"Ss. Cyril and Methodius" University in Skopje
FACULTY OF COMPUTER SCIENCE AND ENGINEERING

Lesson 9

Static members and exceptions

Object oriented programming

Problem 1
Part 1

Model a base class `PaymentCard`, and two derived classes `CreaditCard` and `DebitCard`. Each card is described with its id number and the current balance.
When paying with debit cards, on each amount there is a discount of 5% for ALL users and this percent cannot be changed.
When paying with credit cards, on each amount there is discount of 10% if the limit is over 6,000, and 3% otherwise. The 10% limit is the same for all users but it can be changed from state bank.

Problem 1

Solution 1/5

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

class PaymentCard {
protected:
    int id;
    double balance;
public:
    PaymentCard(const int id = 0, const double balance = 0) {
        this->id = id;
        this->balance = balance;
    }

    virtual void pay(double amount) = 0;

    void deposit(double amount) {
        this->balance += amount;
    }

    void show() {
        cout << "ID: " << id << endl;
        cout << "Balance: " << balance << endl;
    }

};
```

Problem 1
Solution 2/5

```
class DebitCard : public PaymentCard {
private:
    static const double DISCOUNT = 5;
    char pin[5];
public:
    DebitCard(const int id = 0, const double balance = 0, const char *pin = "")
         : PaymentCard(id, balance) {
        strcpy(this->pin, pin);
    }

    void pay(double amount) {
        double toPay = amount * (1 - 1 / DISCOUNT);
        if (toPay < balance) {
            balance -= toPay;
        } else {
            cout << "Not enough money :(" << endl;
        }
    }
};
```

Problem 1
Solution 3/5

```cpp
class CreditCard : public PaymentCard {
private:
    double limit;
public:
    CreditCard(const int id = 0, const double balance = 0, const double limit =
         1000) : PaymentCard(id, balance) {
        this->limit = limit;
    }

    void pay(double amount) {
        double toPay = amount * (1 - 1 / 10.);
        if (limit < 6000) {
            toPay = amount * (1 - 1 / 3.);
        }
        if (toPay > balance + limit) {
            balance -= toPay;
        } else {
            cout << "Not enough money :(" << endl;
        }
    }
};
```

Problem 1
Part 2

Create a class `CashRegister` where customers can pay with cards or in cash. For each cash register there are two amounts:

- amount payed in cash
- amount payed with cards

and each object of this class has the date when it is created. Implement the following two functions in this class:

- `pay(double)` - for paying in cash
- `pay(double, PaymentCard)` - for paying with card

Problem 1

Solution 4/5

```
class CashRegister {
private:
    double amountCash;
    double amountCard;
    int day, month, year;
public:
    CashRegister(const int d = 1, const int m = 1, const int y = 2000) {
        amountCash = 0;
        amountCard = 0;
        day = d;
        month = m;
        year = y;
    }

    void pay(double amount) {
        amountCash += amount;
    }

    void pay(double amount, PaymentCard &card) {
        card.pay(amount);
        amountCard += amount;
    }

    void show() {
        cout << day << "." << month << "." << year << endl;
        cout << "Amount cash: " << amountCash << endl;
        cout << "Amount card: " << amountCard << endl;
    }
};
```

Problem 1
Solution 5/5

```
int main() {
    CashRegister daily(22, 4, 2014);
    PaymentCard *card;
    daily.show();
    cout << "Paying in cash!" << endl;
    daily.pay(5000);
    daily.show();
    card = new CreditCard(12345678, 54000.00, 10000.00);
    cout << "Paying with card!" << endl;
    daily.pay(10000, *card);
    daily.show();
    delete card;
    card = new DebitCard(123456789, 54000.00, "4321");
    cout << "Paying with card!" << endl;
    daily.pay(10000, *card);
    daily.show();
    delete card;
    return 0;
}
```

Problem 2

Part from the products from one store after the new policy of the store must have some discount. To acomplish this the store system must model abstract class Discount. This class keeps info about the exchange rates for euros and dollars in denars and each class deriving from it must implement the following methods:

- `float discount_price()`
- `float price()`
- `void print_rule()`

For each Product the store keeps info about the name and the price. Products are divided in few types: FoodProduct, Drinks and Cosmetics. According to the new policy, food does not have discount. Alchohol drinks more expensive than 20 euros have discount of 5%, and alchohol free drinks from the brand CocaCola have discount of 10%. All cosmetic products have discount of 12%, and those more expensive than 20 dollars have discount of 14%.

Problem 2
Part 2

Compute the total price of all products with the discount applied.
The price of the products should not be negative. Throw an exception in the constructor of the class Product, if the data is not ok.
Try different strategies of handling the exception:

- Catch the exception in the constructor, so the price is set to 0
- Catch the exception in the main function, where we instantiate objects of classes derived from Product.

Problem 2

Solution 1/5

```
#include <iostream>
#include <cstring>
using namespace std;
class Discount {
public:
    static float euro;
    static float dollar;
    virtual float discount_price() = 0;
    virtual float price() = 0;
    virtual void print_rule() = 0;
};
float Discount::euro = 61.7;
float Discount::dollar = 44.5;

class Product {
protected:
    char name[100];
    float price;
public:
    Product(const char *name = "", const float price = 0) {
        try {
            if (price < 0) throw 1;
            strcpy(this->name, name);
            this->price = price;
        } catch (int) {
            cout << "Wrong price. Exception is thrown!" << endl;
            this->price = 0.0;
        }
    }
    float getPrice() {
        return price;
    }
    void print() {
        cout << "Product{ name=" << name << ", price=" << price << "}" << endl;
    }
};
```

Problem 2

Solution 2/5

```
class Cosmetics : public Product, public Discount {
private:
    int weight;
public:
    Cosmetics(const char *name = "", const float price = 0,
              const int weight = 0) : Product(name, price) {
        this->weight = weight;
    }
    float discount_price() {
        if (getPrice() / Discount::dollar > 20)
            return 0.86 * getPrice();
        if (getPrice() / Discount::euro > 5)
            return 0.88 * getPrice();
        return getPrice();
    }
    float price() {
        return getPrice();
    }
    void print_rule() {
        cout << "All cosmetic products with price over 5 euros have 12% discount
             ." << endl;
        cout << "All cosmetic products with price over 10 dollars have discount
             14%." << endl;
    }
};
```

Problem 2
Solution 3/5

```
class FoodProduct : public Product, public Discount {
private:
    float callories;
public:
    FoodProduct(const char *name = "", const float price = 0,
                const float callories = 0) : Product(name, price) {
        this->callories = callories;
    }
    float discount_price() {
        return getPrice();
    }
    float price() {
        return getPrice();
    }
    void print_rule() {
        cout << "No discount for food products" << endl;
    }
};
```

Problem 2
Solution 4/5

```
class Drinks : public Product, public Discount {
private:
    char brand[100];
    bool alcoholic;
public:
    Drinks(const char *name = "", const float price = 0,
           const char *brand = "", const bool alcoholic = false) : Product(name,
                 price) {
        strcpy(this->brand, brand);
        this->alcoholic = alcoholic;
    }
    float discount_price() {
        if (this->alcoholic && (getPrice() / Discount::euro > 20))
            return 0.95 * getPrice();
        if (!this->alcoholic && (strcmp(this->brand, "Coca-Cola") == 0))
            return 0.90 * getPrice();
        return getPrice();
    }
    float price() {
        return getPrice();
    }
    void print_rule() {
        cout << "All alcholic beverages with price over 20 euros have 5%
             discount. " << endl;
        cout << "All no alcholic beverages of the brand Coca-Cola have discount
             of 10%" << endl;
    }
};
```

Problem 2

Solution 5/5

```
float total_discount(Discount **d, int n) {
    float discount = 0;
    for (int i = 0; i < n; ++i) {
        discount += d[i]->discount_price();
        cout << "Normal price: " << d[i]->price() << endl;
        cout << "Discounted price: " << d[i]->discount_price() << endl;
        d[i]->print_rule();
    }
    return discount;
}

int main() {
    int n = 7;
    Discount **d = new Discount*[n];
    try {
        d[0] = new FoodProduct("Bread", -30);
    } catch(int) {
        cout << "Negative price in constructor" << endl;
    }
    d[1] = new Drinks("Whiskey", 1350, "Jack Daniel's", true);
    d[2] = new FoodProduct("Cheese", 390, 105);
    d[3] = new Drinks("Vodka", 850, "Finlandia", true);
    d[4] = new Cosmetics("Cream", 72, 100);
    d[5] = new Drinks("Soda", 50, "Coca-Cola", false);
    d[6] = new Cosmetics("Parfume", 3500, 50);
    cout << "Total price is: " << total_discount(d, n) << endl;
    for (int i = 0; i < n; ++i) {
        delete d[i];
    }
    delete [] d;
    return 0;
}
```

Materials and Questions

Lectures, exsercises and announcements
courses.finki.ukim.mk

Source code of all examples and problems
https://github.com/tdelev/SP/tree/master/latex/src

Questions and discussion
forum.finki.ukim.mk