



Универзитет „Св. Кирил и Методиј“ - Скопје
**ФАКУЛТЕТ ЗА ИНФОРМАТИЧКИ НАУКИ
И КОМПЈУТЕРСКО ИНЖЕНЕРСТВО**

Објектно ориентирано програмирање

Аудиториски вежби 7

Содржина

1. Наследување	1
1.1. Задача	1
1.2. Задача	2
2. Изворен код од примери и задачи	5

1. Наследување

1.1. Задача

Да се дефинира класа за репрезентација на тениски играч. За секој тениски играч треба да се чуваат името, презимето како и тоа дали игра во лига (bool).

Од класата тениски играч да се изведе класа за рангиран тениски играч, која ќе репрезентира играч кој игра на меѓународно ниво. За рангираните тениски играчи дополнително треба да се чува и рангот на тенискиот играч.

Решение oop_av71.cpp

```
#include <iostream>
#include <cstring>

using namespace std;

class TennisPlayer {
private:
    enum {STRMAX = 50};
    char name[STRMAX];
    char surname[STRMAX];
    bool playsInLeague;

public:
    TennisPlayer(const char* fn, const char* sn, bool pl);

    void Print() const;

    bool PlaysInLeague();

    void setPlaysInLeague(bool pl);

    ~TennisPlayer();

    friend ostream& operator<<(ostream &o, const TennisPlayer &tp);
};

ostream& operator<<(ostream &o, const TennisPlayer &tp){
    o << "-- Tennis Player --\n"
      << tp.name << " "
      << tp.surname << " - " << tp.playsInLeague << endl;
    return o;
}

TennisPlayer::TennisPlayer(const char* fn="", const char* sn="", bool pl = false) {
    cout << "Constructor:" << endl;
    strncpy(name, fn, STRMAX-1);
    name[STRMAX-1] = '\0';
    strncpy(surname, sn, STRMAX-1);
    surname[STRMAX-1] = '\0';
    playsInLeague = pl;
}

void TennisPlayer::Print() const {
    cout << surname << ", " << name;
}

void TennisPlayer::setPlaysInLeague(bool pl){
    this->playsInLeague = pl;
}
```

```

bool TennisPlayer::PlaysInLeague(){
    return this->playsInLeague;
}

TennisPlayer::~TennisPlayer() {
    cout << "Destructor TennisPlayer for: " << this->name << " " << this->surname << endl;
}

class RankedTennisPlayer : public TennisPlayer {
private:
    unsigned int rank;
public:
    RankedTennisPlayer(const char* n, const char* sn, bool pl = false, int r = 0)
        : TennisPlayer(n,sn, pl)
    {
        cout << "Constructor RankedTennisPlayer" << endl;
        this->rank = r;
    }

    RankedTennisPlayer(const TennisPlayer& t, unsigned int r=0) : TennisPlayer(t){
        this->rank = r;
    }

    ~RankedTennisPlayer() {
        cout << "Destructor RankedTennisPlayer\n" << endl;
    }

    friend ostream& operator<<(ostream& out, const RankedTennisPlayer &tp) {
        cout << "-- RANKED TENNIS PLAYER --\n";
        out << TennisPlayer (tp);
        out << "Rank: " << tp.rank << endl;
        return out;
    }
};

int main() {
    TennisPlayer rf("Roger", "Federer");
    TennisPlayer ng("Novak", "Djokovikj");
    cout << rf;
    cout << ng;
    //TennisPlayer t;
    RankedTennisPlayer rn("Rafael", "Nadal", true, 2750);
    cout << rn;
    TennisPlayer tp = rn;
    cout << tp;
    //RankedTennisPlayer copy(tp);
    RankedTennisPlayer copy(ng, 3320);
    cout << copy;
    return 0;
}

```

1.2. Задача

Да се дефинира класа DebitAccount за работа со дебитна банкарска сметка. За секоја банкарска сметка треба да се чува име и презиме на корисникот (низа од макс. 100 знаци), број на сметка (long број) и моментално салдо (double). Да се овозможат методи за преглед на сметката, депонирање и подигнување на пари од сметката.

Потоа да се дефинира класа CreditAccount што ќе овозможува корисникот на сметката да зема заем од банката. Треба да се овозможи механизам за пресметување на камата доколку корисникот должи пари на банката.

```

#include <iostream>
#include <cstring>
using namespace std;

class DebitAccount {
protected:
    char name[100];
    long number;
    double balance;
public:
    DebitAccount(const char *name = "----", const long number = 0,
                 const double balance = 0.0) {
        strncpy(this->name, name, 99);
        this->name[99] = 0;
        this->number = number;
        this->balance = balance;
    }
    void showInfo() {
        cout << name << '\n'
              << "\t Bank No: " << number << '\n'
              << "\t Balance: " << getBalance() << '\n';
    }

    void deposit(double amount) {
        if (amount >= 0) {
            balance += amount;
        }
        else {
            cout << "You can not add negative amount to your balance!" << endl;
        }
    }

    void withdraw(double amount) {
        if (amount < 0) {
            cout << "You can not withdraw negative amount from your account!" << endl;
            return;
        }

        if (amount <= balance) {
            balance -= amount;
        }
        else {
            cout << "You can not withdraw more money than you have on your account.\n"
                  << "Please upgrade your debit account to credit account!" << endl;
        }
    }

    double getBalance() {
        return this->balance;
    }

    ~DebitAccount() {}
};

class CreditAccount : public DebitAccount {
private:
    double limit;
    double interest; // % percent
    double minus;
public:
    CreditAccount(const char *name="----", const long number=0,
                  const double balance=0, const double limit=1000,
                  const double interest=0.05, const double minus=0):
        DebitAccount(name, number, balance) {

        this->limit = limit;
        this->interest = interest;
        this->minus = minus;
    }

    void withdraw(double amount) {
        int balance = getBalance();
    }

```

```

    if (amount <= balance){
        DebitAccount::withdraw(amount);
    }

    else if (amount <= balance + limit - minus) {
        double advance = amount - balance;
        this->minus +=advance * (1.0 + interest);

        cout<<"Minus: " << advance <<"\n"
             << "Minus with interest: " << advance*interest<<endl;

        deposit(advance);
        DebitAccount::withdraw(amount);

    } else {
        cout << "The bank is not giving you that much money..." << endl;
        this->showInfo();
    }
}

void showInfo() {
    DebitAccount::showInfo();
    cout<<"\t Limit: " <<this->limit << "\n"
         <<"\t In minus: " << this->minus << "\n"
         <<"\t Interest: " << this->interest << "%\n";
}

double getInterest() {
    return this->interest;;
}

};

int main() {
    DebitAccount d("Pero Perovski", 6, 100000);
    CreditAccount ca("Mitko Mitkovski", 10, 5000, 1000);
    d.showInfo();
    d.deposit(50000);
    d.withdraw(600000);
    d.showInfo();
    ca.showInfo();
    ca.deposit(500);
    ca.showInfo();
    ca.withdraw(6200);
    ca.showInfo();
    return 0;
}

```

2. Изворен код од примери и задачи

<https://github.com/finki-mk/OOP/>

Source code ZIP