Sprawozdanie

Temat pracy: Sklep Wędkarski od strony sprzedawcy
Autorzy: Piotr Jaglarz, Mateusz Kupczyk, Leon Wałach

1. Wstęp

Projekt sklepu wędkarskiego “Barwena” napisanego w języku c++. Aplikacje umożliwia kupowanie różnych produktów wędkarskich z różnych kategorii, tj. Wędki, Kołowrotki, Przynęty, Spławiki, Żyłki i plecionki oraz Akcesoria. Zakupy są przechowywane w koszyku, a po dokonaniu zakupu otrzymujemy raport.

2. Opis implementacji

W tym punkcie opiszemy poszczególne części naszego kodu:

StronaStartowa()

Ta funkcja wyświetla ekran powitalny sklepu. Za pomocą sleep(2000) zatrzymujemy program na 2000ms, a dzięki system("cls") możemy wyczyścić ekran dzięki czemu zniknie nam ekran powitalny.

```
void StronaStartowa() {
    cout << "*****************************************************************" << endl;
    cout << "*                  Sklep Wedkarski \"Barwena\"                   *" << endl;
    cout << "*              Projekt obiektowej aplikacji C++                 *" << endl;
    cout << "*                                                               *" << endl;
    cout << "*****************************************************************" << endl;


    Sleep(2000);
    system("cls");
}
```

wyswietl_kategorie()

Ta funkcja wyświetla kategorie znajdujące się w sklepie

```
void wyswietl_kategorie(){
    cout<<"Wybierz kategorie produktu: "<<endl;
    cout<<"1. Wedki"<<endl;
    cout<<"2. Kolowrotki"<<endl;
    cout<<"3. Przynety"<<endl;
    cout<<"4. Splawiki"<<endl;
    cout<<"5. Zylki i plecionki"<<endl;
    cout<<"6. Akcesoria"<<endl<<"Wybor: "<<endl;


}
```

class Produkt

To jest klasa abstrakcyjna, ma dwie wirtualne metody wyswietl() i pobierzCene(). Klasy które dziedziczą po tej funkcji muszą zawierać te metody.

```
class Produkt{
public:
    virtual void wyswietl()= 0;
    virtual double pobierzCene() const = 0;
};
```

class SklepWedkarski

Ta klasa ma metodę pokaz(), która przyjmuje wskaźnik do Klasy Produkt i wywołuje metodę wyswietl(). Jest to przykład polimorfizmu.

```
class SklepWedkarski {
public:
    void pokaz(Produkt* produkt){
        produkt->wyswietl();
    }

};
```

class Wedka

Klasa Wedka dziedziczy publicznie po klasie Produkt
Wedka(string n, double c, string p) inicjalizuje obiekt wedki przy użyciu podanych wartości.
Wedka() ustawia domyślne wartości, jeśli nie są dostarczone żadne inne.
wyswietl(): Wyświetla informacje o wedce, w tym nazwę, cenę i długość. ustaw():Ustawia parametry wedki na podstawie dostarczonych wartości.
pobierzCene():Zwraca cenę wedki.

```
class Wedka :public Produkt{
public:

    Wedka(string n, double c, string p): nazwa(n), cena(c), parametr(p) {}
    Wedka() : nazwa("brak"), cena(0.0), parametr("brak") {}

    void wyswietl(){
        cout<<"Nazwa: "<<nazwa<<" Cena"<<cena<<" Dlugosc:"<<parametr<<" m"<<endl;
    }
    string nazwa;
    double cena;
    string parametr;

        void ustaw(string naz, double cen, string par)
    {
        nazwa=naz;
        cena=cen;
        parametr=par;
    }

     double pobierzCene() const override {
        return cena;
    }
};
```

class Kolowrotek

Klasa Kolowrotek dziedziczy publicznie po klasie Produkt

Kolowrotek(string n, double c, string p) inicjalizuje obiekt kolowrotka przy użyciu podanych wartości.

Kolowrotek() ustawia domyślne wartości, jeśli nie są dostarczone żadne inne.

wyswietl(): Wyświetla informacje o kołowrotku, w tym nazwę, cenę i długość.

ustaw():Ustawia parametry kołowrotka na podstawie dostarczonych wartości.

pobierzCene():Zwraca cenę kołowrotka.

```
class Kolowrotek :public Produkt{
public:

    string nazwa;
    double cena;
    string parametr;

    Kolowrotek(string n, double c, string p): nazwa(n), cena(c), parametr(p) {}
    Kolowrotek() : nazwa("brak"), cena(0.0), parametr("brak") {}
    void wyswietl(){
        cout<<"Nazwa: "<<nazwa<<" Cena: "<<cena<<" "<<endl;
    }


        void ustaw(string naz, double cen, string par)
    {
        nazwa=naz;
        cena=cen;
        parametr=par;
    }

     double pobierzCene() const override {
        return cena;
    }
};
```

class Przyneta

Klasa Przyenta dziedziczy publicznie po klasie Produkt

Przyneta(string n, double c, string p) inicjalizuje obiekt przynęty przy użyciu podanych wartości.

Przyneta() ustawia domyślne wartości, jeśli nie są dostarczone żadne inne.

wyswietl():Wyświetla informacje o przynęcie, w tym nazwę, cenę i kolor.

ustaw():Ustawia parametry przynęty na podstawie dostarczonych wartości.

pobierzCene():Zwraca cene przynety.

```
class Przyneta :public Produkt{
public:

    string nazwa;
    double cena;
    string parametr;


    Przyneta(string n, double c, string p): nazwa(n), cena(c), parametr(p) {}
    Przyneta() : nazwa("brak"), cena(0.0), parametr("brak") {}
    void wyswietl(){
        cout<<"Nazwa: "<<nazwa<<" Cena: "<<cena<<" Kolor: "<<parametr<<endl;
    }

        void ustaw(string naz, double cen, string par)
    {
        nazwa=naz;
        cena=cen;
        parametr=par;
    }

     double pobierzCene() const override {
        return cena;
    }
};
```

class Splawik

Klasa Splawik dziedziczy publicznie po klasie Produkt

Splawik(string n, double c, string p) inicjalizuje obiekt spławika przy użyciu podanych wartości.

Splawik() ustawia domyślne wartości, jeśli nie są dostarczone żadne inne.

wyswietl(): Wyświetla informacje o spławika, w tym nazwę, cenę i długość.

ustaw():Ustawia parametry spławika na podstawie dostarczonych wartości.

pobierzCene():Zwraca cenę spławika.

```
class Splawik :public Produkt{
public:

    string nazwa;
    double cena;
    string parametr;

    Splawik(string n, double c, string p): nazwa(n), cena(c), parametr(p) {}
    Splawik() : nazwa("brak"), cena(0.0), parametr("brak") {}
    void wyswietl(){
        cout<<"Nazwa: "<<nazwa<<" Cena: "<<cena<<" Rodzaj: "<<parametr<<endl;
    }


        void ustaw(string naz, double cen, string par)
    {
        nazwa=naz;
        cena=cen;
        parametr=par;
    }

     double pobierzCene() const override {
        return cena;
    }
};
```

class Zylka

Klasa Zylka dziedziczy publicznie po klasie Produkt

Zylka(string n, double c, string p) inicjalizuje obiekt żyłki przy użyciu podanych wartości.

Zylka() ustawia domyślne wartości, jeśli nie są dostarczone żadne inne.

wyswietl(): Wyświetla informacje o żyłki, w tym nazwę, cenę i grubość.

ustaw():Ustawia parametry spławika na podstawie dostarczonych wartości.

pobierzCene():Zwraca cenę żyłki.

```
class Zylka :public Produkt{
public:

    string nazwa;
    double cena;
    string parametr;

    Zylka(string n, double c, string p): nazwa(n), cena(c), parametr(p) {}
    Zylka() : nazwa("brak"), cena(0.0), parametr("brak") {}
    void wyswietl(){
        cout<<"Nazwa: "<<nazwa<<" Cena: "<<cena<<" Grubosc: "<<parametr<<" mm"<<endl;
    }

        void ustaw(string naz, double cen, string par)
    {
        nazwa=naz;
        cena=cen;
        parametr=par;
    }

     double pobierzCene() const override {
        return cena;
    }
};
```

class Akcesoria

Klasa Akcesoria dziedziczy publicznie po klasie Produkt

Akcesoria(string n, double c, string p) inicjalizuje obiekt akcesorium przy użyciu podanych wartości.

Akcesoria() ustawia domyślne wartości, jeśli nie są dostarczone żadne inne.

wyswietl(): Wyświetla informacje o akcesorium, w tym nazwę, cenę i dodatkowy parametr.

ustaw():Ustawia parametry akcesorium na podstawie dostarczonych wartości.

pobierzCene(): Zwraca cenę akcesorium.

```
class Akcesoria :public Produkt{
public:
    string nazwa;
    double cena;
    string parametr;



    Akcesoria(string n, double c, string p): nazwa(n), cena(c), parametr(p) {}
    Akcesoria() : nazwa("brak"), cena(0.0), parametr("brak") {}

    void wyswietl(){
        cout<<"Nazwa: "<<nazwa<<" Cena: "<<cena<<" "<<endl;
    }

    void ustaw(string naz, double cen, string par)
    {
        nazwa=naz;
        cena=cen;
        parametr=par;
    }
     double pobierzCene() const override {
        return cena;
    }

};
```

Class Kup

- class Koszyk

klasa reprezentująca koszyk zakupów.

vector<Produkt*> produkty: Wektor przechowujący wskaźniki do produktów w koszyku.

dodajProdukt(Produkt* produkt): Metoda dodająca produkt do koszyka.

wyczyscKoszyk(): Metoda usuwająca wszystkie produkty z koszyka.

```
class Kup {
private:
    class Koszyk {
    public:
        vector<Produkt*> produkty;

        void dodajProdukt(Produkt* produkt) {
            produkty.push_back(produkt);
        }

        void wyczyscKoszyk() {
            produkty.clear();
        }
    };
```

- koszykowyKoszt()

Oblicza i zwraca łączny koszt zakupów w koszyku.

```
double koszykowyKoszt() const {
    double koszt = 0.0;
    for (Produkt* produkt : koszyk.produkty) {
        koszt += produkt->pobierzCene();
    }
    return koszt;
}
```

- czasAktualny()

Zwraca bieżącą datę i czas w formie ciągu znaków.

```
string czasAktualny() const {
    time_t teraz = time(0);
    tm* czasTeraz = localtime(&teraz);
    char czasChar[80];
    strftime(czasChar, 80, "%Y-%m-%d %H:%M:%S", czasTeraz);
    return czasChar;
}
```

- dodajProduktDoKoszyka(Produkt* produkt)

Dodaje produkt do koszyka

```
public:


    void dodajProduktDoKoszyka(Produkt* produkt) {
        koszyk.dodajProdukt(produkt);
    }
```

- dokonajZakupu()

Dokonuje zakupu, generuje raport, wyświetla zakupione produkty i czyści koszyk.

```
void dokonajZakupu() {
    ofstream raport("raport.txt", ios::app);



    raport << "Data zakupu: " << czasAktualny() << endl;

    raport << "Koszt ca³kowity zakupow: " << koszykowyKoszt() << " PLN" << endl;
    raport << "-----------------------------------------------" << endl;



    cout<<"Kupione produkty: "<<endl;
    for (Produkt* produkt : koszyk.produkty) {
        produkt->wyswietl();
        cout<< endl;
    }


    cout<<endl<<endl<<"Za zakupy zaplacisz: "<<koszykowyKoszt()<<endl;

    raport.close();
    koszyk.wyczyscKoszyk();
}
};
```

Main()

- tworzenie obiektów

```
SklepWedkarski sklep;
Kup kupowanie;
```

- Tablice przechowujące produkty różnych kategorii

```
Wedka wedki[5]; int wedkii=0;
Kolowrotek kolowrotki[5]; int kolowrotkii=0;
Przyneta przynety[7]; int przynetyi=0;
Splawik splawiki[6]; int splawikii=0;
Zylka zylki[6]; int zylkii=0;
Akcesoria akcesor[9]; int akcesori=0;
```

- Otwarcie pliku produkty.txt w trybie do odczytu, odczyt danych i inicjalizacja tablic produktów

```
ifstream plik_wejsciowy("produkty.txt");

while (plik_wejsciowy >> n >> c >> p >> kategoria)
 {

    //cout<<n<<" "<<c<<" "<<p<<" "<<kategoria<<endl;

    switch (kategoria) {
case 1:
    wedki[wedkii].ustaw(n, c, p);
    wedkii++;
    break;

case 2:
    kolowrotki[kolowrotkii].ustaw(n, c, p);
    kolowrotkii++;
    break;

case 3:
    przynety[przynetyi].ustaw(n, c, p);
    przynetyi++;
    break;

case 4:
    splawiki[splawikii].ustaw(n, c, p);
    splawikii++;
    break;

case 5:
    zylki[zylkii].ustaw(n, c, p);
    zylkii++;
    break;

case 6:
    akcesor[akcesori].ustaw(n, c, p);
    akcesori++;
    break;

default:
    // Obs³uga sytuacji, gdy kategoria jest nieznana
    break;
}

 }
```

- Pętla obsługująca interakcję z użytkownikiem. Wyświetla dostępne kategorie produktów. Pozwala użytkownikowi wybrać kategorię i dodawać produkty do koszyka. Pyta użytkownika, czy chce kontynuować zakupy.

```
while(!koniec_zakupow)
{

wyswietl_kategorie();

cin>>kategoria;

system("cls");


if (kategoria == 1) {

        for(int i=0;i<=(wedkii-1);i++)
        {
            cout<<" "<< i<<".  ";
            wedki[i].wyswietl();
        }

        int wybieranie=-1;
        cout<<"Ktory produkt dodac do koszyka: "<<endl;
        cin>>wybieranie;
        kupowanie.dodajProduktDoKoszyka(&wedki[wybieranie]);

    } else if (kategoria == 2) {

        for(int i=0;i<=(kolowrotkii-1);i++)
        {
            cout<<" " <<i<<".  ";
            kolowrotki[i].wyswietl();

        }

        int wybieranie=-1;
        cout<<"Ktory produkt dodac do koszyka: "<<endl;
        cin>>wybieranie;
        kupowanie.dodajProduktDoKoszyka(&kolowrotki[wybieranie]);

    } else if (kategoria == 3) {

    for(int i=0;i<=(przynetyi-1);i++)
        {
            cout<<" "<< i<<".  ";
            przynety[i].wyswietl();
        }

        int wybieranie=-1;
        cout<<"Ktory produkt dodac do koszyka: "<<endl;
        cin>>wybieranie;
        kupowanie.dodajProduktDoKoszyka(&przynety[wybieranie]);

    } else if (kategoria == 4) {

        for(int i=0;i<=(splawikii-1);i++)
        {
            cout<<" "<< i<<".  ";
            splawiki[i].wyswietl();
        }
```

```
    } else if (kategoria == 5) {

      for(int i=0;i<=(zylkii-1);i++)
        {
            cout<<" "<< i<<".  ";
            zylki[i].wyswietl();
        }

        int wybieranie=-1;
        cout<<"Ktory produkt dodac do koszyka: "<<endl;
        cin>>wybieranie;
        kupowanie.dodajProduktDoKoszyka(&zylki[wybieranie]);

    } else if (kategoria == 6) {

        for(int i=0;i<=(akcesori-1);i++)
        {
            cout<<" "<< i<<".  ";
            akcesor[i].wyswietl();
        }

        int wybieranie=-1;
        cout<<"Ktory produkt dodac do koszyka: "<<endl;
        cin>>wybieranie;
        kupowanie.dodajProduktDoKoszyka(&akcesor[wybieranie]);
    }


cout<<"Jesli chcesz zakonczyc zakupy wybierz 1, aby kontynuowac wybierz 0"<<endl;
cin>>koniec_zakupow;

system("cls");
}
```

- Dokonuje zakupów na podstawie produktów w koszyku

```
kupowanie.dokonajZakupu();
```

3. Przykładowe działanie programu

- strona startowa

```
************************************************************
*                  Sklep Wedkarski "Barwena"               *
*              Projekt obiektowej aplikacji C++            *
*
************************************************************
```

- Wybór kategorii

```
Wybierz kategorie produktu:
1. Wedki
2. Kolowrotki
3. Przynety
4. Splawiki
5. Zylki i plecionki
6. Akcesoria
Wybor:
1
```

- Wybór produktu

```
 0.  Nazwa: Shimano Cena350 Dlugosc:3.66 m
 1.  Nazwa: Daiwa Cena519 Dlugosc:3 m
 2.  Nazwa: Mikado Cena207 Dlugosc:3.96 m
 3.  Nazwa: Jaxon Cena380 Dlugosc:3.5 m
 4.  Nazwa: Madcat Cena550 Dlugosc:3.23 m
Ktory prokdukt dodac do koszyka:
3
```

- koniec lub dalsze zakupy

```
Jesli chcesz zakonczyc zakupy wybierz 1, aby kontynuowac wybierz 0
0
```

- dalsze zakupy

```
 0.  Nazwa: Robaki-1l Cena: 25 Kolor: czerwone
 1.  Nazwa: Robaki-1l Cena: 23 Kolor: biale
 2.  Nazwa: Mucha Cena: 60 Kolor: czerwona
 3.  Nazwa: Wobler Cena: 30 Kolor: zielony
 4.  Nazwa: Blystka Cena: 25 Kolor: zlota
 5.  Nazwa: Mormyszka Cena: 15 Kolor: srebrna
 6.  Nazwa: Guma Cena: 7 Kolor: zolta
Ktory prokdukt dodac do koszyka:
5
Jesli chcesz zakonczyc zakupy wybierz 1, aby kontynuowac wybierz 0
1
```

- zakończenie zakupów

```
Kupione produkty:
Nazwa: Jaxon Cena380 Dlugosc:3.5 m

Nazwa: Mormyszka Cena: 15 Kolor: srebrna



Za zakupy zaplacisz: 395
```

4. Wnioski

Projekt sklepu wędkarskiego umożliwiająca zakupy produktów wędkarskich wykorzystuje obiektowość dzięki czemu program jest bardziej czytelny, a implementacja różnych klas dla poszczególnych kategorii produktów daje nam większą rozszerzalność programu