ФАКУЛТЕТ ЗА ЕЛЕКТРОТЕХНИКА И ИНФОРМАЦИСКИ ТЕХНОЛОГИИ

ПОДАТОЧНИ СТРУКТУРИ И ПРОГРАМИРАЊЕ 2023/2024

Аудиториски вежби 11

Наследување

Наследување на класи

```
class ime_na_izvedena_klasa : pristap ime_na_osnovna_klasa
{
// definicija na klasa
};
```

Наследство со public пристап до основната класа

```
#include <iostream>
using namespace std;
class Osnovna
{
	int i, j;
	void fja() {}
public:
	Osnovna() { i = j = 0; }
	Osnovna(int x, int y) { i = x; j = y; }
	void set(int a, int b) { i = a; j = b; }
	void show() { cout << i << " " << j << "\n"; }
};
class Izvedena : public Osnovna
{
	int k;
public:
	Izvedena(int x) { k = x; }
	void showk() {
	cout << k << "\n";
	fja(); /* грешка, не може да се пристапи до private членовите од надкласата */
	}
};

int main()
{
	Izvedena iz(3);
	Osnovna os(2, 4);
	os = iz; //објект од основна може да се изедначи со објект од изведена
	iz = os; //грешка, но не и обратно!!!
	os.showk(); // грешка, showk() е ф-ја дефинирана само во изведената класа!!!
	iz.set(1, 2); // пристап до метод од класата Osnovna
	iz.show(); // пристап до метод од класата Osnovna
	iz.showk(); // пристап до метод од класата Izvedena
	iz.fja(); // грешка, методот се наследува, но е приватен
	return 0;
}
```

Конструктори и деконструктори во изведена класа

```
#include <iostream>
using namespace std;
class Element
{
public:
	Element() { cout << "Constructing Element \n"; }
	~Element() { cout << "Destructing Element \n"; }
};
class Osnovna
{
public:
	Osnovna() { cout << "Constructing Osnovna \n"; }
	~Osnovna() { cout << "Destructing Osnovna \n"; }
};
```

```
class Izvedena : public Osnovna
{
	Element x;
public:
	Izvedena() { cout << "Constructing Izvedena \n"; }
	~Izvedena() { cout << "Destructing Izvedena \n"; }
};
int main()
{
	Izvedena ob;
	return 0;
}
```

Конструкторите се повикуваат според редоследот на изведување, а деструкторите во обратен редослед.

Печати:

```
Constructing Osnovna
Constructing Element
Constructing Izvedena
Destructing Izvedena
Destructing Element
Destructing Osnovna
```

ЗАДАЧИ:

1. За еден тениски играч се знае името, презимето и информација која кажува дали игра во лига. Некои од играчите се рангирани во ATP лига и за нив се знае и рангот. Да се напишат класи за реализирање на овие објекти.

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

class TenisIgrach
{
protected:
	string ime;
	string prezime;
	bool igraVoLiga;
public:
	TenisIgrach(string i = "", string p = "", bool il = false);
	TenisIgrach(const TenisIgrach & ti);
	bool igra() { return igraVoLiga; }
	void pechati();
};

// primer za ednostavno izvedena klasa
class RangiranIgrach : public TenisIgrach
{
private:
	int rang;
public:
	RangiranIgrach(int r = 0, string i = "", string p = "", bool ht = false);
	RangiranIgrach(int r, const TenisIgrach & tp);
	RangiranIgrach(const RangiranIgrach &ri);
	unsigned int Rangiranje() { return rang; }
	void ResetRang(unsigned int r) { rang = r; }
	void pechati();
};
```

```
// TenisIgrac metodi
TenisIgrach::TenisIgrach(string i, string p, bool il)
{
	ime = i;
	prezime = p;
	igraVoLiga = il;
}
TenisIgrach::TenisIgrach(const TenisIgrach &ti)
{
	ime = ti.ime;
	prezime = ti.prezime;
	igraVoLiga = ti.igraVoLiga;
}
void TenisIgrach::pechati()
{
	cout << endl << prezime << ", " << ime;
}

// RangiranIgrac metodi
RangiranIgrach::RangiranIgrach(int r, string i, string p, bool ht) : TenisIgrach(i, p,
ht)
{
	rang = r;
}

RangiranIgrach::RangiranIgrach(int r, const TenisIgrach & tp) : TenisIgrach(tp), rang(r)
{ }
RangiranIgrach::RangiranIgrach(const RangiranIgrach &ri) : TenisIgrach(ri), rang(ri.rang)
{ }
void RangiranIgrach::pechati()
{
	TenisIgrach::pechati();
	cout << " rang: " << rang << endl;
}
int main()
{
	TenisIgrach Igrac1("Tara", "Boomdea", false);
	RangiranIgrach rIgrac1(1140, "Mallory", "Duck", true);
	Igrac1.pechati();
	rIgrac1.pechati();
	if (rIgrac1.igra())
		cout << "pripagja na liga\n";
	else
		cout << "ne pripagja na liga\n";
	return 0;
}
```

2. Напишете класи за работа со Точка, Круг и Сфера.

```
#include <iostream>
#include <math.h>
using namespace std;
float const PI = 3.14;

class Tochka
{
private:
	float x, y;
protected:
	float rast(Tochka t)
	{
		return (sqrt((x - t.x)*(x - t.x) + (y - t.y)*(y - t.y)));
	}
```

```
public:
	Tochka(float xx = 0, float yy = 0)
	{
		x = xx; y = yy;
	}
	Tochka(const Tochka &t)
	{
		x = t.x; y = t.y;
	}
	void postaviKoordinati(float xx, float yy)
	{
		x = xx; y = yy;
	}
	/* ***преоптоварување на операторот за печатење. Истото се постигнува и со
функцијата
	pechati() ***
	friend ostream & operator<<(ostream & out, Tochka &t)
	{ return out << "(" << t.x << "," << t.y << ")"<<endl; }
	*/
	void pechati()
	{
		cout << endl << "(" << x << "," << y << ")";
	}
};
class Krug :public Tochka
{
protected:
	float r;

public:
	Krug(float xx = 0, float yy = 0, float rr = 1) :Tochka(xx, yy)
	{
		r = rr;
	}
	Krug(Tochka &t, float rr) :Tochka(t), r(rr) {}
	Krug(const Krug &kr) :Tochka(kr), r(kr.r) {}
	float ploshtina()
	{
		return r*r*PI;
	}
	void pechati()
	{
		Tochka::pechati();
		cout << " i radius:" << r << endl;
	}
};
class Sfera :protected Krug
{
private:
	float z;
public:
	Sfera(float xx, float yy, float zz, float rr) :Krug(xx, yy, rr)
	{
		z = zz;
	}
	Sfera(Krug &k, float zz) : Krug(k), z(zz) {}
	void postavi(float xx, float yy, float zz)
	{
		Tochka::postaviKoordinati(xx, yy); z = zz;
	}
	float ploshtina()
	{
		return 4 * Krug::ploshtina();
	}
```

```
        float volumen()
        {
                return 4 / 3 * Krug::ploshtina()*r;
        }

        void pechati()
        {
                Krug::pechati();
                cout << " i z:" << z << endl;
        }
};

int main()
{
        Tochka t1(5, 5);
        Krug k(t1, 3);
        Sfera s(k, 3);
        t1.pechati();
        k.pechati();
        s.pechati();
        k.postaviKoordinati(7, 7);
        k.pechati();
        t1 = k;
        t1.pechati();
        cout << endl;

        cout << k.ploshtina() << endl;
        cout << s.ploshtina() << endl;
        cout << s.volumen() << endl;
        return 0;
}
```