максимальная детализация и точность описания внутренних компонентов системы.