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

Lesson 8

Multiple Inheritance

Object oriented programming

Problem 1

Create a class hierarchy for `JetCar` which derives from two classes, car and jet (diamond problem).

Problem 1

Solution 1/4

```
#include <iostream>
using namespace std;
class Vehicle {
public:
    Vehicle() {
        cout << "Vehicle Constructor" << endl;
    }
    virtual ~Vehicle() {
        cout << "Vehicle Destructor" << endl;
    }
    virtual void accelerate() const {
        cout << "Vehicle Accelerating" << endl;
    }
    void setAcceleration(double a) {
        acceleration = a;
    }
    double getAcceleration() const {
        return acceleration;
    }
protected:
    double acceleration;
};
```

Problem 1
Solution 2/4

```
class Car: public Vehicle {
public:
    Car() {
        cout << "Car Constructor" << endl;
    }
    virtual ~Car() {
        cout << "Car Destructor" << endl;
        virtual void accelerate() const {
            cout << "Car Accelerating" << endl;
        }
        virtual void drive() const {
            cout << "Car Driving" << endl;
        }
    }
private:
    // Car inherits acceleration accessors, member
};
```

Problem 1
Solution 3/4

```
class Jet: public Vehicle {
public:
    Jet() {
        cout << "Jet Constructor" << endl;
    }
    virtual ~Jet() {
        cout << "Jet Destructor" << endl;
    }
    virtual void fly() const {
        cout << "Jet flying" << endl;
    };
    class JetCar: public Car, public Jet {
    public:
        JetCar() {
            cout << "JetCar Constructor" << endl;
        }
        virtual ~JetCar() {
            cout << "JetCar Destructor" << endl;
            virtual void drive() const {
                cout << "JetCar driving" << endl;
            }
            virtual void fly() const {
                cout << "JetCar flying" << endl;
            }
        };
    }
};
```

Problem 1
Solution 4/4

```
void analyzeCarPerformance(Car *testVehicle) {
    testVehicle->drive();
    //drive() exists for both base and sub class
}
void analyzeJetPerformance(Jet *testVehicle) {
    testVehicle->fly();
    //fly() exists for both base and sub class
}
int main() {
    Car myCar;
    Jet myJet;
    JetCar myJetCar;
    cout << endl << endl;
    cout << "Car testing in progress" << endl;
    analyzeCarPerformance(&myCar);
    analyzeCarPerformance(&myJetCar);
    cout << "Jet testing in progress" << endl;
    analyzeJetPerformance(&myJet);
    analyzeJetPerformance(&myJetCar);
    cout << endl << endl;
    return 0;
}
```

Problem 2

Implement class for `Product` that keeps name and price. Then implement abstract class `Discount` that have two pure virtual functions for price and price on discount. From this classes derive:

- `FoodProduct` - additionally keeps calories
- `DigitalProduct` - additionally keeps size (in MB).

Implement global function `total_discount` that will compute the total discount of N products passed as argument to this function.

Problem 2

Solution 1/4

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

class Discount {
public:
    virtual float discount_price() = 0;
    virtual float price() = 0;
};

class Product {
protected:
    char name[100];
    float price;
public:
    Product(const char *name = "", const float price = 0) {
        strcpy(this->name, name);
        this->price = price;
    }

    float getPrice() {
        return price;
    }

};
```

Problem 2

Solution 2/4

```
class DigitalProduct : public Product, public Discount {
private:
    float size;
public:
    DigitalProduct(const char *name = "", const float price = 0, const float
         size = 0) : Product(name, price) {
        this->size = size;
    }

    float discount_price() {
        // 10% discount
        return 0.9 * getPrice();
    }

    float price() {
        return getPrice();
    }
};
```

Problem 2
Solution 3/4

```
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() {
        // 20% discount
        return .8 * getPrice();
    }

    float price() {
        return getPrice();
    }
};
```

Problem 2
Solution 4/4

```cpp
class ClothesProduct : public Product {
private:
    char brand[100];
public:
    ClothesProduct(const char *name = "", const float price = 0, const char *
         brand = "") : Product(name, price) {
        strcpy(this->brand, brand);
    }
};
float total_discount(Discount **d, int n) {
    float price = 0;
    for (int i = 0; i < n; ++i) {
        price += d[i]->price();
    }
    float discount = 0;
    for (int i = 0; i < n; ++i) {
        discount += d[i]->discount_price();
    }
    return price - discount;
}
int main() {
    Discount **d = new Discount*[3];
    d[0] = new FoodProduct("Cheese", 450, 1200);
    d[1] = new FoodProduct("Wine", 780, 250);
    d[2] = new DigitalProduct("WOW", 380, 400);
    cout << "Difference: " << total_discount(d, 3) << endl;
    for (int i = 0; i < 3; ++i) {
        delete d[i];
    }
    delete [] d;
```

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