ИНТЕРФЕЙСЫ

1. `iPurchaseService`:  
   - Методы:  
     - `void orderTickets(int clientID, List<Ticket> tickets)` - заказывает билеты для указанного клиента  
     - `void returnTickets(int clientID, List<Ticket> tickets)` - возвращает билеты указанного клиента  
  
2. `iCustomerCart`:  
   - Методы:  
     - `void addToCart(Ticket ticket)` - добавляет билет в корзину клиента  
     - `void removeFromCart(Ticket ticket)` - удаляет билет из корзины клиента  
     - `void clearCart()` - очищает корзину клиента  
     - `ArrayList<Ticket> getCart()` - возвращает список билетов из корзины клиента  
  
3. `iSearchTickets`:  
   - Методы:  
     - `ArrayList<Ticket> searchTickets(SearchData searchData)` - ищет билеты на основе указанных данных поиска  
  
4. `iTicketsReserve`:  
   - Методы:  
     - `void reserveTickets(int ticketID)` - резервирует указанный билет  
     - `void unreserveTickets(int ticketID)` - снимает резервацию с указанного билета  
  
5. `iPaymentService`:  
   - Методы:  
     - `int makePayment(int customerID, double amount)` - производит оплату указанной суммы для указанного клиента  
     - `int cancelPayment(int customerID, int paymentID)` - отменяет оплату по указанному идентификатору платежа для указанного клиента

КЛАССЫ

Класс UserInterface представляет собой пользовательский интерфейс для взаимодействия с клиентом.  
  
Поля:  
- private PurchaseService purchaseService: поле для работы с сервисом покупки билетов.  
- private TicketsService ticketsService: поле для работы с сервисом билетов.  
- private Customer customer: поле для хранения информации о клиенте.  
- private ArrayList<Ticket> customerCart: поле для хранения корзины клиента.  
- private String login: поле для хранения логина клиента.  
- private String password: поле для хранения пароля клиента.  
  
Зависимости:  
- Класс UserInterface зависит от класса PurchaseService для работы с сервисом покупки билетов.  
- Класс UserInterface зависит от класса TicketsService для работы с сервисом билетов.  
- Класс UserInterface создает экземпляр класса CustomerService для получения информации о клиенте.  
  
Класс UserInterface предоставляет пользователю функциональности покупки, возврата и поиска билетов, а также взаимодействия с корзиной клиента. Он хранит экземпляры PurchaseService и TicketsService для работы с соответствующими сервисами. При создании экземпляра класса UserInterface необходимо передать логин и пароль клиента из внешнего источника. Затем UserInterface использует эти данные для создания экземпляра класса Customer с помощью класса CustomerService, чтобы получить информацию о клиенте.

1. `PurchaseService`:  
   - Интерфейсы: iPurchaseService, iPaymentService, iTicketReserve  
   - Поля:

     - transactions (ArrayList<Transaction>): список транзакций, для сохранения данных о транзакциях  
   - Методы:  
     - orderTickets(Customer customer, ArrayList<Ticket> tickets): void - заказывает билеты для указанного клиента (iPurchaseService)  
     - returnTickets(Customer customer, ArrayList<Ticket> tickets): void - возвращает билеты указанного клиента (iPurchaseService)

- void addTransaction(Transaction) – добавляет транзакцию в список

- ArrayList<Transaction> findTransaction(SearchData) – возвращает список транзакций подходящих под условия поиска  
 - `void reserveTickets(int ticketID)` - резервирует указанный билет

- `void unreserveTickets(int ticketID)` - снимает резервацию с указанного билета  
     - makePayment(int customerID, double amount): int - производит оплату указанной суммы для указанного клиента (iPaymentService)  
     - cancelPayment(int customerID, int paymentID): int - отменяет оплату по указанному идентификатору платежа для указанного клиента   
- isPaymentCorrect(paymentId): paymentId – проверка выполнения платежа для дальнейшей обработки логики. Возвращает true, если платеж успешен, иначе - false  
  
  
2. `TicketsService`:  
   - Интерфейсы: iSearchTickets, iCustomerCart  
   - Поля:  
     - ArrayList<Ticket> customerCart: поле для хранения корзины клиента  
     - searchResult (ArrayList<Ticket>): список результатов поиска, для сохранения результатов поиска  
   - Методы:  
     - searchTickets(SearchData searchData): List<Ticket> - ищет билеты на основе указанных данных поиска (iSearchTickets)  
     - addToCart(int ticketID): void - добавляет билет в корзину клиента (iCustomerCart)  
     - removeFromCart(int ticketID): void - удаляет билет из корзины клиента (iCustomerCart)  
     - clearCart(): void - очищает корзину клиента (iCustomerCart)  
     - getCart(): ArrayList<Ticket> - возвращает список билетов из корзины клиента (iCustomerCart)

- TicketData getTicketData(Ticket) – получение дополнительных сведений о билете, не участвующих в логике приложения: номер места, класс обслуживания, и т.п.  
  
3. `CustomerService`:  
   - Интерфейс: нет интерфейса  
   - Поля:   
     нет  
   - Методы:  
     - getCustomer(String login, String password): Customer - метод для получения информации о клиенте по заданным логину и паролю  
  
4. `Ticket`:  
   - Интерфейсы: нет интерфейсов  
   - Поля:

- id (int): идентификатор билета из базы данных  
     - price (double): цена билета  
     - ticketData (TicketData): объект содержащий данные о билете  
   - Методы:  
     - getPrice(): double - возвращает цену билета  
     - getTicketData(): TicketData - возвращает данные о билете  
     - toString(): String - переопределенный метод для возвращения строкового представления объекта  
  
5. `Customer`:  
   - Интерфейсы: нет интерфейсов  
   - Поля:  
     - customerId (int): идентификатор клиента  
     - tickets (ArrayList<Ticket>): список билетов клиента  
     - customerData (CustomerData): идентификатор клиента  
   - Методы:  
     - getCustomerId(): int - возвращает идентификатор клиента  
     - getTicket(SearchData searchData): List<Ticket> - возвращает список билетов на основе указанных данных поиска  
     - removeTicket(int index): void - удаляет билет из списка по заданному индексу  
     - addTicket(Ticket ticket): void - добавляет билет в список  
- CustomerData getCustomerData () – получение дополнительных сведений о клиенте, не участвующих в логике приложения: номер ФИО, возраст и т.п.

6. `TicketProvider`:  
   - Интерфейс: iSearchTickets, iTicketsReserve  
   - Поля:  
     - searchResult (ArrayList<Ticket>): результат поиска  
   - Методы:  
      - `ArrayList<Ticket> searchTickets(SearchData searchData)` - ищет билеты на основе указанных данных поиска

- `void reserveTickets(int ticketID)` - резервирует указанный билет

- `void unreserveTickets(int ticketID)` - снимает резервацию с указанного билета

7. `PaymentService`:  
   - Интерфейс: нет интерфейса  
   - Поля: отсутствуют  
   - Методы:  
     - makePayment(customerId, payAmount): paymentId - метод для обработки платежа на заданную сумму, возвращает идентификатор операции от удаленного сервиса

     - cancelPayment(customerId, paymentId): paymentId - метод для отмены платежа, возвращает идентификатор операции от удаленного сервиса

- isPaymentCorrect(paymentId): paymentId – проверка выполнения платежа для дальнейшей обработки логики. Возвращает true, если платеж успешен, иначе – false

определение типа операции ("покупка", "отмена" или "ошибка") основывается на условии, что customerId, начинающийся с 1, будет связан с покупкой, и paymentId, начинающийся с 2, будет связан с отменой. Ошибочные типы операций образуются добавлением нуля в конец числа.

8. `Transaction`:  
   - Интерфейс: нет интерфейса  
   - Поля:  
     - paymentId (int): идентификатор операции  
  - customerId (Customer): клиент, связанный с транзакцией

     - ticketId (int): билет, связанный с транзакцией  
     - date (DateTime): временная метка транзакции  
     - price (double): сумма транзакции  
     - operationType (enum): тип транзакции (покупка, возврат)

   - Методы:

     - toString(): String - переопределенный метод для возвращения строкового представления объекта