Ticket.py

from abc import ABC, abstractmethod

from exceptions import TicketAlreadyBookedError

class Ticket(ABC):

    def \_\_init\_\_(self, price, event):

        self.price = price

        self.event = event

        self.booking\_date = None

    @abstractmethod

    def get\_ticket\_info(self):

        pass

    def book\_ticket(self): #бронь

        if self.booking\_date is not None:

            raise TicketAlreadyBookedError("Билет уже забронирован.")

        self.booking\_date = "Текущая дата"

    def cancel\_booking(self): # отмена брони

        self.booking\_date = None

class RegularTicket(Ticket):

    def get\_ticket\_info(self):

        return f"Обычный билет на событие: {self.event.name}, Цена: {self.price} руб."

class VIPTicket(Ticket):

    def \_\_init\_\_(self, price, event, privileges):

        super().\_\_init\_\_(price, event)

        self.privileges = privileges

    def get\_ticket\_info(self):

        return f"VIP билет на событие: {self.event.name}, Цена: {self.price} руб., Привилегии: {self.privileges}"

Booking.py

class Booking:

    def \_\_init\_\_(self, tickets):

        self.tickets = tickets

        self.booking\_date = "Текущая дата"

        self.status = "Подтверждено"

    def cancel\_booking(self): #отмена брони билета

        self.status = "Отменено"

        for ticket in self.tickets:

            ticket.cancel\_booking()

Ticket\_manager.py

from ticket import RegularTicket, VIPTicket

from booking import Booking

from exceptions import TicketAlreadyBookedError

class TicketManager:

    def \_\_init\_\_(self):

        self.events = []

        self.bookings = []

    def add\_event(self, event): #добавляет событие

        self.events.append(event)

    def create\_booking(self, tickets): #создает бронирование для билетов

        booking = Booking(tickets)

        self.bookings.append(booking)

    def get\_all\_events(self):

        return [event.get\_event\_info() for event in self.events]

exceptions.py

class TicketAlreadyBookedError(Exception):

    pass

class EventNotFoundError(Exception):

    pass

storage.py

import json

def save\_to\_json(filename, data):

    with open(filename, 'w', encoding='utf-8') as f:

        json.dump(data, f, ensure\_ascii=False, indent=4)

def load\_from\_json(filename):

    with open(filename, 'r', encoding='utf-8') as f:

        return json.load(f)

event.py

class Event:

    def \_\_init\_\_(self, name, date, location, description):

        self.name = name

        self.date = date

        self.location = location

        self.description = description

    def get\_event\_info(self):

        return f"Событие: {self.name}\nДата: {self.date}\nМесто: {self.location}\nОписание: {self.description}"

main.py

from ticket import RegularTicket, VIPTicket

from ticket\_manager import TicketManager

from event import Event

from exceptions import TicketAlreadyBookedError

from booking import Booking

import storage

def main():

    ticket\_manager = TicketManager()

    event1 = Event(name="Концерт Rammstein", date="30.07.2019", location="Москва", description="имба")

    event2 = Event(name="Нашествие", date="10.08.2002", location="Раменское", description="база")

    ticket\_manager.add\_event(event1)

    ticket\_manager.add\_event(event2)

    events\_data = ticket\_manager.get\_all\_events()

    storage.save\_to\_json("events.json", events\_data)

    print("Список всех событий:")

    for event\_info in ticket\_manager.get\_all\_events():

        print(event\_info)

    # Создание билетов

    regular\_ticket1 = RegularTicket(price=100000, event=event1)

    vip\_ticket1 = VIPTicket(price=3000000000, event=event1, privileges="Доступ в VIP-зону")

    print("\nИнформация о билетах:")

    print(regular\_ticket1.get\_ticket\_info())

    print(vip\_ticket1.get\_ticket\_info())

    # Создание бронирования

    try:

        regular\_ticket1.book\_ticket()

        print(f"\nБронирование для билета на {regular\_ticket1.event.name} успешно создано.")

        vip\_ticket1.book\_ticket()

        print(f"Бронирование для VIP билета на {vip\_ticket1.event.name} успешно создано.")

        print("\nПопытка повторного бронирования обычного билета:")

        regular\_ticket1.book\_ticket()

    except TicketAlreadyBookedError as e:

        print(f"Ошибка при бронировании билета: {e}")

    # Создание бронирования для нескольких билетов

    booking = Booking([regular\_ticket1, vip\_ticket1])

    ticket\_manager.create\_booking([regular\_ticket1, vip\_ticket1])

    print("\nБронирование для нескольких билетов успешно создано.")

    # Отмена бронирования

    print("\nОтмена бронирования для всех билетов:")

    booking.cancel\_booking()

    print(f"Статус бронирования: {booking.status}")

    print(f"Статус обычного билета: {regular\_ticket1.booking\_date}")

    print(f"Статус VIP билета: {vip\_ticket1.booking\_date}")

    for event\_info in ticket\_manager.get\_all\_events():

        print("/////////////////")

        print(event\_info)