|  |  |
| --- | --- |
| sigla_usv_nou | Universitatea „Ştefan cel Mare” – Suceava Facultatea de Inginerie Electrică şi Ştiinţa Calculatoarelor Specializarea Calculatoare |

PROIECT DISCIPLINA POO

Autor:

Gratii Stanislav

- SUCEAVA 2023 -

CUPRINS

Cuprins

1.. [TEMA SI MOTIVATIA ALEGERII 2](file:///C:\Users\grati\Desktop\POO%20PROIECT\Model_proiect_POO.doc#_Toc338017069)

[1.1. DESCRIEREA PROBLEMEI 4](file:///C:\Users\grati\Desktop\POO%20PROIECT\Model_proiect_POO.doc#_Toc338017071)

[1.2. ABORDAREA TEORETICA A PROBLEMEI 4](file:///C:\Users\grati\Desktop\POO%20PROIECT\Model_proiect_POO.doc#_Toc338017072)

[1.3. ELEMENTE SPECIFICE POO 4](file:///C:\Users\grati\Desktop\POO%20PROIECT\Model_proiect_POO.doc#_Toc338017073)

[2. IMPLEMENTARE 5](file:///C:\Users\grati\Desktop\POO%20PROIECT\Model_proiect_POO.doc#_Toc338017075)

[2.1. TEHNOLOGII FOLOSITE 5](file:///C:\Users\grati\Desktop\POO%20PROIECT\Model_proiect_POO.doc#_Toc338017076)

[2.2. DIAGRAMA DE CLASE 5](file:///C:\Users\grati\Desktop\POO%20PROIECT\Model_proiect_POO.doc#_Toc338017077)

[2.3. ALTE CAPITOLE SPECIFICE 5](file:///C:\Users\grati\Desktop\POO%20PROIECT\Model_proiect_POO.doc#_Toc338017078)

[3. ANALIZA SOLUTIEI IMPLEMENTATE 6](file:///C:\Users\grati\Desktop\POO%20PROIECT\Model_proiect_POO.doc#_Toc338017079)

[3.1. FORMATUL DATELOR DE I/O 6](file:///C:\Users\grati\Desktop\POO%20PROIECT\Model_proiect_POO.doc#_Toc338017080)

[3.2. STUDII DE CAZ 6](file:///C:\Users\grati\Desktop\POO%20PROIECT\Model_proiect_POO.doc#_Toc338017081)

[3.3. PERFORMANTE OBTINUTE 6](file:///C:\Users\grati\Desktop\POO%20PROIECT\Model_proiect_POO.doc#_Toc338017082)

[4. MANUAL DE UTILIZARE 7](file:///C:\Users\grati\Desktop\POO%20PROIECT\Model_proiect_POO.doc#_Toc338017083)

[5. CONCLUZII 8](file:///C:\Users\grati\Desktop\POO%20PROIECT\Model_proiect_POO.doc#_Toc338017084)

[6. BIBLIOGRAFIE 9](file:///C:\Users\grati\Desktop\POO%20PROIECT\Model_proiect_POO.doc#_Toc338017085)

[6.1. CARTI 9](file:///C:\Users\grati\Desktop\POO%20PROIECT\Model_proiect_POO.doc#_Toc338017086)

[6.2. ARTICOLE 9](file:///C:\Users\grati\Desktop\POO%20PROIECT\Model_proiect_POO.doc#_Toc338017087)

[6.3. SURSE BIBLIOGRAFICE DIVERSE 9](file:///C:\Users\grati\Desktop\POO%20PROIECT\Model_proiect_POO.doc#_Toc338017088)

1.TEMA SI MOTIVATIA ALEGERII

Tema acestui proiect consta in Gestiunea unei parcări gratuite cu durată limitată de 2 ore, iar pentru perioadele mai lungi se va introduce plata în funcție de ora parcată.

Motivul alegerii acestui proiect este problema cu care ne confruntăm zilnic și care afectează în mod direct viața noastră. Din cauza numărului mare de autovehicule în circulație și a numărului limitat de locuri de parcare. Această problemă poate duce la aglomerarea străzilor, la creșterea timpului de căutare a unui loc de parcare. Prin urmare, este important să dezvoltăm soluții pentru gestionarea mai eficientă a locurilor de parcare și pentru a facilita procesul de parcare pentru șoferi.

1.1. DESCRIEREA PROBLEMEI

Principiu funtionararii acestei parcacri :

La intrarea în parcare, șoferul trebuie să ia un bilet de parcare, care poate fi generat de un dispozitiv automatizat sau poate fi dat de operator. Biletul ar trebui să conțină un cod unic(Codul unic de pe biletul de parcare este necesar pentru a putea identifica în mod unic fiecare șofer care intră și iese din parcarea cu plată) și ora de intrare a mașinii. La ieșirea din parcare, șoferul trebuie să introducă biletul de parcare într-un dispozitiv de plată. Dispozitivul ar trebui să citească codul unic de pe bilet și să calculeze timpul petrecut în parcare. Dacă timpul petrecut este sub 2 ore, dispozitivul nu ar trebui să ceară nicio plată. Dacă timpul petrecut este peste 2 ore, dispozitivul ar trebui să ceară o plată de 5 lei. Dacă timpul petrecut este mai mare de 3 ore , dispozitivul ar trebui să calculeze plata în funcție de timpul petrecut și să ceară plata corespunzătoare. Pentru fiecare ora in plus se va introduce un leu la cei 5.

O problema care poate apărea în timpul implementării este:

Una dintre problemele care ar putea apărea în ceea ce privește parcare gratuită pentru angajați este faptul că aceștia ar putea fi tentați să selecteze opțiunea "Angajat" chiar dacă nu sunt angajați, doar pentru a obține acces gratuit la parcare. Pentru a evita această situație, putem implementa un sistem de verificare a datelor. Astfel, în momentul în care un angajat alege opțiunea "Angajat", va trebui să introducă un cod unic care va fi setat in fisierul “angajat.txt”. În plus, se poate solicita confirmarea suplimentară de la angajator pentru a verifica dacă persoana care utilizează codul este într-adevăr un angajat și are dreptul de a beneficia de acces gratuit în parcare. Aceste măsuri vor asigura că doar angajații vor beneficia de parcare gratuită și că sistemul va funcționa în mod eficient.în cazul în care angajatul greșește codul unic el mai are la dispoziție 3 încercări dacă le greșește și pe acestea ,în parcare automobilul va fi introdus ca client nu că angajat

Acest proiect ar poate duce la o utilizare mai eficientă a spațiului disponibil și la reducerea supraaglomerării dar si Încurajarea respectării normelor de parcare și a limitelor de timp, ceea ce poate duce la o reducere a abuzurilor și a neregulilor în ceea ce privește utilizarea parcării.

1.2 Aboradarea Teoretica a problemei

Dacă mașina a stat 3 ore și 24 de minute în parcarea cu plată, putem calcula plata în funcție de următoarele reguli:

- Primele 2 ore sunt incluse în tariful de bază și nu necesită o plată suplimentară deci ne ramane sa calculam plata pentru doar 1 ora si 24 minute = 1.4 ore

-Prima ora: 5 lei.

-24 minute = 0.083(5/60 = 0,083lei/minut)\*24=1,99 lei

Deci, pentru 3 ore și 24 de minute, va trebui să plătiți 6,99 lei

1.3 Elemente Specifice POO

În ceea ce privește implementarea acestui sistem de parcare, există mai multe elemente specifice programării orientate pe obiecte (POO) care ar putea fi utilizate.

Acestea includ:

Clase: Codul definește mai multe clase, inclusiv BiletParcare, Dispozitiv, Settings, CarClient, CarAngajat, Parcare,CitireNumar și Administrare\_Fisier.

Încapsulare: Fiecare clasă își încapsulează membrii de date și funcțiile membre, ascunzând detaliile de implementare din lumea exterioară. De exemplu, clasa Parcare încapsulează detaliile și operațiunile parcării, iar clasele CarClient și CarAngajat încapsulează informații despre mașini.

Moștenire: clasa CarAngajat moștenește din clasa CarClient, reprezentând o relație „este-a”. CarAngajat moștenește proprietățile și metodele CarClient și adaugă propriile sale funcționalități specifice.

Polimorfism: Polimorfismul este utilizat prin supraîncărcare de funcții și funcții virtuale. De exemplu, funcțiile AddCarClient și AddCarAngajat din clasa Parcare sunt supraîncărcate pentru a gestiona diferite tipuri de mașini. În plus, funcția de autentificare din clasa CarAngajat este o funcție virtuală care poate fi suprascrisă de clasele derivate.

Abstracție: Clasele și funcțiile lor membre oferă abstractizare prin ascunderea detaliilor complexe de implementare și expunerea doar a interfețelor necesare. De exemplu, clasa Parcare abstractizează funcționalitatea parcării, iar clasele CarClient și CarAngajat abstractizează informațiile și operațiunile despre mașină.

Gestionarea fișierelor: clasa Administrare\_Fisier se ocupă de operațiuni cu fișiere, cum ar fi citirea și scrierea în fișiere.

2. Implementare

Pentru a rezolva tema propusă, am început prin identificarea cerințelor și a obiectivelor. Apoi, am efectuat o cercetare extensivă pentru a obține informațiile necesare pentru a aborda problema în mod adecvat. Am căutat resurse, precum informația despre modul de calculare a plății în această parcare

După ce am dobândit o înțelegere solidă a subiectului, am creat un plan detaliat pentru rezolvarea temei: am identificat clasele principale care vor fi implicate în implementarea sistemului. Acestea includ clasele "Parcare", "BiletParcare" și "CarClient",”CarAngajat”. și am stabilit o structură logică pentru a aborda fiecare aspect al problemei. Am folosit apoi cunoștințele mele de programare și abilitățile tehnice pentru a implementa soluția propusă.

2.1 Tehnologii folosite

Limbaje de programare: c++

Medii IDE (Integrated Development Environment): CodeBlocks

2.2. DIAGRAMA DE CLASE, SCHEMA BLOC, WORKFLOW

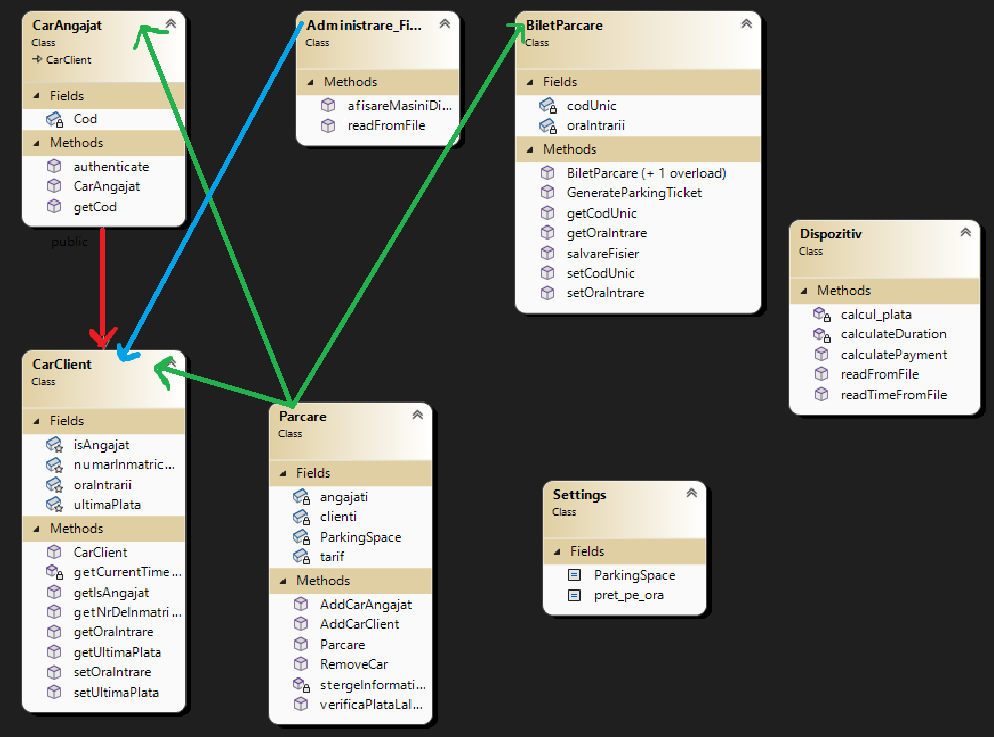


Figura 1. Digrama UML de clase a aplicatiei

Cu verde-> Relația de agregare/compoziție: Clasa "Parcare" utilizează clasele "CarAngajat" și "CarClient" prin adăugarea lor în parcare cu ajutorul metodelor "AddCarAngajat" și "AddCarClient". Precum si clasa BiletParcare pentru a genera un ticket si pentru a salva in fisier

Cu Rosu->Relația de moștenire: Clasa "CarAngajat" este o clasă derivată din clasa de bază "CarClient".

Cu albastru->Relația de asociere: Clasa "Administrare\_Fisier" utilizează clasa "CarClient" prin metoda "readFromFile".

3. Analiza soluției implementate

3.1 Formatul datelor de I/O

Datele de intrare:

Pentru citirea numărului de înmatriculare în opțiunea 1:utilizatorul este invitat să introducă numărul de înmatriculare(dar să zicem că acest lucru se face automat cu o camera)

Numărul de înmatriculare introdus de utilizator este stocat în variabila nrInmatriculare.

Pentru introducerea mașini în parcare că angajat sau că client în opțiunea 2:

Utilizatorul este întrebat dacă este angajat sau nu. Utilizatorul introduce un răspuns de tip caracter ('y' sau 'n'). Dacă răspunsul este 'y' sau 'Y', utilizatorul este invitat să introducă un cod unic. Codul introdus este verificat într-un fișier și este autentificat sau nu. Numărul de înmatriculare introdus anterior (nrInmatriculare) și autentificarea angajatului sunt utilizate pentru a crea un obiect de tip CarAngajat sau CarClient, care este adăugat în parcarea parcare.

Pentru extragerea unei mașini din parcare în opțiunea 3:

Utilizatorul este invitat să introducă numărul de înmatriculare al mașinii de extras. Numărul de înmatriculare introdus este stocat în variabila ID.

Datele de ieșire:

Mesaje informative care sunt afișate în consolă pentru a comunica rezultatele acțiunilor efectuate. Dacă autentificarea angajatului în opțiunea 2 este corectă, se afișează un mesaj de succes. În opțiunea 3, se afișează suma de plată și se solicită introducerea bancnotelor pentru achitare. Dacă plata este suficientă sau insuficientă, se afișează un mesaj corespunzător. În opțiunea 4, se afișează mașinile din parcare înregistrate într-un fișier.

3.2 Studio de caz

A screen shot of a computer

Description automatically generated with low confidence

Sa introducem optiunea 1: introducem un număr de imatriculare putem fie cu litere mici fie cu majuscule

A screenshot of a computer screen

Description automatically generated with medium confidence

Să introducem acum optiunea2: suntem invitați să alegem dacă suntem angat sau nu să alegem Y după trebuie să introducem un cod unic care este stocat în fișierul (angaj.txt)

În caz în care greșim mai avem 3 încercări

Dacă greșim toate cele 3 încercări mașină va fi adăugat că client dar dacă introducem codul correct mașină va fi adăugat că Angajat.Se afiseaza un mesaj cu Biletul Parcare si se salveaza in fisier.

A screenshot of a computer

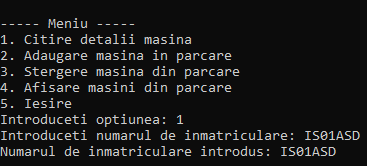
Description automatically generated with medium confidence

Sa introducem optiunea 3:

Introducem din nou numarul de inmatriculare: sv98GTM (fie cu majuscule fi cu minuscule)

Se afiseaza mesajul Suma catre achitare este de: 0 deoarce Masina a fost parcata ca Angajat

Sa incercam acum un client sa introducem.

A screen shot of a computer

Description automatically generated with medium confidenceA screen shot of a computer

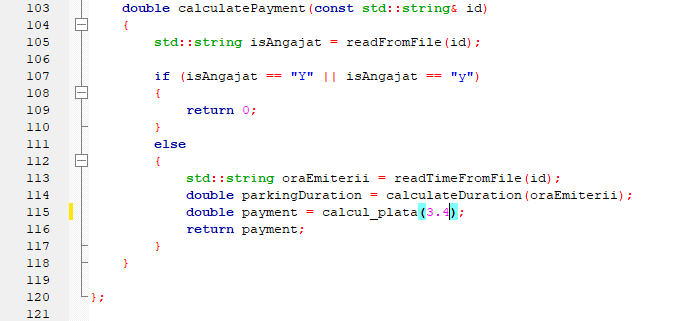
Description automatically generated with medium confidence

Și de dată această Suma către achitare este de:0 fiindcă nu au trecut 2 ore de când a intrat mașînă in parcare

Sa incercam daca masina a stat mai mult de 2 ore se afiseaza correct suma catre achitare

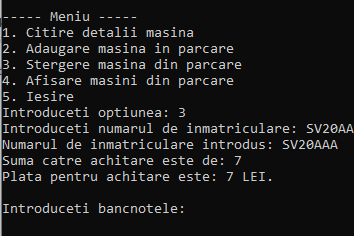
Fie modificam in cod in clasa Dispozitiv la linia 115 in loc de parkingDuration introducem 3.4 sa verificam daca rezultatul va fi 7 lei ca in capitolul 1.2 Aboradarea Teoretica a problemei

Sau alegem o masina din fisier care a stat mia mult de 2 ore in cazul meu aleg prima optiune



A screenshot of a computer program

Description automatically generated with medium confidence

Și cum se observă Plata pentru Achitare este: 7 Lei

Acum trebuie să introducem Banconte să zicem că introducem 5 leiA picture containing text, screenshot, font

Description automatically generated

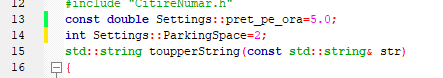
Se afișează un mesaj Plata Insuficientă

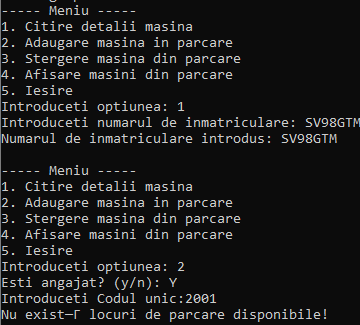
Și mai punem 5 lei

A black screen with white text

Description automatically generated with low confidence

Se afișează un mesaj Plata este suficientă și restul returnat de apparat

Să verificăm acum dacă numărul de locuri disponibile ajunge la Maxim pentru această schimbăm în cod la linia 14  
  
în loc de 100 cum era de la început punem 2 (ParkingSpace se setează de la început și cu câte mașini avem în fișierul “bilet.txt”) exemplu dacă în bilet.txt avem 70 mașini la începutul programului ParkingSpace va avea 30 deci vom putea introduce inca 30 de masini



Apare un mesaj că nu există locuri de parcare și Mașînă nu se adaugă in parcare

4.Manual de utilizare

* Descrierea interfetei produsului program

Interfața produsului nostru va fi o aplicație de parcare a mașinilor, care va oferi funcționalități pentru adăugarea mașinilor în parcarea noastră, obținerea biletelor de parcare și plata acestora.

* Care sunt datele de intrare si cum sunt introduse

La pornirea aplicației, utilizatorul va fi întâmpinat cu un meniu principal care va conține opțiuni pentru diferite acțiuni, cum ar fi adăugarea unei mașini în parcare si stergerea ei , dar si afisarea a primelor 15 masini din fisier si ieșirea din aplicație.

Pentru adăugarea unei mașini în parcare, utilizatorul va trebui să introducă numărul de înmatriculare al mașinii. Utilizatorul va fi informat cu privire la succesul sau eșecul adăugării mașinii în parcare, precum și orarul de intrare asociat acestei mașini.

Biletul de parcare se afiseaza pe ecran in momentul in care masina a fost adaugat cu success in parcare

Plata biletului de parcare se va face în momentul în care utilizatorul dorește să părăsească parcarea. Utilizatorul va trebui să introducă numărul de înmatriculare al mașinii dupa introducere se va afisa suma catre achitare și Utilizatorul va introduce suma pe care dorește să o plătească. Aplicația va verifica suma introdusă și va actualiza valoarea plății curente pentru mașina respectivă. De asemenea, aplicația va afișa valoarea totală a plății acumulate pentru mașina respective dar si restul pe care dispozitivul trebuie sa-l ofere.

În ceea ce privește operațiile ce pot fi efectuate, utilizatorul va putea adăuga mașini în parcare, obține biletul de parcare, plăti biletul și părăsi parcarea. De asemenea, utilizatorul va putea obține informații despre toate masinile parcate .

Exemple dar si explicatia lor le puteti gasi la capitolul 3.2 Studiu de caz

5.Concluzii

Este un proiect interesant și util pentru a rezolva problema parcărilor cu durată limitată și gestionarea eficientă a acestora. Implementarea unui sistem automatizat de gestionare a parcării poate reduce timpul de căutare a unui loc de parcare și poate încuraja respectarea limitelor de timp pentru a evita abuzurile și neregulile. Proiectul propus are potențialul de a rezolva problema parcărilor cu durată limitată și de a îmbunătăți gestionarea locurilor de parcare. Implementarea prin programare orientată pe obiecte aduce beneficii precum încapsularea, moștenirea, polimorfismul și abstracția.

Implementarea soluției de parcare descrisă prezintă mai multe avantaje și dezavantaje importante. Avantajele includ:

Autentificare pentru angajați: Implementarea autentificării utilizatorilor angajați prin intermediul unui cod unic aduce beneficii semnificative în ceea ce privește securitatea și controlul accesului în anumite zone sau facilități ale parcării. Prin verificarea codului unic introdus, sistemul poate confirma identitatea angajatului și poate permite accesul restricționat în zonele specifice doar pentru personalul autorizat.

Ușurință în utilizare: O interfață intuitivă și un meniu principal clar facilitează utilizarea aplicației de către utilizatorii, indiferent de nivelul lor de experiență. Instrucțiunile clare și structura logică a funcționalităților permit utilizatorilor să navigheze fără dificultate prin diferitele operațiuni disponibile și să efectueze acțiunile dorite cu ușurință.

Modularitate: Implementarea codului în fișiere separate pentru diferitele entități implicate în sistemul de parcare (cum ar fi utilizatorii, mașinile, tranzacțiile etc.) aduce un avantaj în ceea ce privește întreținerea, dezvoltarea și reutilizarea codului. Prin împărțirea codului în module bine definite, este mai ușor să se facă modificări sau adăugiri și să se mențină un nivel înalt de organizare și structură în aplicație.

Pe de altă parte, există și câteva dezavantaje ale implementării soluției: Necesitatea introducerii manuale a datelor: Utilizatorii trebuie să introducă manual numărul de înmatriculare al mașinii și alte informații necesare pentru operațiunile din aplicație. Acest proces poate fi supus erorilor umane și poate duce la introducerea incorectă a datelor sau la creșterea timpului necesar pentru efectuarea operațiunilor dar acest lucru poate fi ușor de rezolvat prin introducerea unui dispozitiv automat de citire a numărului de înmatriculare. Validarea insuficientă a intrărilor utilizatorului: Soluția nu efectuează o validare explicită a datelor introduse de utilizatori, cum ar fi verificarea corectitudinii numărului de înmatriculare sau a altor informații. Această poate duce la erori sau la comportamente neașteptate în cazul  în care utilizatorii introduc date incorecte sau incomplete. Lipsa gestionării excepțiilor: Implementarea nu include gestionarea adecvată a excepțiilor sau a situațiilor neașteptate. Acest lucru poate duce la comportamente nedorite sau la întreruperea programului în cazul unor erori sau condiții neașteptate, fără o gestionare corespunzătoare a acestora.

Direcții pentru continuarea proiectului:

Integrarea cu un sistem de plată online: Pentru a facilita procesul de plată și a oferi o mai mare flexibilitate utilizatorilor, proiectul poate fi extins pentru a permite integrarea cu un sistem de plată online, cum ar fi plata cu cardul.

Integrarea cu sisteme de recunoaștere a plăcilor de înmatriculare: Implementarea unui sistem de recunoaștere a plăcilor de înmatriculare poate permite automatizarea procesului de înregistrare a mașinilor în parcare. Aceasta ar reduce efortul utilizatorilor de a introduce manual numărul de înmatriculare și ar oferi o experiență mai rapidă și mai eficientă.

Implementarea unui sistem de gestionare a abonamentelor: Pentru utilizatorii frecvenți sau pentru firme care doresc să închirieze locuri de parcare pe termen lung, se poate implementa un sistem de gestionare a abonamentelor. Acesta ar permite utilizatorilor să achiziționeze abonamente lunare sau anuale și să beneficieze de parcare la tarife preferențiale.