Проект по Обектно-ориентирано програмиране

Тема: “Билети”

Изготвил:

Кристина Бориславова Попова, ф.н. 81933, Компютърни науки, втори курс, втори поток, шеста група

1.Увод

1.1. Описание и идея на проекта

Основната идея на проекта “Билети” е реализация на информационна система, която обслужва билетна каса. Проектът прочита и обработва данните за дадено представление от текстов файл (с разширение .txt), позволявайки на потребителя да запазва, отменя или закупува билет за прочетените или създадените, по време на програмата, представления на дадена дата, в дадена зала. След указание на потребителя направените промени могат да се запишат в първоначално отворения файл или да се запазят в нов файл, указан от потребителя. Ако потребителят не е указал какво да се направи с новите промени, те не се запазват никъде.

1.2. Цел и задачи на разработката

Целта на проекта е създаването на билетна каса, която да поддържа съответните функционалности, с които позволява на потребителя да:

- отваря съдържанието на даден файл;
- затваря отворения файл;
- записва направените промени обратно в същия файл;
- записва направените промени в нов, указан от потребителя, файл;
- вижда кратка информация за поддържаните от програмата команди;
- излиза от програмата;
- създава ново представление, като проверява дали вече не съществува в системата представление на тази дата и в тази зала;
- вижда справка за свободните места на дадено представление;
- запазва билет;
- отменя резервацията на билет;
- закупува билет;
- вижда справка за запазените билети;
- да провери валидността на билет;
- вижда справка за закупените билети;
- вижда статистика за най-гледаните представления
- вижда статистика за представления с под 10% посещаемост

2. Преглед на предметната област

2.1. Основни дефиниции и концепции , които ще бъдат използвани

Обектно-ориентираното програмиране е сравнително нова програмистка парадигма, принуждаваща програмистите да преразгледат мисленето си за програмирането, за смисъла на изпълнение на програмата и за това как информацията да бъде структурирана в компютъра.

Обектно-ориентираното програмиране е съсредоточено върху обектите, които капсулират състояние и поведение. Тази дефиниция оприличава обектите на променливи величини от абстрактни типове данни. Възможността за дефиниране на класове позволява да се създават и обработват типове данни, които липсват в езика и да се създават специфични приложения.

Средствата за дефиниране на производни класове и наследяването са сред най-важните характеристики на обектно-ориентираното програмиране. Като се използва механизмът на наследяване от съществуващ клас може да се създаде нов клас, който да използва вече създадените компоненти и поведение на базовия/те си клас/ове. Така се избягва повторното дефиниране на тези компоненти и поведение.

Терминът абстрактен тип данни е много важен в компютърната наука и особено в обектно-ориентираното програмиране. Абстрактните типове данни са пример за реализирането на пакетни данни и операции в едно цяло, позволяващо тяхната взаимосвързаност. Такива средства предоставят класовете. Информационното скриване на реализацията на типа данни се изразява в това, че програмист, използващ величина от някакъв абстрактен тип данни, не трябва да бъде напълно запознат с вътрешната структура на типа. Начинът, по който типът е реализиран, трябва да бъде скрит от потребителя. Открити трябва да са единствено свойствата на типа, които се изразяват чрез неговите операции.

3.Проектиране

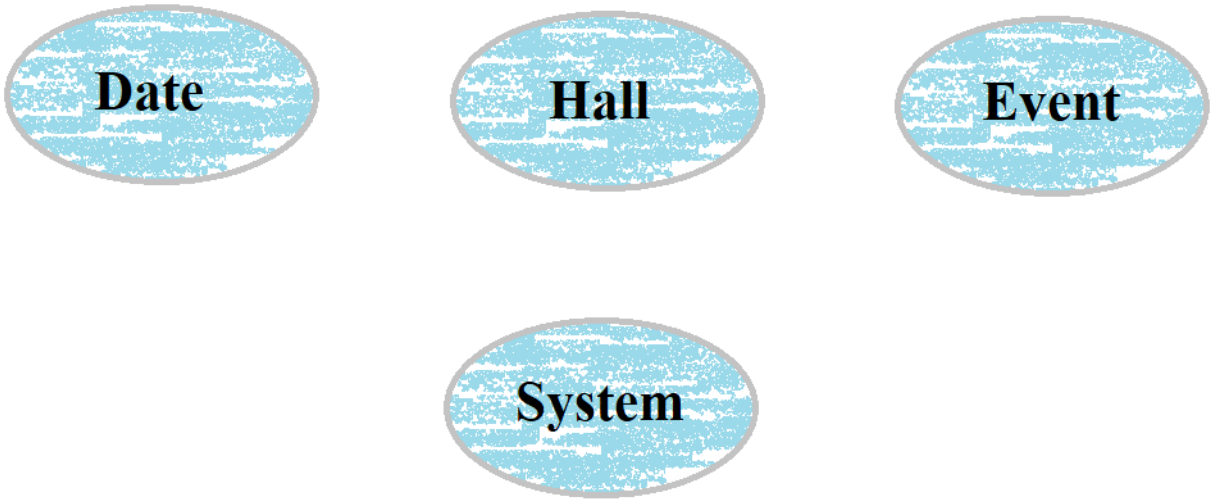
3.1.Обща архитектура – ООП дизайн

За проекта „Билети“ бяха реализирани класовете `Date, Hall, Event` и `System.` За работата с командите, беше използван поведенческият шаблон за дизайн - *command pattern,* за който бяха необходими следните класове :

- `Invoker`
- `ICommand`
- `Validator –` в който се валидира въведения вход от потребителя.
- `OpenCommand, SaveCommand, ExitCommand, SaveAsCommand, CloseCommand, HelpCommand, FreeSeatsCommand, BookCommand, UnbookCommand, BuyCommand, CheckCommand, BookingsCommand, ReportCommand и UnderTenCommand`

3.2. Диаграми

На фиг.1 са изобразени основните класове използвани в архитектурата на проекта “Билети”. Класът `Date` и класът `Hall` се включват в класа `Event`, защото всяко представление си има дата и зала. Класът `System` играе ролята на мястото, където се провеждат самите представления (театър, кино център и т.н.). В него се съдържат всички представления, зали и функционалности, които билетната каса предлага.



фиг.1

4.Реализация и тестване

4.1. Реализация на класове

Класът `Date` е датата, на която се провеждат представленията. По-важните член-функции реализирани в този клас са:

- `bool operator==(const Date& other)const;` - която връща истина, ако две дати са едни и същи, и лъжа, когато са различни. Тази функционалност е необходима за да можем да проверяваме дали две представления се провеждат на една и съща дата.
- `int getDay()const;` - *гетър* връщащ деня.
- `int getMonth()const;` - *гетър* връщащ месеца.
- `int getYear()const;` - *гетър* връщащ годината.

Класът `Hall` е залата, в която се провеждат представленията. Част от член функциите са:

- `bool operator==(const Hall& other)const;` - проверяваща дали две дали са едни и същи;

- `int getNumber()const;` - гетър връщащ номера на залата.
- `int getRows()const;` - гетър връщащ броя на редовете в залата.
- `int getCapacity()const;` - гетър връщащ капацитета на залата (редове*брой места на ред).
- `int getSeats()const;` - гетър връщащ броя на местата на ред.

Класът `Event` е представлението. По-важните член-функции реализирани в този клас са:

- `void createId(int row, int seat);` - генерира идентификатор, за всяко запазено или закупено място. Идентификаторът е от вида "ред-място-бележка-година-месец-ден".
- `bool operator==(const Event& other)const;` - проверява дали две представления са едни и същи;
- `std::string getName()const;` - *гетър* връщащ името на представлението.
- `Date getDate()const;` - *гетър* връщащ датата на представлението.
- `Hall getHall()const;` - *гетър* връщащ залата на представлението.
- `std::string getId(int row, int seat)const;` - *гетър* връщащ идентификаторът на дадено място.
- `int getFreeSeats()const;` - *гетър* връщащ броя на свободните места за представлението.
- `int getSoldSeats()const;` - *гетър* връщащ броя на продадените места за представлението.
- `void bookSeat(int row, int seat, std::string name);` - запазва билет на даден ред и дадено място с бележка.
- `void unbookSeat(int row, int seat);` - отменя резервацията билет на даден ред и дадено.
- `void buy(int row, int seat, std::string name);` - закупува билет на даден ред и дадено място с бележка.
- `void save(std::ostream& os);` - записва данните на дадено представление.

В класът `System` държим всички прочетени и добавени от потребителя представления и зали, за да можем да работим с тях. Имаме две допълнителни член-данни `std::string fileName`, в която помним името на последно отворения файл, и `bool isOpen`, която е `true`, когато сме отворили даден файл, и `false`, когато не сме отворили даден файл или сме затворили вече отворения такъв. Член-функциите реализирани в този клас са всички операции, които трябва да поддържа билетната система.

- `void open(const std::string fileName);`
- `void save();`
- `void saveAs(const std::string fileName);`
- `void close();`
- `void help();`

- `void exit();`
- `void addEvent(Event* event);`
- `void freeSeats(const std::string name, const Date& date);`
- `void book(const std::string name, const Date& date, int row, int seat, const std::string note);`
- `void unbook(const std::string name, const Date& date, int row, int seat);`
- `void buy(const std::string name, const Date& date, int row, int seat, const std::string note);`
- `void bookings(const std::string name, const Date& date)const;`
- `void bookings(const std::string name)const;`
- `void bookings(const Date& date)const;`
- `void check(std::string id);`
- `void report(const Date& from, const Date& to,int hall);`
- `void report(const Date& from, const Date& to);`
- `void mostWatched();`
- `void underTen(const Date& from, const Date& to, std::ostream& os);`

4.2. Създаване на тестови сценарии (създаване на примери)

Системата беше тествана върху следните тестови сценарии:

- Отваряне на несъществуващ файл, при което системата създава нов файл и извежда съобщение за добавяне на зали и представления.

- Създаване на представление във вече заета зала, при което системата връща грешка и не добавя представлението.

- Запазване на билет на вече запазено или закупено място, при което системата извежда, че мястото е запазено или закупено, и не прави промени по билетите.

- Отменяне на вече закупен билет, при което системата извежда съобщение, че билетът вече е закупен. Позволява се отменяне на билет само при запазени билети.

- Закупуване на вече закупен билет, при което системата връща съобщение, че билетът вече е бил закупен. Закупуване на вече запазен билет, при което система закупува билета, само ако той е бил запазен от същия човек (прави се проверка с идентификаторите). Закупуване на билет, който не е бил запазван предварително.

5.Заключение

5.1. Обобщение на изпълнението на началните цели

Проектът удовлетворява исканите функционалности. Бяха създадени, разработени и разширени (с допълнителни член-функции и член-данни) исканите класове. Създадоха се и допълнителни класове, с които да се изпълнят поставените задачи. Системата беше тествана и коригирана, когато не отговаряше на изискванията.

5.2. Насоки за бъдещо развитие и усъвършенстване

Нещата, които биха усъвършенствали проекта са:

- създаване на графичен интерфейс за по-удобно използване на системата.
- добавяне на допълнителни функционалности и член-данни на класовете, с които да се разшири обхвата на билетната система, като например - добавяне на точен час на представленията, за да може в един ден повече от едно представление да се провежда в дадена зала; добавяне на цени за всеки един билет (с варианти за намаления за деца, ученици, студенти, пенсионери и т.н.);

Използвана литература

1. Тодорова, Магдалина, “Обектно-ориентирано програмиране на базата на езика C++”,