Објектно ориентирано програмирање

Аудиториски вежби 7

Верзија 1.0, 03 април, 2017

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>
using namespace std;

class Array {
private:
    int *x;
    int size;
    int capacity;
public:
    Array(const int capacity = 5) {
        x = new int[capacity];
        size = 0;
        this->capacity = capacity;
    }

    // copy constructor
    Array(const Array &a) {
        size = a.size;
        capacity = a.capacity;
        x = new int[capacity];
        for (int i = 0; i < size; ++i) {
            x[i] = a.x[i];
        }
    }

    // asignment operator =
    Array& operator=(const Array &a) {
        if (this == &a) return *this;
        size = a.size;
        capacity = a.capacity;
        delete [] x;
        x = new int[capacity];
        for (int i = 0; i < size; ++i) {
            x[i] = a.x[i];
        }
        return *this;
```

```
    }

    // destructor
    ~Array() {
        delete [] x;
    }
    int getSize() {
        return size;
    }

    int getCapacity() {
        return capacity;
    }

    const int *getX() {
        return x;
    }
    //na nizata x od dadeniot objekt dodadi go elementot n
    Array & operator+= (int n) {
        if (capacity == size) {
            int *y = new int[2 * capacity];
            for (int i = 0; i < size; ++i) {
                y[i] = x[i];
            }
            delete [] x;
            x = y;
            capacity = capacity * 2;
        }
        x[size] = n;
        size++;
        return *this;
    }

    //od nizata x od dadeniot objekt izbrishi go elementot n
    Array & operator-= (int n) {
        int newSize = 0;
        for (int i = 0, j = 0; i < size; ++i)
            if (x[i] != n) {
                x[j++] = x[i];
                newSize++;
            }
        size = newSize;
        return *this;
    }
//friend ostream & operator<<(ostream &o, Array &a);


};

ostream& operator<<(ostream &o, Array &a) {
    for (int i = 0; i < a.getSize(); ++i) {
```

```
        o << a.getX()[i] << " ";
    }
    for (int i = a.getSize(); i < a.getCapacity(); ++i) {
        o << "- ";
    }
    o << endl;
    return o;
}
int main() {

    Array a;
    a += (6);
    a += (4);
    a += (3);
    a += (2);
    a += (1);

    Array b(a);

    b -= (2);
    b -= (3);

    cout << " a: " << a;
    cout << " b: " << b;

    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.

Решение oop_av72.cpp

```
#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 showBalance() {
    cout << name << '\n' << number << '\n' << balance << endl;
  }

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

  void withdrawMoney(double amount) {
    if (amount < 0) {
      cout<< "You can note 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 debut account to credit account!"<<endl;
    }
  }
};

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

  void withdraw(double amount) {
    if (amount <= balance + limit) {
```

```
      balance -= amount;
    } else {
      cout << "The bank is not giving you that much money..." << endl;
      cout << "Balance: " << balance << endl;
      cout << "Limit: " << limit << endl;
    }
  }

  static void showInterest() {
    cout << interest << endl;
  }


};

int main() {
  BankAccount ba("Pero", 6, 1000000);
  BankAccountPlus bap("Mite", 10, 5000, 1000);
  ba.showBalance();
  ba.addMoney(50000);
  ba.withdrawMoney(600000);
  ba.showBalance();
  bap.showBalance();
  bap.addMoney(500);
  bap.showBalance();
  bap.withdrawMoney(7000);
  bap.showBalance();
  bap.showInterest();
  BankAccountPlus::showInterest();
  return 0;
}
```

2. Изворен код од примери и задачи

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

Source code ZIP