Dokumentacija projekta

Kasa

1. Uvod

Cilj ove aplikacije je da simulira osnovnu funkcionalnost fiskalne kase u maloprodajnom objektu. Aplikacija omogućava unos artikala, formiranje računa na osnovu unetih artikala i količina, naplatu računa, kao i vođenje evidencije o realizovanim transakcijama.

Aplikacija je razvijena u programskom jeziku **C++**, koristeći **objektno-orijentisani pristup** kroz definisanje klasa kao što su Artikal, Stavka, Racun i Kasa.

Glavne funkcionalnosti koje aplikacija podržava uključuju:

- Unos i brisanje artikala iz baze proizvoda.
- Formiranje računa na osnovu unetih artikala.
- Prikaz iznosa za naplatu i izračunavanje povraćaja novca.
- Ispis svih računa sa sumama i PDV-om.
- Prikaz najvećeg i prosečnog računa.
- Dnevni izveštaj o poslovanju.

Aplikacija je zamišljena kao **konzolni program**, jednostavan za korišćenje, sa menijem koji korisnika vodi kroz sve raspoložive opcije.

2. Korišćene tehnologije i alati

Aplikacija je razvijena u programskom jeziku **C++**, koji je odabran zbog svoje **efikasnosti**, **fleksibilnosti** i **objektno-orijentisanog pristupa**. Korišćenje **C++ jezika** omogućilo je jasno definisanje **hijerarhije klasa**, **enkapsulaciju podataka** i organizaciju funkcionalnosti kroz metode i atribute.

Razvoj aplikacije odvijao se u okviru integrisanog razvojnog okruženja **Microsoft Visual Studio**, koje omogućava jednostavno **upravljanje projektom**, **organizaciju fajlova**, **praćenje grešaka** i izvođenje programa u **konzolnom režimu**. IDE je iskorišćen i za **debagovanje**, **formatiranje koda** i **testiranje** svih funkcionalnosti.

U okviru implementacije korišćene su standardne **C++ biblioteke** kao što su **<iostream>** i **<iomanip>**, koje omogućavaju **unos i prikaz podataka**, kao i njihovo **precizno formatiranje** (npr. za prikaz cena i količina na računu). Takođe je korišćena i biblioteka **<string>** za rad sa **tekstualnim podacima**.

Program je koncipiran prema principima **objektno-orijentisanog programiranja (OOP)**. Klase **Artikal**, **Stavka**, **Racun** i **Kasa** predstavljaju osnovne **apstrakcije** u sistemu, dok su odnosi među njima rešeni pomoću **nasleđivanja**, **kompozicije** i metoda koje omogućavaju **interakciju objekata**. Posebna pažnja posvećena je **enkapsulaciji podataka**, tako da su svi atributi **privatni** i dostupni isključivo putem **javnih metoda** (*get* i *set*).

Aplikacija **ne koristi baze podataka** ni **spoljne biblioteke**, već radi u potpunosti kao **konzolna aplikacija**, **samostalna** i **jednostavna za korišćenje i testiranje**.

3. Korisnički interfejs (UI)

Aplikacija **„Kasa“** koristi konzolni (tekstualni) korisnički interfejs, koji omogućava interaktivno korišćenje svih funkcionalnosti kroz jasno strukturisan meni i jednostavne korisničke instrukcije. Ovakav interfejs omogućava korisniku unos podataka putem tastature i dobijanje trenutnog povratnog odgovora aplikacije.

Nakon pokretanja programa, korisniku se prikazuje **glavni meni**, iz kojeg se biraju opcije putem unosa brojeva od 1 do 7.

```
****************************
Izaberite opciju:
1. Racun
2. Presek stanja
3. Dnevni izvestaj
4. Izmena artikla
5. Maksimalni racun
6. Prosek racuna
7. Izlaz
****************************
```

3.1. Glavni meni

Korisnik bira željenu opciju unosom odgovarajućeg broja. Svaka od opcija vodi ka odgovarajućoj funkcionalnosti aplikacije.

```
1
****************************
   :0: za kraj racuna
Unesite sifru artikla:
1
Unesite kolicinu:
2
Unesite sifru artikla:
3
Unesite kolicinu:
3
Unesite sifru artikla:
4
Unesite kolicinu:
4
Unesite sifru artikla:
0
****************************
****************************
Racun stampa:
      Hleb    2             62
    Jogurt    3            240
      Pivo    4            200
****************************
****************************
Izaberite opciju:
1. Naplata
2. Brisanje racuna
****************************
1
Za naplatu:               502
Iznos:
510
Povracaj:                   8
Racun je placen.
```

3.2. Kreiranje računa (Opcija 1 – Račun)

Korisnik započinje unos novog računa tako što redom unosi šifre artikala i količine. Svaki artikal se unosi pojedinačno, nakon čega se traži unos količine. Postupak se ponavlja dok korisnik ne unese šifru 0, čime se završava unos i prelazi na prikaz stavki na računu.

Nakon toga, prikazuje se štampana verzija računa sa listom stavki (naziv artikla, količina i ukupna cena).

Ako korisnik izabere opciju **"1. Naplata"**, unosi se ukupna suma, računa se kusur i potvrđuje da je račun plaćen. Ako izabere opciju **"2. Brisanje racuna".**

```
2
****************************
Presek stanja:
Racun   1                 502
Racun   2                  31
Zaradjeno:                533
PDV:                      107
****************************
```

3.3. Presek stanja (Opcija 2)

Daje **sveobuhvatan prikaz podataka o prometu do određenog trenutka** u toku dana:

Listu izdatih računa **Ukupan porez (PDV)**
Ukupan prihod

```
3
*****************************
Dnevni izvestaj:
Racun   1                  502
Racun   2                   31
Zaradjeno:                 533
PDV:                       107
*****************************
```

3.4. Dnevni izveštaj (Opcija 3)

Daje **sveobuhvatan prikaz podataka o prometu u toku dana** u toku dana: **Listu izdatih računa, Ukupan porez (PDV), Ukupan prihod.** Potom **izlazi iz kase.**

```
4
*****************************
Izaberite opciju:
1. Unos novog artikla
2. Brisanje artikla
3. Ispis artikla
```

3.5. Izmena artikala (Opcija 4)

Opcija **"Izmena artikla"** omogućava korisniku potpunu kontrolu nad bazom artikala koji se nalaze u prodavnici. Nakon izbora ove opcije iz glavnog menija, korisniku se prikazuje podmeni.

```
4
*****************************
Izaberite opciju:
1. Unos novog artikla
2. Brisanje artikla
3. Ispis artikla
*****************************
1
Unesite sifru artikla:
10
Unesite naziv artikla:
Pivo
Unesite cenu artikla:
100
Unesite tip artikla:
1. Mlecni Proizvod
2. Meso
3. Voce
4. Pecivo
5. Pice
6. Hrana
5
Artikal je dodat.
```

Korisnik **unosi novi artikal** u sistem tako što redom popunjava sledeće podatke:

- **Šifra artikla** (jedinstveni broj, npr. 10)
- **Naziv artikla** (npr. "Pivo")
- **Cena artikla** (npr. 120)
- **Tip artikla** – predstavlja kategoriju kojoj artikal pripada. Tipovi su predstavljeni brojevima:

1 – Mlečni proizvodi	4 – Pecivo
2 – Piće	5 – Meso
3 – Hleb i pekara	6 – Hrana (ostalo)

```
4
*****************************
Izaberite opciju:
1. Unos novog artikla
2. Brisanje artikla
3. Ispis artikla
*****************************
2
Unesite sifru artikla za brisanje:10
Artikal je obrisan.
```

Sistem potvrđuje da je novi artikal uspešno dodat u bazu. Zatim vraća korisnika na glavni meni.

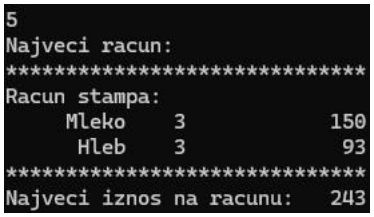
Brisanje artikla se vrši unosom šifre artikla koji korisnik želi da ukloni. Zatim vraća korisnika na glavni meni.

```
4
*****************************
Izaberite opciju:
1. Unos novog artikla
2. Brisanje artikla
3. Ispis artikla
*****************************
3
*****************************
Artikli u bazi:
Sifra|    Naziv|   Cena|   Tip
    1      Hleb      31      3
    2     Mleko      50      0
    3    Jogurt      80      0
    4      Pivo      50      4
    5    Banane     100      2
    6       Jaja    1.12      2
   10      Pivo     100      4
*****************************
```

Ispis svih artikala prikazuje tabelarni prikaz svih trenutno dostupnih artikala u bazi. Svaki red sadrži:

- Šifru artikla
- Naziv artikla
- Jediničnu cenu
- Tip proizvoda (broj koji označava kategoriju)

Zatim vraća korisnika na glavni meni.



3.6 Maksimalni račun (Opcija 5)

Prikazuje račun sa **najvećom ukupnom vrednošću** iz svih izdatih računa do tog trenutka.

3.7 Prosečna vrednost računa (Opcija 6)

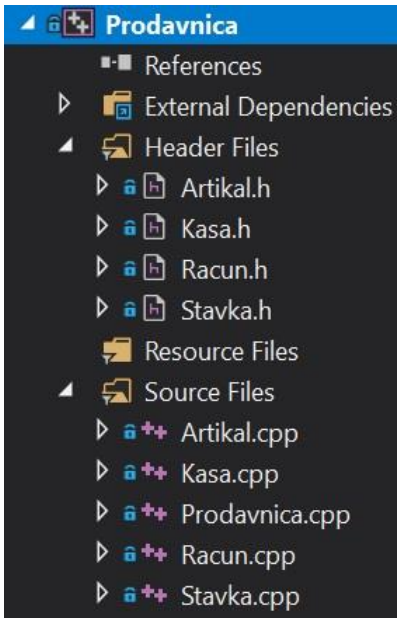
Računa i prikazuje prosečnu vrednost svih do tada napravljenih računa.

3.8. Izlaz (Opcija 7)

Izlazi iz aplikacije i zatvara program.

4. Struktura projekta

4.1. Organizacija fajlova i foldera



Projekat je razvijen kao **C++ konzolna aplikacija.** Svaka klasa je podeljena u dva fajla – **header fajl (.h)** koji sadrži deklaraciju klase i **implementacioni fajl (.cpp)** koji sadrži implementaciju metoda. Time je omogućeno **jasno odvajanje interfejsa i logike**, kao i bolja čitljivost i modularnost koda.

Header Files – sadrže deklaracije klasa:

Artikal.h **Racun.h**

Kasa.h **Stavka.h**

Source Files – sadrže implementacije klasa i logiku aplikacije:

Artikal.cpp **Kasa.cpp**

Racun.cpp **Stavka.cpp**

Prodavnica.cpp – **glavni fajl** koji sadrži funkciju main() i korisnički interfejs aplikacije.

4.2. Glavne klase i njihova uloga

Klasa **Artikal** predstavlja osnovni entitet – proizvod koji se može naći na računu. Svaki artikal ima sledeće atribute:

Šifra (int) **Naziv (string)**

Cena (double) **Tip (enum TipArtikla)**

Klasa sadrži metode za pristup i izmenu ovih vrednosti, kao i ispis podataka.

Primer: Artikal.cpp

```cpp
#include "Artikal.h"
#include <iostream>
#include <iomanip>

Artikal::Artikal(int sifra, string naziv,
    double cena, TipArtikla tip) {
    this->sifra = sifra;
    this->naziv = naziv;
    this->cena = cena;
    this->tip = tip;
}

int Artikal::getSifra() {
    return this->sifra;
}
string Artikal::getNaziv() {
    return this->naziv;
}
double Artikal::getCena() {
    return this->cena;
}
TipArtikla Artikal::getTip() {
    return this->tip;
}
void Artikal::setSifra(int sifra) {
    this->sifra = sifra;
}
void Artikal::setNaziv(string naziv) {
    this->naziv = naziv;
}
void Artikal::setCena(double cena) {
    this->cena = cena;
}
void Artikal::setTip(TipArtikla tip) {
    this->tip = tip;
}
void Artikal::ispisi() {
    cout << setw(5) << this->getSifra() << setw(10)
        << this->naziv << setw(8) << setprecision(3)
        << this->getCena() << setw(7) << this->getTip ()<< endl;
}
bool Artikal::isEqual(int sifra) {
    if (sifra == this->getSifra()) {
        return true;
    }
    else {
        return false;
    }
}

string Artikal::toString() {
    char nizChar[31];
    int n;
    n = sprintf_s(nizChar, "%2d. %-15s%11.2f", this->getSifra(),
        this->getNaziv().c_str(), this->getCena());
    string s = nizChar;
    return s;
}
```

Primer: Artikal.h

```
#pragma once
#include <string>
using namespace std;
enum TipArtikla { MlecniProizvod, Meso, Voce, Pecivo, Pice, Hrana };

class Artikal
{
    private:
        int sifra;
        string naziv;
        double cena;
        TipArtikla tip;

    public:
        Artikal() {};
        Artikal(int sifra,  string naziv, double cena,    TipArtikla tip);
        void ispisi();


        void setSifra(int sifra);
        int getSifra();
        void setNaziv(string naziv);
        string getNaziv();
        void setCena(double cena);
        double getCena();
        void setTip(TipArtikla tip);
        TipArtikla getTip();
        bool isEqual(int sifra);
        string toString();
};
```

Klasa **Stavka** je izvedena iz klase Artikal i predstavlja konkretan unos na računu. Dodaje dodatni atribut – **količinu**, i metode koje omogućavaju:

- Izračunavanje cene stavke (cenaStavke)
- Poređenje artikala po šifri (isEqual)
- Ispis i formatiranje (ispisi, toString)

Primer: Stavka.h

```
#pragma once
#include "Artikal.h"
class Stavka :
    public Artikal
{
private:
    double kolicina;

public:
    Stavka() {};
    Stavka(Artikal artikal, double kolicna);

    double getKolicina();
    void setKolicina(double kolicina);
    void ispisi();
    bool isEqual(Stavka s);
    double cenaStavke();
    string toString();
};
```

Primer: Stavka.cpp

```cpp
#include "Stavka.h"
#include <iomanip>
#include <iostream>

Stavka::Stavka(Artikal artikal, double kolicina) :
    Artikal(artikal.getSifra(), artikal.getNaziv(),
        artikal.getCena(), artikal.getTip()) {
    this->kolicina = kolicina;
}

void Stavka::setKolicina(double kolicina) {
    this->kolicina = kolicina;
}

double Stavka::getKolicina() {
    return this->kolicina;
}

void Stavka::ispisi() {
    cout << setw(10) << this->getNaziv() << setw(5)
        << setprecision(3) << this->getKolicina()
        << setw(15) << setprecision(3) << this->cenaStavke()
        << endl ;
}

double Stavka::cenaStavke() {
    return this->getCena() * this->getKolicina();
}

bool Stavka::isEqual(Stavka s) {
    if (this->getSifra() == s.getSifra()) {
        return true;
    } else {
        return false;
    }
}
string Stavka::toString() {
    char nizChar[31];
    int n;
    n = sprintf_s(nizChar, "%2d. %-15s%11.2f",this->getSifra(),
        this->getNaziv().c_str(), this->cenaStavke());
    string s = nizChar;
    return s;
}
```

Klasa **Racun** predstavlja listu stavki i omogućava rad sa jednim računom. Sadrži:

- Niz stavki (maksimalno 100)
- Brojač stavki
- Metode za dodavanje i brisanje stavki, ispis, izračunavanje ukupnog iznosa i PDV-a, kao i poređenje (toCompare)

Klasa **Kasa** upravlja svim računima i artiklima. Čuva niz svih artikala i svih računa (maksimalno po 100), kao i metode za:

- Naplatu (platiRacun)
- Dnevni izveštaj (dnevniIzvestaj)
- Presek stanja (presekStanja)
- Dodavanje i brisanje artikala i računa
- Analitiku: najveći i prosečan račun

4.3 Glavni program

Fajl **Prodavnica.cpp** predstavlja **ulaznu tačku aplikacije** i sadrži funkciju main(), kao i funkcionalnosti korisničkog interfejsa. Kroz jednostavan tekstualni meni, korisniku se omogućava interakcija sa aplikacijom putem sledećih opcija:

- **Kreiranje novog računa** – korisnik unosi šifre i količine artikala, nakon čega se formira račun sa stavkama. Račun se može naplatiti ili obrisati pre finalizacije.
- **Prikaz preseka stanja** – omogućava uvid u sve kreirane račune, njihov ukupan iznos i zbirnu zaradu.
- **Dnevni izveštaj** – prikazuje sve račune u toku dana, kao i ukupnu zaradu i iznos PDV-a.
- **Upravljanje artiklima** – korisnik može da doda novi artikal, obriše postojeći ili prikaže sve artikle koji su uneti u sistem.
- **Statistički podaci** – prikaz najvećeg pojedinačnog računa i izračunavanje prosečne vrednosti svih računa.

Logika korisničkog interfejsa je razdvojena u posebne funkcije:

- racunF(Kasa* k) – obrada i kreiranje računa,
- izmenaArtikla(Kasa* k) – dodavanje i brisanje artikala,
- max(Kasa* k) – prikaz najvećeg računa,
- prosek(Kasa* k) – prikaz prosečne vrednosti računa.

Ova podela doprinosi **boljoj organizaciji koda**, olakšava buduće izmene i unapređenja, i jasno razdvaja poslovnu logiku od korisničkog interfejsa.

Primer max(Kasa* k) i prosek(Kasa* k) funkcija iz Prodavnica.cpp

```
void max(Kasa*  k) {
    Racun r = k->najveciRacun();
    double d = r.zaNaplatu();
    r.ispisi();
    cout << "Najveci iznos na racunu:" << setw(6) << setprecision(3) << d << endl;
}

void prosek(Kasa* k) {
    double r = k->prosecanRacun();
    cout << "Prosek racuna:" << setw(16) << setprecision(3) << r << endl;

}
```

Primer main() iz Prodavnica.cpp

```cpp
#include "Artikal.h"
#include <iostream>
#include <iomanip>

Artikal::Artikal(int sifra, string naziv,
    double cena, TipArtikla tip) {
    this->sifra = sifra;
    this->naziv = naziv;
    this->cena = cena;
    this->tip = tip;
}

int main()
{
    Kasa* k = new Kasa();
    Racun r;
    Artikal a1 = Artikal(1, "Hleb", 31.0, Pecivo);
    Artikal a2 = Artikal(2, "Mleko", 50.0, MlecniProizvod)
    Artikal a3 = Artikal(3, "Jogurt", 80.0, MlecniProizvod
    Artikal a4 = Artikal(4, "Pivo", 50.0, Pice);
    Artikal a5 = Artikal(5, "Banane", 100.0, Voce);
    Artikal a6 = Artikal(6, "Jaja", 1.1234, Voce);


    k->dodajArtikal(a1);
    k->dodajArtikal(a2);
    k->dodajArtikal(a3);
    k->dodajArtikal(a4);
    k->dodajArtikal(a5);
    k->dodajArtikal(a6);

    Stavka s1 = Stavka(a1, 5.0);

    while (true) {
        cout << "*****************************" << endl;
        cout << "Izaberite opciju:" << endl;
        cout << "1. Racun" << endl;
        cout << "2. Presek stanja" << endl;
        cout << "3. Dnevni izvestaj" << endl;
        cout << "4. Izmena artikla" << endl;
        cout << "5. Maksimalni racun" << endl;
        cout << "6. Prosek racuna" << endl;
        cout << "7. Izlaz" << endl;
        cout << "*****************************" << endl;
        int c;
        cin >> c;

        switch (c) {
        case 1:
            racunF(k);
            break;
        case 2:
            k->presekStanja();
            break;
        case 3:
            k->dnevniIzvestaj();
            return 0;
            break;
        case 4:
            izmenaArtikla(k);
            break;
        case 5:
            max(k);
            break;
        case 6:
            prosek(k);
            break;
        case 7:
            return 0;
        }
    }
}
```

5. Zaključak

Projekat „**Kasa**“ predstavlja jednostavnu, ali funkcionalno bogatu **C++ konzolnu aplikaciju** namenjenu simulaciji rada kase u maloprodajnom objektu. Kroz modularnu strukturu, aplikacija omogućava efikasno **upravljanje artiklima**, kreiranje i naplatu računa, kao i generisanje korisnih izveštaja poput preseka stanja, dnevnog izveštaja i statistike.

Korišćenjem **objektno-orijentisanog pristupa**, svaki entitet (artikal, stavka, račun, kasa) je jasno definisan kroz posebnu klasu, što značajno povećava čitljivost i održavanje koda. Klasa Kasa centralizuje rad sa artiklima i računima, dok Prodavnica.cpp omogućava korisniku jednostavan način interakcije sa sistemom putem menija.

Ovaj projekat je odlična osnova za dalje proširenje – kao što su **čuvanje podataka u fajl**, **grafički interfejs**, **povezivanje sa bazom podataka**, ili čak integraciju sa fiskalnim sistemima. Takođe, služi kao praktičan primer za učenje i demonstraciju ključnih principa **OOP-a u C++ jeziku**.