Sprawozdanie
z przedmiotu
Programowanie w Języku Java
Laboratorium komputerowe nr 1

Autorzy:
Sylwia Jaworska
Grzegorz Listwan
Krzysztof Pacura

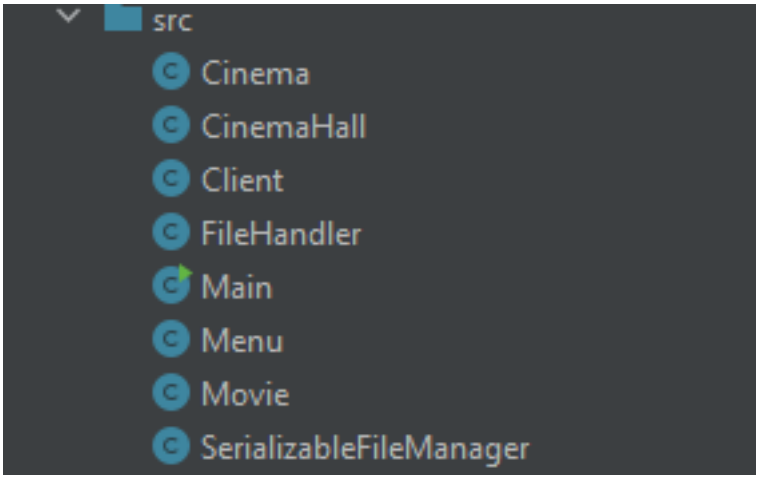
Na pierwszym laboratorium komputerowym zostało nam zadane następujące zadanie:

Napisz program realizujący rezerwację/kupno biletów do kina:

- Klasa Klient (nazwisko, imię, mail, telefon, seans, miejsce/miejsca - kolekcja)
- Klasa Seans (tytuł, dzień, godzina, ograniczenia wiekowe, liczba miejsc - HashMap<Character, HashMap<Integer, Boolean>>)
- zapis/odczyt danych z pliku, serializacja.

Poniżej przedstawiamy zrealizowaną przez nas wersję aplikacji.

Aplikacja składa się z 8 klas: Cinema, CinemaHall, Client, FileHandler, Main, Menu, Movie, SerializableFileManager. Program wykorzystuje serializację klasy Movie z zapisem i odczytem filmów do/z pliku *.dat, oraz mechanizm zapisu i odczytu klientów do/z pliku *.txt.



1. Klasa Main

Klasa ta jest główną uruchomieniową klasą programu. Zawiera statyczną metodę main która jest głównym punktem wejścia do programu. W tej metodzie tworzymy obiekt klasy Cinema następnie wywołujemy na nim metodę getMenu().

```
import java.io.IOException;
no usages  Grzegorz Listwan +2
public class Main {
    no usages  Grzegorz Listwan +2
    public static void main(String[] args) throws IOException, ClassNotFoundException {
        Cinema cinema = new Cinema();
        cinema.getMenu();
    }
}
```

2. Klasa Cinema

Klasa ta zawiera konstruktor który może rzucać wyjątki IOException, ClassNotFoundException. W klasie tej tworzymy nowy obiekt klasy Menu. Klasa ta zawiera też publiczną metodę getMenu() która też rzuca wyjątek IOException w metodzie tej na obiekcie menu wywołujemy metodę mainMenu().

```
import java.io.IOException;
2 usages  Grzegorz Listwan +2 *
public class Cinema {
    1 usage
    Menu menu = new Menu();
    1 usage  Krzysztof Pacura
    public Cinema() throws IOException, ClassNotFoundException {
    }
    1 usage  Grzegorz Listwan +2
    public void getMenu() throws IOException {
        menu.mainMenu();
    }
}
```

3. Klasa Menu

Klasa Menu zawiera konstruktor w którym odczytujemy dane seansów z serializowanego pliku *.dat i dodaje je do ArrayListy movies.

```
public class Menu {
    18 usages
    Scanner sc = new Scanner(System.in);
    7 usages
    ArrayList<Movie> movies;

    1 usage
    ArrayList<Client> clients = new ArrayList<>();


    1 usage  Grzegorz Listwan
    public Menu() throws IOException, ClassNotFoundException {
        try {
            movies = SerializableFileManager.readFromFile();
        }catch (FileNotFoundException e){
             movies = new ArrayList<>();
            System.out.println("Stworzono nową baze");
        }
    }
```

Ponadto klasa ta zawiera metody do komunikacji z użytkownikiem pozwalające wybrać opcje z menu głównego, wyświetlić dostępne seanse, zarezerwować film i miejsca, dodać nowe seanse oraz wyświetlić dokonane rezerwacje.

Metoda menu głównego z opcjami do wyboru.

```java
private final File fileName = new File( pathname: "clientsCinema.txt");

1 usage  Sylwia Jaworska +2
public void mainMenu() throws IOException {
    int input = 9;
    do {
        System.out.println("Witamy w naszsym kinie");
        System.out.println("Wybierz opcje;");
        System.out.println("1 - Wyświetl dostepne filmy");
        System.out.println("2 - Dodaj film");
        System.out.println("3 - Wyświetl rezerwacje");
        System.out.println("0 - Koniec");
        try {
            input = sc.nextInt();
        } catch (InputMismatchException e) {
            System.out.println("Niepoprawne dane");
        }
        sc.nextLine();
        switch (input) {
            case 0 -> {
                SerializableFileManager.writeToFile(movies);
                return;
            }
            case 1 -> displayMovie();
            case 2 -> addMovie();
            case 3 -> displayReservations();

        }
    } while (true);
}
```

Metoda dodająca nowy seans.

```java
private void displayReservations() {
    FileHandler.readFromFile(fileName);


}

1 usage  Sylwia Jaworska +1
private void addMovie()  {
    String title;
    String date;
    String hour;
    int ageLimit;

    System.out.println("Dodaj nowy film:");
    System.out.println("Tytuł: ");
    title = sc.nextLine();
    System.out.println("Data: (YYYY-MM-DD)");
    date = sc.nextLine();
    System.out.println("Godzina: (HH:MM)");
    hour = sc.nextLine();
    System.out.println("Ograniczenie wiekowe:");
    try {
        ageLimit = sc.nextInt();
        sc.nextLine();
        Movie movie = new Movie(title, LocalDate.parse(date), LocalTime.parse(hour), ageLimit, new CinemaHall().numberOfSeats());
        movies.add(movie);
        System.out.println("Dodano film");
    } catch (InputMismatchException | DateTimeParseException e) {
        System.out.println("Niepoprawne dane");
    }
}
```

Metoda wyświetlająca dostępne seanse oraz metoda do wyboru seansu który chcemy zarezerwować.

```
private void displayMovie() {
    for(int i=0; i<movies.size();i++){
        System.out.print((i+1)+". ");
        System.out.println(movies.get(i).toString());
    }
    selectMovie();
}

1 usage  Grzegorz Listwan +1
private void selectMovie() {
    System.out.println("Wbierz film który chcesz zarezerwować lub 0 - rezygnacja: ");
    try {
        int yourChose = sc.nextInt();
        sc.nextLine();
        if(yourChose==0)
            return;
        Movie yourMovie = movies.get(yourChose-1);
        createClient(yourMovie);
    }catch (InputMismatchException | IndexOutOfBoundsException e){
        System.out.println("Niepoprawne dane!");
    }
}
```

Metoda tworząca nowego klienta z podanych przez niego danych oraz na podstawie wybranego filmu i miejsc.

```
private void createClient(Movie movie){
    List<String> clientSeats = new ArrayList<>();
    System.out.println("Nazwisko: ");
    String lastName = sc.nextLine();
    System.out.println("Imię: ");
    String name = sc.nextLine();
    System.out.println("E-mail: ");
    String mail = sc.nextLine();
    System.out.println("Telefon: ");
    String phone = sc.nextLine();
    Movie chosenMovie = new Movie(movie.getTitle(), movie.getDay(), movie.getHour(), movie.getAgeRestriction());
    choseSeats(movie,clientSeats);
    Client client = new Client(lastName, name, mail, phone, chosenMovie, clientSeats);
    try {
        FileHandler.addToFile(client, fileName);
    }catch (IOException e) {

    }
    clients.add(client);
}
```

Metoda służąca do wybrania miejsc przez klienta.

```
private void choseSeats(Movie movie, List<String> clientSeats) {
    boolean flag = false;
    do {
        System.out.println();
        System.out.println("Widok sali kinowej");
        movie.displayCinemaRoom();
        char row;
        int seat=0;
            System.out.println("Wybierz rząd: ");
            row = sc.nextLine().charAt(0);
            System.out.println("Wybierz miejsce: ");
            try {
                seat = sc.nextInt();
                sc.nextLine();
                flag = movie.reservation(row, seat);
            } catch (InputMismatchException | NullPointerException e) {
                System.out.println("Niepoprawne dane!");
            }
            if(flag)
                clientSeats.add(row+"/"+seat);
        System.out.println("0 - koniec; 1 - zarezerwuj  miejsce");
        try {
            flag = sc.nextInt() == 0;
            sc.nextLine();
        }catch (InputMismatchException e){
            System.out.println("Błędne dane!");
        }
    }while(!flag);
}
```

4. Klasa CinemaHall

Metoda generująca HashMap z miejscami na sali kinowej.

```
public class CinemaHall {
    1 usage  Sylwia Jaworska
    public HashMap<Character, HashMap<Integer, Boolean>> numberOfSeats() {
        HashMap<Character, HashMap<Integer, Boolean>> seats = new HashMap<>();
        HashMap<Integer, Boolean> row = new HashMap<>();
        for (int i = 1; i <= 10; i++) {
            row.put(i, true);
        }
        for (int j = 0; j < 5; j++) {
            seats.put((char) (65 + j), row);
        }
        return seats;
    }
}
```

5. Klasa SerializableFileManager

Klasa ta odpowiada za zapis i odczyt z serializowanego pliku. Klasa ta jest wykorzytywana przy zapisie seansów do pliku.

```
public class SerializableFileManager {

    1 usage  Grzegorz Listwan
    public static void writeToFile(ArrayList<Movie> movies)throws IOException{
        ObjectOutputStream out = new ObjectOutputStream(new FileOutputStream( name: "movies.dat"));
        for(Movie movie:movies){
            out.writeObject(movie);
        }
        out.close();
    }
    1 usage  Grzegorz Listwan
    public static ArrayList<Movie> readFromFile() throws IOException,ClassNotFoundException {
        ObjectInputStream in = new ObjectInputStream(new FileInputStream( name: "movies.dat"));
        ArrayList<Movie> movies = new ArrayList<>();
        while (true){
            try {
                Movie movie = (Movie) in.readObject();
                movies.add(movie);
            }catch (IOException e){
                break;
            }
            }
        return movies;
    }
}
```

6. Klasa FileHandler

Klasa zawiera dwie statyczne metody do pracy z plikiem.

Metoda służąca do zapisu danych do pliku. Metoda ta przyjmuje obiekt typu Object oraz plik.

```
public class FileHandler {
    1 usage  Sylwia Jaworska +1 *
    public static void addToFile(Object object, File fileName) throws IOException {
        FileWriter f = new FileWriter(fileName,  append: true);
        try (BufferedWriter out = new BufferedWriter(f)) {
            out.write(object.toString());
            out.newLine();
        } catch (IOException e) {
            throw new RuntimeException(e);
        }

    }
```

Metoda do odczytu danych z pliku, oraz wyświetla dane użytkownikowi.

```
public static void readFromFile(File fileName) {
    try {
        FileInputStream f = new FileInputStream(fileName);
        DataInputStream in = new DataInputStream(f);
        BufferedReader r = new BufferedReader(new InputStreamReader(in));
        String strLine;
        int i = 1;
        while ((strLine = r.readLine()) != null) {
            String[] line = strLine.split( regex: ";");
            System.out.println(i++ + ".");
            System.out.println("Dane klienta: ");
            System.out.println(line[0]);
            System.out.println("Informacje o filmie: ");
            System.out.println(line[1]);
            System.out.println(line[2]);
            System.out.println();

        }
        in.close();
    } catch (IOException e) {
        System.out.println("Błąd dostępu do pliku");
    }
}
```

7. Klasa Client

Klasa ta zawiera pola reprezentujące różne cechy klienta, konstruktor do inicjalizacji obiektu oraz nadpisaną metodę toString();

```
public class Client {
    2 usages
    private final String lastName;
    2 usages
    private final String name;
    2 usages
    private final String email;
    2 usages
    private final String phone;
    5 usages
    private final Movie movie;
    2 usages
    private final List<String> clientSeats;

    1 usage  Grzegorz Listwan +1
    public Client(String lastName, String name, String email, String phone, Movie movie, List<String> clientSeats) {
        this.lastName = lastName;
        this.name = name;
        this.email = email;
        this.phone = phone;
        this.movie = movie;
        this.clientSeats = clientSeats;
    }
```

```java
@Override
public String toString() {
    String seats = "";
    for (String value : clientSeats) {
        seats += value + " ";
    }
    return "Nazwisko i Imię: " + lastName + "  " + name + ", Email: " + email + ", Telefon: " + phone + ";Film:  Tytuł: " + movie.getTitle() + ", Data: " +
            movie.getDay() + ", Godzina: " + movie.getHour() + ", Ograniczenie wiekowe: "
            + movie.getAgeRestriction() + ";Zarezerwowane miejsca(Rząd/Miejsce): " + seats;
}
```

8. Klasa Movie

Klasa ta zawiera pola reprezentujące róże cechy seansów, dwa przeciążone konstruktory.

```java
public class Movie implements Serializable {
    5 usages
    private final String title;
    5 usages
    private final LocalDate day;
    5 usages
    private final LocalTime hour;
    5 usages
    private final Integer ageRestriction;
    4 usages
    private HashMap<Character, HashMap<Integer, Boolean>> numberOfSeats;

    1 usage  Grzegorz Listwan
    public Movie(String title, LocalDate day, LocalTime hour, Integer ageRestriction) {
        this.title = title;
        this.day = day;
        this.hour = hour;
        this.ageRestriction = ageRestriction;
    }

    1 usage  Sylwia Jaworska +1
    public Movie(String title, LocalDate day, LocalTime hour, Integer ageRestriction, HashMap<Character, HashMap<Integer, Boolean>> numberOfSeats) {
        this.title = title;
        this.day = day;
        this.hour = hour;
        this.ageRestriction = ageRestriction;
        this.numberOfSeats = numberOfSeats;
    }
```

Nadpisaną metodę toString() oraz metodę displayCinemaRoom() która wyświetla widok Sali kinowej z zaznaczonymi miejscami(* - miejsce wolne, X – miejsce zajęte)

```java
@Override
public String toString() {
    return "Tytuł: " + title + ", Data: " + day + ", Godzina: " + hour + ", Wiek: " + ageRestriction;

}

1 usage  Krzysztof Pacura +2
public void displayCinemaRoom() {
    System.out.println("Tytuł: " + title + "   Data: " + day + "   Godzina: " + hour + "   Wiek: " + ageRestriction);
    System.out.println("  1 2 3 4 5 6 7 8 9 10");
    for (Map.Entry<Character, HashMap<Integer, Boolean>> row : numberOfSeats.entrySet()) {
        char keyRow = row.getKey();
        System.out.print(keyRow);
        HashMap<Integer, Boolean> seats = row.getValue();
        for (Map.Entry<Integer, Boolean> seat : seats.entrySet()) {
            boolean valueSeat = seat.getValue();
            if (valueSeat)
                System.out.print(" *");
            else
                System.out.print(" X");
        }
        System.out.println();
    }

}
```

Metodę reservation() odpowiedzialną za sprawdzenie czy dane miejsce jest dostępne a następnie za zarezerwowanie miejsca na podstawie przyjętych argumentów oraz getery.

```
    public boolean reservation(char row, int seat) {
        HashMap<Integer, Boolean> rowMap = new HashMap<>(numberOfSeats.get(row));
        boolean value = rowMap.get(seat);
        if(value) {
            rowMap.put(seat, false);
            numberOfSeats.put(row, rowMap);
            return true;
        }else {
            System.out.println("To miejsce jest już zajęte!!!");
            System.out.println("Proszę wybrać inne miejsce");
        return false;
        }

    }

    2 usages  Grzegorz Listwan
    public String getTitle() { return title; }

    2 usages  Grzegorz Listwan
    public LocalDate getDay() { return day; }

    2 usages  Grzegorz Listwan
    public LocalTime getHour() { return hour; }

    2 usages  Grzegorz Listwan
    public Integer getAgeRestriction() { return ageRestriction; }
}
```

Sposób działania aplikacji:

Widok menu głównego

```
Witamy w naszsym kinie
Wybierz opcje;
1 - Wyświetl dostepne filmy
2 - Dodaj film
3 - Wyświetl rezerwacje
0 - Koniec
```

1 – Wyświetl dostępne filmy.

```
1. Tytul: Iron Man, Data: 2023-10-20, Godzina: 10:00, Wiek: 16
2. Tytul: Piraci z Karaibów, Data: 2023-10-20, Godzina: 14:00, Wiek: 12
Wbierz flim który chcesz zarezerwować lub 0 - rezygnacja:
```

Oraz pytanie czy chcesz zarezerwować film.

Po wybraniu konkretnego filmu aplikacja prosi o podanie danych niezbędnych do dokonania rezerwacji.

```
Nazwisko:
Kowalski
Imię:
Jan
E-mail:
jan.kowalski@o2.pl
Telefon:
123123123
```

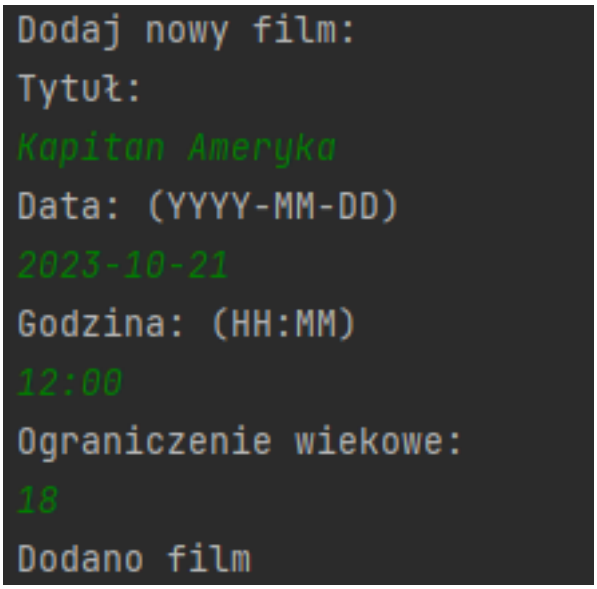
Po poprawnym podaniu danych wyświetla się widok Sali kinowej z dostępnymi miejscami oraz prośbą o wybranie rzędu i miejsca w rzędzie.

```
Widok sali kinowej
Tytuł: Iron Man   Data: 2023-10-20   Godzina: 10:00   Wiek: 16
  1 2 3 4 5 6 7 8 9 10
A X * * * * * * * * *
B * * * * * * * * * *
C * * * X X * * * * *
D * * * X X * * * * *
E * * * * X * * * * *
Wybierz rząd:
B
Wybierz miejsce:
9
0 - koniec; 1 - zarezerwuj  miejsce
```

Na zakończenie otrzymujemy pytanie czy chcemy zakończyć proces rezerwacji miejsc czy zarezerwować kolejne miejsce

2 – Dodaj film

Po wybraniu tej opcji aplikacja prosi o podanie danych dotyczących danego filmu. Po podaniu wszystkich niezbędnych danych otrzymujemy informacje o dodaniu filmu.



3 – Wyświetl rezerwacje

Po wybraniu tej opcji wyświetlają się wszystkie dokonane rezerwacje.

```
1.
Dane klienta:
Nazwisko i Imię: Nowak  Andrzej, Email: andrzej.nowak@o2.pl, Telefon: 123123123
Informacje o filmie:
Film:  Tytuł: Iron Man, Data: 2023-10-20, Godzina: 10:00, Ograniczenie wiekowe: 16
Zarezerwowane miejsca(Rząd/Miejsce): C/4

2.
Dane klienta:
Nazwisko i Imię: Kowalski  Jan, Email: jan.kowalski@o2.pl, Telefon: 123123123
Informacje o filmie:
Film:  Tytuł: Iron Man, Data: 2023-10-20, Godzina: 10:00, Ograniczenie wiekowe: 16
Zarezerwowane miejsca(Rząd/Miejsce): B/9
```

Aplikacja jest odporna na podanie niepoprawnych danych takich jak np.:

- błędy wybór w menu

```
Witamy w naszsym kinie
Wybierz opcje;
1 - Wyświetl dostepne filmy
2 - Dodaj film
3 - Wyświetl rezerwacje
0 - Koniec
4
Witamy w naszsym kinie
Wybierz opcje;
1 - Wyświetl dostepne filmy
2 - Dodaj film
3 - Wyświetl rezerwacje
0 - Koniec
t
Niepoprawne dane
```

- błędna data i godzina

```
Dodaj nowy film:
Tytuł:
222
Data: (YYYY-MM-DD)
222
Godzina: (HH:MM)
2
Ograniczenie wiekowe:
2
Niepoprawne dane
```

```
Dodaj nowy film:
Tytuł:
txt
Data: (YYYY-MM-DD)
tnt
Godzina: (HH:MM)
tnt
Ograniczenie wiekowe:
tnt
Niepoprawne dane
```

- błędny wybór filmu do zarezerwowania lub wpisane znaki zamiast cyfry

```
1. Tytul: Iron Man, Data: 2023-10-20, Godzina: 10:00, Wiek: 16
2. Tytul: Piraci z Karaibów, Data: 2023-10-20, Godzina: 14:00, Wiek: 12
3. Tytul: Kapitan Ameryka, Data: 2023-10-21, Godzina: 12:00, Wiek: 18
Wbierz flim który chcesz zarezerwować lub 0 - rezygnacja:
4
Niepoprawne dane!
```

```
1. Tytul: Iron Man, Data: 2023-10-20, Godzina: 10:00, Wiek: 16
2. Tytul: Piraci z Karaibów, Data: 2023-10-20, Godzina: 14:00, Wiek: 12
3. Tytul: Kapitan Ameryka, Data: 2023-10-21, Godzina: 12:00, Wiek: 18
Wbierz flim który chcesz zarezerwować lub 0 - rezygnacja:
tr
Niepoprawne dane!
```

- błędy wybór rzędu lub miejsca do zarezerwowania

```
Widok sali kinowej
Tytuł: Kapitan Ameryka   Data: 2023-10-21   Godzina: 12:00   Wiek: 18
  1 2 3 4 5 6 7 8 9 10
A * * * * * * * * * *
B * * * * * * * * * *
C * * * * * * * * * *
D * * * * * * * * * *
E * * * * * * * * * *
Wybierz rząd:
F
Wybierz miejsce:
11
Niepoprawne dane!
```

```
Widok sali kinowej
Tytuł: Kapitan Ameryka   Data: 2023-10-21   Godzina: 12:00   Wiek: 18
  1 2 3 4 5 6 7 8 9 10
A * * * * * * * * * *
B * * * * * * * * * *
C * * * * * * * * * *
D * * * * * * * * * *
E * * * * * * * * * *
Wybierz rząd:
3
Wybierz miejsce:
4
Niepoprawne dane!
0 - koniec; 1 - zarezerwuj  miejsce
Błędne dane!
```

Zawartość pliku *.txt

```
Nazwisko i Imię: Nowak  Andrzej, Email: andrzej.nowak@o2.pl, Telefon: 123123123;Film:  Tytuł: Iron Man, Data: 2023-10-20, Godzina: 10:00,
Nazwisko i Imię: Kowalski  Jan, Email: jan.kowalski@o2.pl, Telefon: 123123123;Film:  Tytuł: Iron Man, Data: 2023-10-20, Godzina: 10:00, 
```

Zawartość pliku *.dat

```
◆◆NULENQsrNULENQMovie◆┬SICENQ◆◆STXNULENQLNULSOageRestrictiontNULDC3Ljava/lang/Integer;LNULETXdaytNULNAKLjava/time/LocalDate;LNUL
numberOfSeatstNULDC3Ljava/util/HashMap;LNULENQtitletNULDC2Ljava/lang/String;xpsrNULDC1java.lang.IntegerDC2‥◆◆◆8STXNULSOHINULENQv
java.time.Ser◆]◆◆ESC"H◆FFNULNULxpwBELETXNULNULBEL◆
DC4xsqNUL~NUL
wSTXEOT◆xsrNULDC1java.util.HashMapENQBEL◆◆◆SYN`◆ETXNULSTXFNUL
loadFactorINUL  thresholdxp?@NULNULNULNULNULFFwBSNULNULNULDLENULNULNULENQsrNULDC3java.lang.Character4◆G◆kSUB&xSTXNULSOHCNULENQva
?@NULNULNULNULNULFFwBSNULNULNULDLENULNULNUL
sqNUL~NULBELNULNULNULSOHsrNULDC1java.lang.Boolean◆ r◆ˆ◆◆STXNULSOHZNULENQvaluexpNULsqNUL~NULBELNULNULNULSTXsqNUL~NULDC3SOHsqNUL~N
qNUL~NULSYNxsqNUL~NULSINULBsqNUL~NUL
?@NULNULNULNULNULFFwBSNULNULNULDLENULNULNUL
qNUL~NULDC2qNUL~NULSYNqNUL~NULNAKqNUL~NULSYNqNUL~NULETBqNUL~NULSYNqNUL~NULCANqNUL~NULSYNqNUL~NULEMqNUL~NULSYNqNUL~NULSUBqNUL~NUL
?@NULNULNULNULNULFFwBSNULNULNULDLENULNULNUL
qNUL~NULDC2qNUL~NULSYNqNUL~NULNAKqNUL~NULSYNqNUL~NULETBqNUL~NULSYNqNUL~NULCANsqNUL~NULDC3NULqNUL~NULEMsqNUL~NULDC3NULqNUL~NULSUB
?@NULNULNULNULNULFFwBSNULNULNULDLENULNULNUL
qNUL~NULDC2qNUL~NULSYNqNUL~NULNAKqNUL~NULSYNqNUL~NULETBqNUL~NULSYNqNUL~NULCANsqNUL~NULDC3NULqNUL~NULEMqNUL~NUL(qNUL~NULSUBqNUL~N
?@NULNULNULNULNULFFwBSNULNULNULDLENULNULNUL
qNUL~NULDC2qNUL~NULSYNqNUL~NULNAKqNUL~NULSYNqNUL~NULETBqNUL~NULSYNqNUL~NULCANqNUL~NULSYNqNUL~NULEMsqNUL~NULDC3NULqNUL~NULSUBqNUL
wBELETXNULNULBEL◆
DC4xsqNUL~NUL
wSTXEOT◆xsqNUL~NUL
?@NULNULNULNULNULFFwBSNULNULNULDLENULNULNULENQqNUL~NULDLEsqNUL~NUL
?@NULNULNULNULNULFFwBSNULNULNULDLENULNULNUL
```

Wnioski:

Stworzenie klas Klient i Seans pomaga w lepszej organizacji danych dotyczących klientów i seansów. Dzięki temu można przechowywać informacje na temat klientów i dostępności miejsc w wygodny i zrozumiały sposób. Klienci mogą dokonywać rezerwacji miejsc na konkretne seanse. Obsługa zapisu i odczytu danych z pliku pozwala na zachowanie danych klientów, seansów i dostępności miejsc między różnymi sesjami programu. To jest przydatne, gdy program jest zamykany i uruchamiany ponownie, a także przy przechodzeniu danych między różnymi komputerami. Serializacja pozwala na zapis i odczyt obiektów Java do/z pliku. To ułatwia przechowywanie i odzyskiwanie danych w bardziej elastyczny sposób. Wykorzystanie wyjątków pozwala na efektywniejszą obsługę błędów. Warto zauważyć, że program ten stanowi podstawę do dalszego rozwijania, na przykład poprzez dodanie interfejsu użytkownika (GUI) lub bardziej zaawansowanej obsługi seansów, cen biletów, płatności, współbieżności itp.