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

# Објектно ориентирано програмирање

Аудиториски вежби 8

Верзија 1.0, 10 април, 2017

# Содржина

# 1. Полиморфизам

## 1.1. Задача

Да се дефинира класа за репрезентација на публикација. За секоја публикација се чуваат информации за годината на издавање (цел број) и за називот на издавачката куќа (низа од знаци).

Од класата публикација да се изведе (public) класа за книга. За секоја книга дополнително треба да се чува и бројот на страници на книгата.

Од класата публикација да се изведе (protected) класа за весник. За весник се чува редниот број на весникот.

Од класата весник да се изведе (private) класа за дневен весник. За секој дневен весник се чува информација за денот и месецот на издавање на весникот.

Со разни функции да се испита пристапот до податоците и функциите!

*Решение* `oop_av81.cpp`

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

using namespace std;
class Publikacija {
private:
  char naziv [100];
protected:
  int godina;
  char* getNaziv() {
    return naziv;
  }

public:
  int getGodina () { return godina;}
  void pecati () {
    cout << "Publikacija: "<< naziv << " - " << godina << endl;
 }
 Publikacija( char *naziv, int godina ) {
   strcpy(this->naziv, naziv);
   this->godina = godina;
 }
};

// public изведување
class Kniga: public Publikacija {
private:
  int broj_strani;
public:
  Kniga(char *naziv, int godina, int broj_strani):Publikacija(naziv, godina){
    this->broj_strani = broj_strani;
  }
  void pecatiGodinaKniga(){
    cout << godina; //пристап до protected променливата godina
  }
```

```
  void pecatiNazivKniga (){
    //пристап до getNaziv(), naziv не може да се пристапи затоа што е private
    cout << getNaziv();
  }
  void pecatiStrani (){
    cout << broj_strani;
  }
};

// protected изведување
class Vesnik: protected Publikacija{
private:
  int broj;
public:
  Vesnik(char* naziv, int godina, int broj):Publikacija(naziv, godina){
    this->broj=broj;
  }
  void pecatiGodinaVesnik (){
    cout << getGodina(); //пристап до public getGodina(), која во оваа класа има protected
пристап
      }
  void pecatiNazivVesnik(){
    cout << getNaziv(); //пристап до getNaziv(), naziv не може да се пристапи затоа што е
private
      }
  void pecatiBroj(){
    cout << broj;
  }
};

//private изведување
class DnevenVesnik: private Vesnik{
private:
  int den;
  int mesec;
public:
  DnevenVesnik(char *naziv, int den, int mesec, int godina,
               int broj):Vesnik(naziv, godina,broj){
    this->den=den;
    this->mesec=mesec;
  }
  using Vesnik::pecati; // функцијата pecati од Publikacija станува public за DnevenVesnik
  using Vesnik::pecatiBroj; // функцијата pecatiBroj од Publikacija станува public за
DnevenVesnik
};

int main()
{
  Publikacija p("Tabernakul", 1992);
  p.pecati(); // public - функција
  Kniga *k = new Kniga("ProsvetnoDelo", 1900, 123);
  k->pecati(); //pecati e public во Kniga
  k->pecatiGodinaKniga(); // public - функција
  // cout<<k->getNaziv(); // грешка! protected - функција
  Vesnik *s = new Vesnik("Tea", 2013 ,30);
  // s->pecati(); //грешка! protected - функција
  // cout<<s->getGodina(); // грешка! protected - функција
  s->pecatiNazivVesnik(); // public - функција
  DnevenVesnik d("Vest",2,3,2014,25);
  d.pecati(); //public-функција
  // d.pecatiNazivVesnik(); // грешка! private - функција
  // cout<<d.getNaziv(); // грешка! private - функција
}
```

## 1.2. Задача

Да се дефинира класа за **Хотелска резервација**. За хотелската резервација се чува бројот на денови, бројот на лица, име и презиме на лицето за контакт. Да се земе дека цената на резервацијата за едно лице за еден ден е 25 евра.

Во класата да се дефинира функција `vratiCena()` која враќа колкава е цената на резервацијата. Во класата да се дефинира функција `vratiCena(int uplata)` која враќа колкава цена треба да се врати од касата ако корисникот на резервацијата ја доставува дадената уплата.

Да се изведе класа **Полупансионска хотелска резервација** за резервирање на хотелска соба со појадок. Цената на појадокот за едно лице за еден ден е 5 евра. Да се препокрие соодветно функцијата `vratiCena(int uplata)`.

Да се дефинира класа за Хотел со информации за името на хотелот и салдо на хотелот. Во класата да се дефинира функција `int uplatiZaRezervacija(HotelsaRezervacija &hr, int uplata);` Со оваа функција треба да се направи уплата за дадена хотелска резервација. Ако уплатата ја надминува бараната сума функцијататреба да врати колку пари треба да му се врати на корисникот кој ја прави уплатата. Уплатата треба да се додаде во салдото на хотелот.

**Што ќе се случеше ако аргументот не е референца?**

*Решение* `oop_av82.cpp`

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

class HotelskaRezervacija{
protected:
  int denovi;
  int broj_lica;
  char ime[50];
  char prezime[50];

public:
  HotelskaRezervacija(char *ime, char *prezime, int denovi, int broj_lica){
    strcpy(this->ime,ime);
    strcpy(this->prezime, prezime);
    this->denovi=denovi;
    this->broj_lica=broj_lica;
 }

  virtual int vratiCena() {
    return denovi * broj_lica * 25;
  }
  virtual int vratiCena(int uplata) {
```

```
    if (uplata >= vratiCena())
      return uplata - vratiCena();
    else {
      cout<<"Za vashata rezervacija treba da naplatite "<<vratiCena()<<endl;
      return -1;
    }
  }
};


class PolupansionskaHotelskaRezervacija: public HotelskaRezervacija{
public:
  PolupansionskaHotelskaRezervacija(char *ime, char *prezime, int denovi, int
                                    broj_lica) : HotelskaRezervacija(ime,prezime,denovi
,broj_lica){}

  //препокривање на vratiCena(int uplata)
  int vratiCena(int uplata){
    int cena= HotelskaRezervacija::vratiCena() + broj_lica * 5; // пристап до protected
податокот broj_lica
    if (uplata >= cena)
      return uplata - cena;
    else {
      cout<<"Za vashata rezervacija treba da naplatite "<<cena<<endl;
      return -1;
    }
  }
};

class Hotel{
private:
  char ime[50];
  int saldo;
public:
  Hotel(char *ime) {
    strcpy(this->ime, ime);
    saldo = 0;
  }
  // референца кон основната класа може да референцира објекти и кон изведените класи
  int uplatiZaRezervacija(HotelskaRezervacija &hr, int uplata) {
    int kusur=hr.vratiCena(uplata); //полиморфизам
    // која дефиниција на vratiCena ќе се повика?
    // важно: vrtiCena е виртуелна функција
    if (kusur != -1)
      saldo += uplata - kusur;
    return kusur;
  }
};

int main() {
  Hotel h("Bristol");
  HotelskaRezervacija *hr1=new HotelskaRezervacija("Petko","Petkovski",5,5);
  int cena = h.uplatiZaRezervacija(*hr1,1000);
  if (cena!=-1)
    cout<<"Kusur : "<<cena<<endl;
  PolupansionskaHotelskaRezervacija *hr2=
    new PolupansionskaHotelskaRezervacija("Risto","Ristovski",5,5);
  cena=h.uplatiZaRezervacija(*hr2,1000);
  if (cena!=-1)
    cout<<"Kusur : "<<cena<<endl;
  //покажувач кон основна класа покажува кон објект од изведена
  HotelskaRezervacija *hr3=new PolupansionskaHotelskaRezervacija("Ana","Anovska",4,2);
  cena=h.uplatiZaRezervacija(*hr3,100);
  if (cena!=-1)
    cout<<"Kusur : "<<cena<<endl;
  PolupansionskaHotelskaRezervacija hr4("Tome","Tomovski",5,3);
  cena=h.uplatiZaRezervacija(hr4,1000);
  if (cena!=-1)
    cout<<"Kusur : "<<cena<<endl;
}
```

## 1.3. Задача

Да се дефинира апстрактна класа за репрезентација на едно геометриско тело. Секое геометриско тело има висина (децимален број) и основа која може да биде различна за различни геометриски фигури.
За апстракната класа да се дефинираат функциите:
- pecati() во која се печатат информациите за телото
- volumen() која ќе го враќа волуменот на телото
- getVisina() која ќе ја враќа висината на телото

Од класата геометриско тело да се изведе класа за цилиндер, конус и квадар. За еден цилиндер и за еден конус покрај висината се чува и информација за радиусот на основата (децимален број).За квадарот се чуваат информации за страните `a` и `b` на основата (децимални броеви).
Во `main` функцијата да се декларира и иницизијализира динамички алоцирана низа од покажувачи кон класата која претставува геометриско тело. Од оваа низа:

1. Да се одреди телото со најголем волумен користејќи ја глобалната функција: `void teloSoNajgolemVolumen(GeomTelo *niza[], int n);` Оваа функција треба да ги отпечати информациите за телото од полето предадено како аргумент (`niza` - поле од покажувачи кон `GeomTelo`) кое има најголем волумен.
2. Да се одреди бројот на геометриски тела кои немаат основа круг користејќи ја глобалната функција со потпис: `double getRadius(GeomTelo *g);`. Оваа функција која ќе го враќа радиусот на основата (ако основата на телото е круг), а -1 во спротивно.

*Решение* `oop_av83.cpp`

```
#include<iostream>
#include<cmath>
using namespace std;
class GeomTelo{
protected:
  double visina;
public:
  GeomTelo (int visina = 0) {
    this->visina=visina;
  }
  virtual void pecati() { //виртуелна функција
    cout << visina;
  }
  virtual double getVolumen() = 0; //чисто виртуелна функција
  double getVisina() const {
    return visina;
  }
};

class Cilinder: public GeomTelo{
private:
  double radius;
public:
```

```
  // конструктор
  Cilinder(double radius, double visina):GeomTelo(visina){
    this->radius = radius;
  }
  //препокривање на функцијата pecati()
  void pecati(){
    cout << "Cilinder so visina ";
    GeomTelo::pecati();
    cout << " i so radius na osnovata " << radius << endl;
  }
  //препокривање на функцијата getVolumen()
  double getVolumen(){
    return M_PI * radius * radius * getVisina();
  }
  double getRadius(){
    return radius;
  }
};

class Konus: public GeomTelo {
private:
  double radius;
public:
  // контсруктор
  Konus(double radius, double visina):GeomTelo(visina){
    this->radius=radius;
  }
  //препокривање на функцијата pecati()
  void pecati(){
    cout << "Konus so visina ";
    GeomTelo::pecati();
    cout << " i so radius na osnovata " << radius << endl;
  }
  //препокривање на функциајта getVolumen()
  double getVolumen(){
    return M_PI*radius*radius*getVisina()/3.0;
  }
  double getRadius(){
    return radius;
  }
};


class Kvadar: public GeomTelo{
private:
  double a,b;
public:
  //конструктор
  Kvadar(double a, double b, double visina):GeomTelo(visina){
    this->a=a;
    this->a=b;
  }
  //препокривање на функцијата pecati()
  void pecati(){
    cout << "Kvadar so visina ";
    GeomTelo::pecati();
    cout << "i so osnova " << this->a << " i " << this->b <<endl;
  }
  //препокривање на функцијата getVolumen()
  double getVolumen(){
    return a * b * getVisina();
  }
};
void teloSoNajgolemVolumen(GeomTelo *niza[], int n );
double getRadius (GeomTelo *g);
int main(){
  GeomTelo** mnozestvoTela; //динамички алоцирано поле од покажувачи кон GeomTelo
  int n;
  cin>>n; //број на тела во множеството
  mnozestvoTela = new GeomTelo*[n]; //се алоцира меморија за полето од покажувачи
  for (int i = 0 ; i < n ; i++){
    int r,a,b,h,type;
    cout<<"Kakvo telo: 1-cilinder 2-konus 3-kvadar "<<endl;
```

```
    cin >> type;
    if (type==1) { //ако корисникот внесува цилиндер
      cin >> r >> h; mnozestvoTela[i]= new Cilinder(r,h); }
    if (type==2) { //ако корисникот внесува конус
      cin >> r >> h; mnozestvoTela[i]= new Konus(r,h); }
    if (type==3) { //ако корисникот внесува квадар
      cin >> a >> b >> h; mnozestvoTela[i]= new Kvadar(a,b,h); }
  }
    //барање 1
    teloSoNajgolemVolumen(mnozestvoTela,n);
    //барање 2
    int brojac=0;
    for (int i = 0 ; i < n ; i++)
      if (getRadius(mnozestvoTela[i]) == -1)
        brojac++;
    cout<<"Brojot na tela koi nemaat osnova krug e "<<brojac;
}
void teloSoNajgolemVolumen(GeomTelo *niza[], int n)
{
  int maxVolumen=0;
  int maxIndex=0;
  for (int i = 0 ; i < n ; i++)
    {
      if (niza[i]->getVolumen() > maxVolumen){
        //се повикува виртуелната функција getVolumen()
        maxVolumen=niza[i]->getVolumen();
        maxIndex=i;
      }
    }
  cout<<"Teloto so najgolem Volumen e:";
  niza[maxIndex]->pecati(); //се повикува виртуелната функција pecati()
}
double getRadius(GeomTelo *g){
  //претворање за време на извршување
  Cilinder* c = dynamic_cast<Cilinder *>(g);
  if (c != 0){ //ако може да се изведе претворањето
    return c->getRadius();
  }
  //претворање за време на извршување
  Konus* k = dynamic_cast<Konus *>(g);
  if (k!= 0 ){ //ако може да се изведе претворањето
    return k->getRadius();
  }
  return -1; // ако g не покажува кон цилиндер, ни кон конус врати -1
}
```

# 2. Изворен код од примери и задачи

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

Source code ZIP