Wyższa Szkoła Bankowa

Programowanie Obiektowe

Ćwiczenia 4 - zadania

Imiona I nazwiska: Artem Pushkarov, Maksym Cherniakov
Adresy mail: furyX66@gmail.com, mchernyako2003@gmail.com
Id's: 144819, 144806
Link na git:
https://github.com/furyX66/ProgramowanieObiektowe_Zadanie4_2023

Zadanie Nr 4

Tytuł projektu: Prosty system rezerwacji biletów na kino

Opis:

Twoim zadaniem jest stworzenie prostej konsolowej aplikacji do rezerwacji miejsc na seanse filmowe w kinie. Aplikacja powinna pozwalać na wybór filmu, wybór godziny seansu, wybór miejsca oraz potwierdzenie rezerwacji

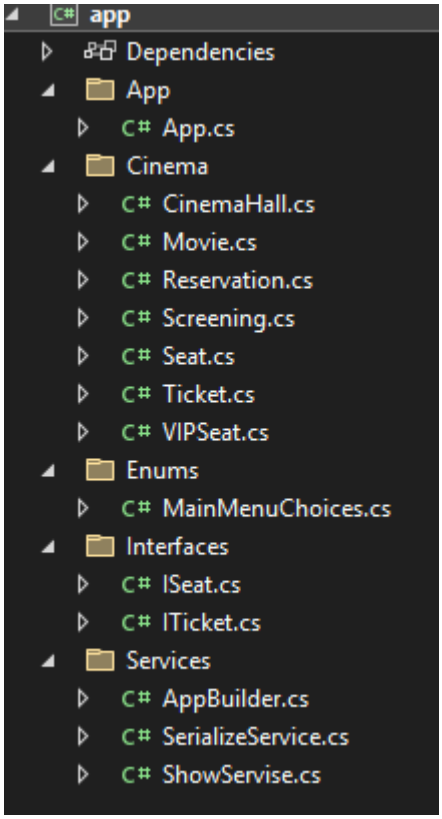
Szczegółowe wymagania:

1. Utwórz klasę `Movie` z właściwościami: `Title`, `Duration`, `AgeRestriction`.
2. Utwórz klasę `Screening` z właściwościami: `Movie`, `DateTime` oraz metodą do wyświetlania dostępnych miejsc.
3. Utwórz klasę `Seat` z właściwościami: `Row`, `Number`, `IsAvailable`.
4. Utwórz klasę `Reservation` z właściwościami: `Screening`, `Seat`, `CustomerName`.
5. Zastosuj zasadę enkapsulacji - pola klasy powinny być prywatne, a dostęp do nich powinien być zapewniony poprzez publiczne metody get i set.
6. Zastosuj zasadę dziedziczenia - utwórz klasę `VIPSeat`, która dziedziczy po klasie `Seat` i dodaje nową właściwość `PriceMultiplier`.

Podpowiedzi:

1. Metoda `DisplayAvailableSeats` w klasie `Screening` powinna przechodzić przez listę miejsc i wyświetlać tylko te, które są dostępne.
2. Po utworzeniu obiektu klasy `Reservation`, miejsce, na które została złożona rezerwacja, powinno zmienić status `IsAvailable` na `false`.

Struktura programu



Klasa Movie

```
public class Movie
{
    #region Public Members

    public string? Title { get; set; }

    public int Duration { get; set; }

    public int AgeRestriction { get; set; }
    #endregion
    #region Public Methods

    public void Show() //Method to show information about movies
    {
        Console.WriteLine($"Title: {Title}");
        Console.WriteLine($"Duration: {Duration} minutes");
        Console.WriteLine($"Age Restriction: {AgeRestriction}+");
    }
    #endregion
}
```

Klasa Reservation

```
4 references
public class Reservation
{
    #region Ctor
    1 reference
    public Reservation(Seat seat, string customerName, CinemaHall cinemaHall)
    {
        Seat = seat;
        CustomerName = customerName;
        CinemaHall = cinemaHall;
    }
    #endregion
    #region Public members
    4 references
    public CinemaHall CinemaHall { get; set; }
    11 references
    public Seat Seat { get; set; }
    3 references
    public string? CustomerName { get; set; }
    0 references
    public int TicketNumber { get; set; }
    #endregion
    #region Public methods
    1 reference
    public void ReserveSeat() //Checks json file "CinemaHalls" and updates it when user chooses seat...
    1 reference
    public static List<Ticket>? GetTicketsFromFile(string fileName) // Deserializes ticket from file...
    1 reference
    public void AddToTicketsFile() //Adds reserved seat to json file...
    1 reference
    private int GenerateRandomTicketNumber() //Generates ticket number...
    #endregion
}
```

Klasa Screening

```
public class Screening
{
    #region Public Members
    0 references
    public string Movie { get; set; }
    0 references
    public List<DateTime> DateTimeList { get; set; }
    #endregion
}
```

Klasa Ticket i interfejs, który implementuje

```
public interface ITicket
{
    #region Public members

    public string Movie { get; set; }

    public DateTime ScreeningTime { get; set; }

    public string CustomerName { get; set; }

    public int Row { get; set; }
    3 references
    public int Number { get; set; }
    3 references
    public int TicketNumber { get; set; }
    #endregion
}
```

```
public class Ticket : ITicket
{
    #region Public members

    public string? Movie { get; set; }

    public DateTime ScreeningTime { get; set; }

    public string? CustomerName { get; set; }

    public int Row { get; set; }

    public int Number { get; set; }
    3 references
    public int TicketNumber { get; set; }
    #endregion
}
```

Klasa Seat i interfejs, który implementuje

```
4 references
public class Seat : ISeat
{
    #region Public members
    10 references
    public int Row { get; set; }
    8 references
    public int Number { get; set; }
    6 references
    public bool IsAvailable { get; set; }
    #endregion
}
```

```
1 reference
public interface ISeat
{
    #region Public members
    10 references
    public int Row { get; set; }
    8 references
    public int Number { get; set; }
    6 references
    public bool IsAvailable { get; set; }
    #endregion
}
```

Klasa VIPSeat

```
public class VIPSeat : Seat
{
    #region Public members
    0 references
    public float PriceMultiplilier { get; set; }
    #endregion
}
```

Klasa CinemaHall

```
9 references
public class CinemaHall //Describes cinema hall for every avaliable movie
{
    #region Public members
    3 references
    public int HallNumber { get; set; }
    3 references
    public List<Seat>? SeatList { get; set; }
    3 references
    public DateTime ScreeningTime { get; set; }
    4 references
    public string? Movie { get; set; }
    #endregion
}
```

Menu do wyboru menu głównego

```
5 references
public enum MainMenuChoices
{
    Exit = 0,
    BookTicket = 1,
    ShowResevedTickets = 2
}
```

Klasa SerializeService

```
4 references
public static class SerializeService
{
    #region Public methods
    3 references
    public static List<T> DeserializeFromFile<T>(string filePath) //Deserializes json file
    {
        string jsonContent = File.ReadAllText(filePath);
        List<T> itemList = JsonConvert.DeserializeObject<List<T>>(jsonContent);
        return itemList;
    }
    1 reference
    public static void SerializeToFile<T>(string fileName, T data)  //Serializes to json file
    {
        string json = JsonConvert.SerializeObject(data, Formatting.Indented);
        File.WriteAllText(fileName, json);
    }

    #endregion
}
```

Klasa ShowService

```
3 references
public class ShowServise
{
    1 reference
    public void ShowMoviesList(List<Movie> movies) //Shows movie list...
    1 reference
    public void MenuShow() //Shows main menu...
    1 reference
    public void BookTicket(ShowServise showServise, List<CinemaHall> cinemaHalls, List<Movie> movies)... //Suggests to choose movie
    1 reference
    public void ScreeningTimeChoice(List<CinemaHall> cinemaHalls, List<DateTime> screeningTimes, string selectedMovie)...//Suggests to choose screening time
    1 reference
    public void SeatChoice(CinemaHall selectedHall)...//Suggests to choose Seat
    1 reference
    public void TicketShowByName()... //Shows ticket information
}
```

Klasa AppBuilder

```
2 references
public class AppBuilder //Service that builds app and all nesssesary objects
{
    public ShowServise showService = new ShowServise();
    public List<Movie> movies = SerializeService.DeserializeFromFile<Movie>("Movies.json");
    public List<CinemaHall> cinemaHalls = SerializeService.DeserializeFromFile<CinemaHall>("CinemaHalls.json");
    1 reference
    public void Menu() //Shows Main menu...
}
```

Klasa App

```
0 references
public class App
{
    private static AppBuilder _appBuilder = new AppBuilder();
    0 references
    static void Main(string[] args)
    {
        _appBuilder.Menu();
    }
}
```

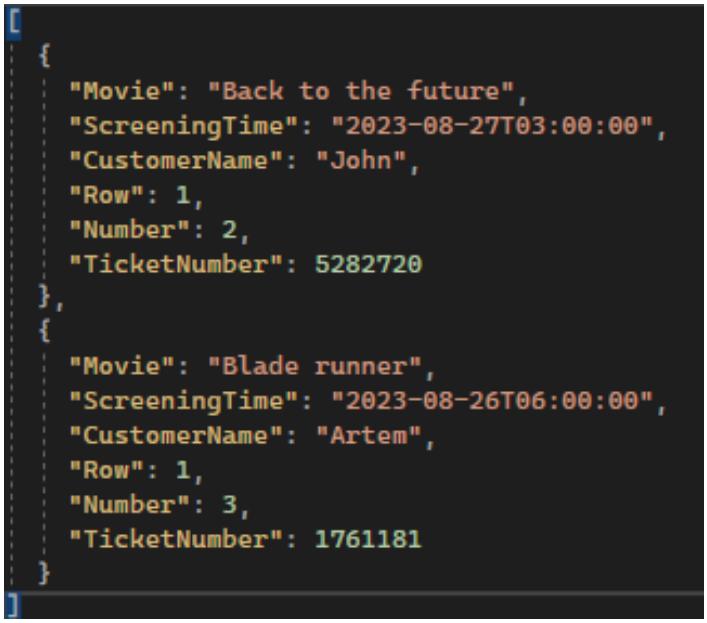
Przykład zamawiania biletu

```
------------------2------------------
Title: Blade runner
Duration: 117 minutes
Age Restriction: 17+
------------------3------------------
Title: Ladybug & Cat Noir, the Movie
Duration: 102 minutes
Age Restriction: 3+

Plese choose movie you want to see (0-exit): 2
Your choice: Avaliable screenings for 'Blade runner':
------------------1------------------
08/27 15:45
------------------2------------------
08/26 6:00
Select a screening by entering its number (0-exit): 2
Selected screening: 08-26 06:00
Available Seats in Hall 4:
Row 1: seat 1; seat 2; seat 3
Row 2: seat 1; seat 2; seat 3
Row 3: seat 1; seat 3
Row 4: seat 2
Row 5: seat 2; seat 3
Enter 0 to exit.
Enter the row number: 1
Enter the seat number: 3
Enter your name: Artem
Seat 1, 3 reserved by Artem
Reservation added to Tickets.json
Select a screening by entering its number (0-exit): _
```

Przykład wyświetlania biletu według nazwy

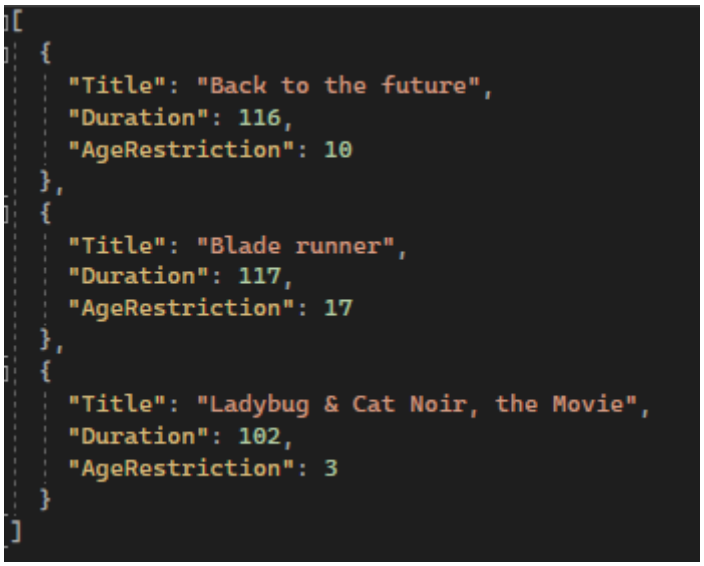
```
Ticket Information:
Movie: Blade runner
Screening Time: 08-26 06:00
Row: 1
Number: 3
Ticket Number: 1761181

Enter the Customer name (0-exit):
```

Tak wygląda plik Tickets.json



```
[
  {
    "Movie": "Back to the future",
    "ScreeningTime": "2023-08-27T03:00:00",
    "CustomerName": "John",
    "Row": 1,
    "Number": 2,
    "TicketNumber": 5282720
  },
  {
    "Movie": "Blade runner",
    "ScreeningTime": "2023-08-26T06:00:00",
    "CustomerName": "Artem",
    "Row": 1,
    "Number": 3,
    "TicketNumber": 1761181
  }
]
```


Tak wygląda plik Movies.json



```
[
  {
    "Title": "Back to the future",
    "Duration": 116,
    "AgeRestriction": 10
  },
  {
    "Title": "Blade runner",
    "Duration": 117,
    "AgeRestriction": 17
  },
  {
    "Title": "Ladybug & Cat Noir, the Movie",
    "Duration": 102,
    "AgeRestriction": 3
  }
]
```

Tak wygląda plik CinemHalls.json

```
  "ScreeningTime": "2023-08-24T10:00:00",
  "Movie": "Back to the future"
},
{
  "HallNumber": 2,
  "SeatList": [
    {
      "Row": 1,
      "Number": 1,
      "IsAvailable": false
    },
    {
      "Row": 1,
      "Number": 2,
      "IsAvailable": false
    },
    {
      "Row": 1,
      "Number": 3,
      "IsAvailable": false
    },
    {
      "Row": 2,
      "Number": 1,
      "IsAvailable": false
    },
    {
      "Row": 2,
      "Number": 2,
      "IsAvailable": true
    }
  ],
  "ScreeningTime": "2023-08-27T03:00:00",
  "Movie": "Back to the future"
},
{
  "HallNumber": 3,
  "SeatList": [
    {
      "Row": 1,
      "Number": 1,
      "IsAvailable": false
    },
    {
      "Row": 1,
```