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

Lesson 7

Inheritance
Polymorphism

Object oriented programming

Contents

Problem 1

Define a class for representing publication. For each publication there is information for year of publishing (integer) and for the name of publishing house (char[100]).
From this class derive (public) a new class for book. For each book additionally there is number of pages.
Also derive (protected) a new class for newspaper. For each newspaper there is a serial number.
From class newspaper derive (private) class for daily newspaper. For each daily newspaper there is information for the day and month of publishing.
Test the access to the data with various functions!

Problem 1
Solution 1/5

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

class Publication {
private:
    char name[100];
protected:
    int year;
    char *getName() {
        return name;
    }
public:
    int getYear() {
        return year;
    }
    void print() {
        cout << "Publication: " << name << " - " << year << endl;
    }
    Publication(char *name, int year) {
        strcpy(this->name, name);
        this->year = year;
    }
};
```

Problem 1
Solution 2/5

```
// public inheritance
class Book: public Publication {
private:
    int pages;
public:
    Book(char *name, int year, int pages): Publication(name, year) {
        this->pages = pages;
    }
    void printYear() {
        cout << year;  //accessing protected variable year
    }
    void printName() {
        cout << getName(); //access to getName(), name cannot be accessed
             because is private
    }
    void printPages() {
        cout << pages;
    }
};
```

Problem 1
Solution 3/5

```
// protected inheritance
class NewsPaper: protected Publication {
private:
    int number;
public:
    NewsPaper(char *name, int year, int number): Publication(name, year) {
        this->number = number;
    }
    void printYear() {
        cout << getYear();  //access to public getYear(), which in this class
             has protected access
    }
    void printName() {
        cout << getName(); //access to getName(), name cannot be accessed
             because is private
    }
    void printNumber() {
        cout << number;
    }

};
```

Problem 1
Solution 4/5

```
//private inheritance
class DailyNewsPaper: private NewsPaper {
private:
    int day;
    int month;
public:
    DailyNewsPaper(char *name, int day, int month, int year, int number):
         NewsPaper(name, year, number) {
        this->day = day;
        this->month = month;
    }
    using NewsPaper::print;  // function print from Publication becomes public
         for DailyNewsPaper
    using NewsPaper::printNumber; // function printNumber from Publication
         becomes public for DailyNewsPaper
};
```

Problem 1
Solution 5/5

```
int main() {
    Publication p("Tabernakul", 1992);
    p.print();  // public - function

    Book *k = new Book("ProsvetnoDelo", 1900, 123);
    k->print(); //print is public in Book
    k->printyearBook(); // public - function
    cout << k->getName(); // error! protected - function

    NewsPaper *s = new NewsPaper("Tea", 2013, 30);
    s->print();  //error! protected - function
    cout << s->getYear(); //  error! protected - function
    s->printName();  // public - function

    DailyNewsPaper d("Vest", 2, 3, 2014, 25);
    d.print(); //public-function
    d.printName(); // error! private - function
    cout << d.getName(); // error! private[U+FFFD]function

}
```

Problem 2

Define a class for hotel reservation. For each hotel reservation is stored the number of days, number of persons, name for contact. The price for the reservation is 25 EUR per day per person.
Define a function `getPrice()` that returns the price of the reservation. Define a function `getPrice(int advance)` that returns the price if the user pays advance.
Derive a new class breakfast hotel reservation for reserving hotel room with breakfast. The price for breakfast for one person is 5 EUR per day. Override the respective function `getPrice(int advance)`.

Problem 2
Solution 1/4

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

class HotelReservation {
protected:
    int days;
    int persons;
    char name[50];
public:
    HotelReservation(const char *name, int days, int persons) {
        strcpy(this->name, name);
        this->days = days;
        this->persons = persons;
    }

    int getPrice() {
        cout << "HotelReservation::getPrice()" << endl;
        return days * persons * 25;
    }

    virtual int getPrice(int advance) {
        cout << "HotelReservation::getPrice(int)" << endl;
        if (advance >= getPrice())
            return advance - getPrice();
        else {
            cout << "For your reservation you should pay " << getPrice() << endl
                ;
            return -1;
        }
```

Problem 2
Solution 2/4

```
};

class BreakfastHotelReservation: public HotelReservation {
public:
    BreakfastHotelReservation(const char *name, int days, int persons):
         HotelReservation(name, days, persons) {
    }

    //overriding getPrice(int advance)
    int getPrice(int advance) {
        cout << "BreakfastHotelReservation::getPrice(int)" << endl;
        int price = HotelReservation::getPrice() + persons * 5; // accessing
             protected data persons
        if (advance >= price)
            return advance - price;
        else {
            cout << "For your reservation you should pay " << price << endl;
            return -1;
```

Problem 2 ++

Define a class Hotel with info about the name and the balance. In the class define a function:

```
int reserve(HotelReservation hr, int advance);
```

With this function we should pay for hotel reservation. If the payment exceeds the needed amount, the function should return the change. The payment should add to the balance of the hotel.

What will happen if the argument of HotelReservation is not reference?

Problem 2
Solution 3/4

```
    }
};

class Hotel {
private:
    char name[50];
    int balance;
public:
    Hotel(char *name) {
        strcpy(this->name, name);
        balance = 0;
    }
    // reference to the base class that can reference to objects of derived
         classes
    int reserve(HotelReservation *hr, int advance) {
        int change = hr->getPrice(advance); //polymorphism
        // which definition of getPrice will be called?
        // important! getPrice is VIRTUAL function
        if (change != -1)
            balance += advance - change;
        return change;
```

Problem 2
Solution 4/4

```
};

int main() {
    Hotel h("Bristol");
    HotelReservation *hr1 = new HotelReservation("Petko Petkovski", 5, 5);
    int price = h.reserve(hr1, 1000);
    delete hr1;
    if (price != -1)
        cout << "Change : " << price << endl;

    HotelReservation *hr2 = new BreakfastHotelReservation("Risto Ristovski", 5,
         5);
    price = h.reserve(hr2, 1000);
    delete hr2;
    if (price != -1)
        cout << "Change : " << price << endl;
    // pointer from the base class points to object of derived
    HotelReservation hr3("Ana Anovska", 4, 2);
    price = h.reserve(&hr3, 100);
        if (price != -1)
        cout << "Change : " << price << endl;

    BreakfastHotelReservation hr4("Tome Tomovski", 5, 3);
    price = h.reserve(&hr4, 1000);
    if (price != -1)
        cout << "Change : " << price << endl;

    return 0;
}
```

Problem 3

Define abstract class for representing a geometric shape. Each geometric shape has height and base which can be different for each shape.
Add the following functions in the class:

- `print()` print the shape info
- `volume()` the volume of the shape
- `getHeight()` returns the height of the shape

From this class derive classes for cylinder, cone and cuboid. For the cylinder and cone, the information for the radius of the basis is stored and for the cuboid, lenght of sides a and b.

Problem 3

Solution 1/4

Problem 3 ++

In `main` declare and initialize dynamically array of pointers to the class gemotric shape. From this array:

1. Find the shape with largest volume, using the function: `void largestShape(GemotricShape **array, int n)`
2. Find the number of geometric shapes that doesn't have basis circle using the function: `double getRadius(GeometricShape *g);` this function returns the radius of the basis of the shapes that have circle basis, and -1 otherwise.

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