Документация на проекта Music Theatre

Явор Василев

22 януари 2023 г.

1 Цел и основни функционалности

Проектът има за цел да моделира система за управление на музикален театър. Чрез основните принципи на ООП и с помощта на езика Java се създава конзолно приложение, което обработва информацията въведена от потребителя и изпълнява зададените команди. Ключовите функционалности включват:

1. Създаване на зали и постановки
2. Отмяна на постановки
3. Продажба на билети на каса
4. Закупуване на билети от потребители
5. Отказване на билети
6. Проверяване на закупените билети

Предвид сложността на имплементацията на всички тези функционалности се наложи използването на множество класове, връзките между които ще бъдат обяснени по-долу.

2 Структура на проекта

При стартиране на програмата автоматично се извиква методът за регистрация на администратор. След успешна регистратиця този администратор автоматично се логва в системата и се извиква метод за посрещането му. Този метод за посрещане му дава възможност да избере едно от действията, които са му позволени да извърши. При излизане от профила системата се връща в начално състояние. От това състояние може или да се регистрира нов потребител, или да се влезе в профил. При регистрация автоматично се влиза в съответния профил. Така отново се извиква методът за посрещане на потребителя и се повтаря описаното по-горе. При избиране на опция за излизане от програмата от началното меню, то цялата информация се изтрива и програмата приключва.

Има четири вида потребители със съответните функционалности:

1. Администратор:
   (а) Създаване на зала
   (б) Създаване на постановка
   (в) Отмяна на постановка

2. Продавач на билети
   (а) Продаване на билет
   (б) Отказ на билет
3. Потребител, купуващ билети
   (а) Закупуване на билет
   (б) Отказ на билет
   (в) Показване на всички закупени билети
4. Проверяващ билетите
   (а) Проверка на билет по негов номер

3 Структура на класовете

В основата на проекта стоят два абстрактни класа User и Ticket, класът Performance, класът Hall и класът Seat. При създаване на потребител се създава обект от тип User, използвайки специфичния за вида потребител конструктор от наследиците на класа User: Admin, Cashier, Customer, Checker. Всеки обект от тип Performance има своя зала (обект на класа Hall), масив от места (обекти на класа Seat) и други не дотолкова важни за общата картина характеристики. Всяко място от своя страна има билет (обект на класа Ticket), зала и постановка. Всеки билет има постановка и място. Билетите се делят на два вида - OnlineTicket и PaperTicket. Онлайн билет се създава от Customer, а хартиен - от Cashier

Данните в приложението се съхраняват в масиви с променлива дължина (ArrayList), като има три такива масива:

1. ArrayList<User> users
2. ArrayList<Hall> halls
3. ArrayList<Performance> performances
4. ArrayList<Ticket> tickets

Посредством тези масиви изключително удобно може да се провери дали съществува напр. зала със същото име, билет с даден номер и т.н. Също така, тъй като обектите на всички класове са референтни типове данни, то всички промени по обектите, ще се отразяват и в тези масиви, защото обекти на класовете извън този масив няма да съществуват. Това е гарантирано, тъй като във всички конструктори създаденият обект се добавя в масива. Т.е. всички обекти на класовете имат референция в тези масиви.

4 Класът User

Класът User е абстрактен клас, който скицира функционалностите на всички потребители. Имплементира интерфейса Unique.

4.1 Характеристики

1. Статични константи за изисквания за потребителското име и паролата
2. username - обект от тип String
3. password - обект от тип String
4. registrationDate - обект от тип Date, стойност равна на датата в момента на създаване на потребителя

4.2 Конструктори

1. User() - създава празен обект
2. User(String, String) - дава стойности на username, password и registrationDate; Проверява дали потребителят е уникален т.е. дали не фигурира в масива users. Това се дължи на имплементацията на интерфейса Unique.

4.2.1 Медоти

1. Get методи за полетата; getPassword() е private от съображения за сигурност
2. void setUsername(String) - дава стойност на username, хвърля изключение ако не са изпълнени изискванията
3. void setPassword(String) - дава стойност на password, хвърла изключение ако не са изпълнени изискванията
4. static User registerUser(Scanner) - показва меню за избор на тип на потребител за регистрация и извиква метода за регистрация на наследника, който презаписва този метод
5. abstract void welcomeUser() - абстрактен метод за посрещане на потребителя и показване на всички възможни действия
6. static String inputPassword() - метод за въвеждане на парола във външен прозорец
7. static User loginUser(Scanner) - метод за логване на потребител; изисква потребителско име и парола; връща обект от тип User
8. booolean isUnique(User) - проверява дали обектът, подаден като параметър се съдържа в масива users; при намерено потребителско име хвърля изключение

5 Класът Admin

Класът Admin е наследник на абстрактния клас User и позволява регистрация на администратор и изпълняване на съответните му функционалности.

5.1 Характеристики

1. Друга стойност на статичната константа за минимална дължина на паролата

5.2 Конструктори

1. Admin(String, String) - извиква конструктора на суперкласа със същите параметри

5.3 Методи

1. void setPassword(String) - проверява дали е спазено допълнителното условие за паролата (дължина по-голяма от константата) и извиква същия метод на суперкласа

2. static User registerUser(Scanner) - въвежда се потребителско име и парола и се връща обект от тип User, с действителен тип Admin

3. void welcomeUser(Scanner) - въвежда се опция от менюто и се вика съответния метод

4. void createHall(Scanner) - създава зала за представления, като се задават въпроси за име на залата, брой редове и брой места на всеки ред; към момента може да се създават само правоъгълни зали

5. void createPerformance(Scanner) - създава постановка, като се избира зала, заглавие на постановката и други детайли; Тук се избира и каква да е цената на всяко място в залата, като може да се избира отделна категория за всеки ред. Типовете места са в enum SeatClass, като са фиксирани на три броя. Избира се и цена за всеки клас място.

6. void cancelPerformance(Scanner) - отменя постановка, като всеки билет за тази постановка се маркира като невалиден

6 Класът Cashier

Класът Cashier е наследник на абстрактния клас User и позволява регистрация на продавач на билети и изпълняване на съответните му функционалности.

6.1 Характеристики

1. Няма специфични характеристики

6.2 Конструктори

1. Cashier(String, String) - извиква конструктора на суперкласа със същите параметри

6.3 Методи

1. static User registerUser(Scanner) - въвежда се потребителско име и парола и се връща обект от тип User, с действителен тип Cashier

2. void welcomeUser(Scanner) - въвежда се опция от менюто и се вика съответния метод

3. void sellTicket(Scanner) - избира се постановка и място и се създава обект от деклариран тип Ticket, но действителен тип - PaperTicket

4. void cancelTicket(Scanner) - въвежда се номер на билет и се извиква метод за отказване на билета; възстановява се част от сумата на билета

7 Класът Customer

Класът Customer е наследник на абстрактния клас User и позволява регистрация на клиент и изпълняване на съответните му функционалности.

7.1 Характеристики

1. Статична константа за максимална дължина на имейл
2. String email - обект от тип String; при отмяна на постановка, за която има закупен билет или при отказване на билет се изпраща писмо на този имейл (посредством съобщение в конзолата - реален имейл не се праща)
3. String realName - обект от тип String; реалното име се отбелязва на закупения билет (обект от тип OnlineTicket)
4. ArrayList<Ticket> ticketsOfUser - използва се за съхранение на всички билети на съответния потребител и тяхното печатане; съхранява същата референция като тази, използвана в масива в Ticket класа

7.2 Конструктори

1. Customer(String, String, String, String) - създава обект от тип Customer, първо извиква конструктора на суперкласа и след това задава стойности на email и realName; инициализира празен ArrayList от билети за потребителя

7.3 Методи

1. void setEmail(String) - дължина по-малка от тази на константата иначе хвърля изключение
2. void setRealName(String) - само букви и интервали иначе хвърля изключение
3. static User registerUser(Scanner) - въвежда се потребителско име и парола и се връща обект от тип User, с действителен тип Customer
4. void welcomeUser(Scanner) - въвежда се опция от менюто и се вика съответния метод
5. void buyTicket(Scanner) - закупуване на OnlineTicket; добавяне в ticketsOfUser
6. void cancelTicket(Scanner) - въвежда се номер на билет и се извиква метод за отказване на билета
7. void showTickets() - извежда всички валидни билети, закупени от този потребител

8 Класът Checker

Класът Checker е наследник на абстрактния клас User и позволява регистрация на лице, проверяващо билетите и изпълняване на съответните му функционалности.

8.1 Характеристики

1. Статична константа за максимална дължина на имейл
2. String email - обект от тип String; при отмяна на постановка, за която има закупен билет или при отказване на билет се изпраща писмо на този имейл (посредством съобщение в конзолата - реален имейл не се праща)
3. String realName - обект от тип String; реалното име се отбелязва на закупения билет (обект от тип OnlineTicket)
4. ArrayList<Ticket> ticketsOfUser - използва се за съхранение на всички билети на съответния потребител и тяхното печатане; съхранява същата референция като тази, използвана в масива в Ticket класа

8.2 Конструктори

1. Customer(String, String, String, String) - създава обект от тип Customer, първо извиква конструктора на суперкласа и след това задава стойности на email и realName; инициализира празен ArrayList от билети за потребителя

8.3 Методи

1. void setEmail(String) - дължина по-малка от тази на константата иначе хвърля изключение
2. void setRealName(String) - само букви и интервали иначе хвърля изключение
3. static User registerUser(Scanner) - въвежда се потребителско име и парола и се връща обект от тип User, с действителен тип Customer
4. void welcomeUser(Scanner) - въвежда се опция от менюто и се вика съответния метод
5. void buyTicket(Scanner) - закупуване на OnlineTicket; добавяне в ticketsOfUser
6. void cancelTicket(Scanner) - въвежда се номер на билет и се извиква метод за отказване на билета
7. void showTickets() - извежда всички валидни билети, закупени от този потребител