Laborator 11

```
/*  Lab 11  Popa Larisa-Ancuta  Prob 2

-la exemplul al treilea extindeți clasa de bază cu alte metode virtuale, redefinite în
clasele derivate, cum ar fi metode get( ) și set( )
   pentru greutatea vehiculului (variabila greutate)

*/

//main
#include <iostream>

using namespace std;

#include "Header.h"

int main()
{
	// apel direct, prin intermediul unor obiecte specifice
	Vehicul monocicleta;
	Automobil ford;
	Camion semi;
	Barca barca_de_pescuit;

	monocicleta.mesaj();
	ford.mesaj();
	semi.mesaj();//din Vehicul ca si CB
	barca_de_pescuit.mesaj();


	// apel prin intermediul unui pointer specific
	Vehicul *pmonocicleta;
	Automobil *pford;
	Camion *psemi;
	Barca *pbarca_de_pescuit;

	cout << "\n";

	pmonocicleta = &monocicleta;
	pmonocicleta->mesaj();
	pford = &ford;
	pford->mesaj();
	psemi = &semi;
	psemi->mesaj();//din CB
	pbarca_de_pescuit = &barca_de_pescuit;
	pbarca_de_pescuit->mesaj();


	// apel prin intermediul unui pointer catre un obiect al clasei de baza
	cout << "\n";
	pmonocicleta = &monocicleta;
	pmonocicleta->mesaj();//Vehicul
	pmonocicleta = &ford;//upcasting
	pmonocicleta->mesaj();//Automobil
	pmonocicleta = &semi;//upcasting
	pmonocicleta->mesaj();//Camion- Vehicul
	pmonocicleta = &barca_de_pescuit;//upcasting
```

```
	pmonocicleta->mesaj();//Barca

	float val;
	cout << "\n\nIntroduceti greutatea automobilului: ";
	cin >> val;
	pford->setGreutate(val);
	cout << pford->getGreutate();

	cout << "\n\nIntroduceti greutatea camionului: ";
	cin >> val;
	psemi->setGreutate(val);
	cout << psemi->getGreutate();

	cout << "\n\nIntroduceti greutatea barcii: ";
	cin >> val;
	pbarca_de_pescuit->setGreutate(val);
	cout <<  pbarca_de_pescuit->getGreutate() << endl << endl;

	return 0;
}

//header Header.h
#pragma once

class Vehicul
{
	int roti;
	float greutate;
public:
	virtual void mesaj()
	{
		cout << "Mesaj din clasa Vehicul\n";
	}

	virtual void setGreutate(float g) { greutate = g; }
	virtual  float  getGreutate() { return  greutate; }
};

class Automobil : public Vehicul
{
	int incarcatura_pasageri;
public:
	void mesaj() override
	{
		cout << "Mesaj din clasa Automobil\n";
	}

	float  getGreutate()
	{
		cout << "Greutate automobil:  ";
		return  Vehicul::getGreutate();
	}
};

class Camion : public Vehicul
{
	int incarcatura_pasageri;
	float incarcatura_utila;
```

```
public:
	int pasageri()
	{
		return incarcatura_pasageri;
	}

	float  getGreutate()
	{
		cout << "Greutate camion:  ";
		return  Vehicul::getGreutate();
	}
};

class Barca : public Vehicul
{
	int incarcatura_pasageri;
public:
	int pasageri()
	{
		return incarcatura_pasageri;
	}
	void mesaj() override
	{
		cout << "Mesaj din clasa Barca\n";
	}

	float  getGreutate()
	{
		cout << "Greutate barca:  ";
		return  Vehicul::getGreutate();
	}
};
```

```
/* Lab 11  Popa Larisa-Ancuta  Prob 3

-să se scrie un program C++ în care se definește o clasă Militar cu o metodă publică
virtuală sunt_militar( ) care indică apartenența la armată
-derivați clasa Militar pentru a crea clasa Soldat și clasa Ofiter
-derivați mai departe clasa Ofiter pentru a obtine clasele Locotenent, Colonel, Capitan,
General
-redefiniti metoda sunt_militar( ) pentru a indica gradul militar pentru fiecare clasa
specifica
-instantiati fiecare clasa Soldat, Locotenent,...,General, si apelati metoda
sunt_militar( )

*/

//main
#define _CRT_SECURE_NO_WARNINGS
#include <iostream>

using namespace std;

#include "Header.h"

int main()
```

```
{
	Militar militar;
	Soldat soldat;
	Ofiter ofiter;
	Locotenent locotenent;
	Capitan capitan;
	Colonel colonel;
	General general;

	cout << "\nClass Militar: ";
	militar.sunt_militar();

	cout << "\nClass Soldat: ";
	soldat.sunt_militar();

	cout << "\nClass Ofiter: ";
	ofiter.sunt_militar();

	cout << "\nClass Locotenent: ";
	locotenent.sunt_militar();

	cout << "\nClass Capitan: ";
	capitan.sunt_militar();

	cout << "\nClass Colonel: ";
	colonel.sunt_militar();

	cout << "\nClass General: ";
	general.sunt_militar();

	return 0;
}

//header Header.h
#pragma once

class Militar
{
public:
	virtual void sunt_militar()
	{
		cout << "Sunt militar!\n";
	}
};

class Soldat:public Militar
{
public:
	virtual void sunt_militar()
	{
		cout << "Sunt soldat!\n";
	}
};

class Ofiter:public Militar
{
public:
	virtual void sunt_militar()
```

```
	{
		cout << "Sunt ofiter!\n";
	}
};

class Locotenent :public Ofiter
{
public:
	virtual void sunt_militar()
	{
		cout << "Sunt locotenent!\n";
	}
};

class Colonel :public Ofiter
{
public:
	virtual void sunt_militar()
	{
		cout << "Sunt colonel!\n";
	}
};

class Capitan :public Ofiter
{
public:
	virtual void sunt_militar()
	{
		cout << "Sunt capitan!\n";
	}
};

class General :public Ofiter
{
public:
	virtual void sunt_militar()
	{
		cout << "Sunt general!\n";
	}
};
```

```
/*  Lab 11  Popa Larisa-Ancuta  Prob 4

-declarati o clasa Animal, care va contine o metoda pur virtuala, respira( ) si doua
metode virtuale manaca( ) si doarme( )
-derivati in mod public o clasa Caine si alta Peste, care vor defini metoda pur virtuala
-clasa Caine va redefini metoda mananca( ), iar Peste metoda doarme( )
-instantiati obiecte din cele doua clase si apelati metodele specifice
-definiti apoi un tablou de tip Animal, care va contine obiecte din clasele derivate,
daca e posibil
-daca nu, gasiti o solutie adecvata

*/
//main
#define _CRT_SECURE_NO_WARNINGS
#include <iostream>
```

```cpp
using namespace std;
#include "Header.h"

int main()
{
	Caine c;
	Peste p;

	cout << "\nClass Caine:";
	c.respira();
	c.mananca();
	c.doarme();

	cout << "\n\nClass Peste:";
	p.respira();
	p.mananca();
	p.doarme();


	Animal A[3];

	A[0] = c;
	A[1] = p;

	cout << "\n\nTablou Animal:\n" << "A[0]:";
	A[0].respira();
	A[0].mananca();
	A[0].doarme();

	cout << "\n\nA[1]:";
	A[1].respira();
	A[1].mananca();
	A[1].doarme();

	cout << endl << endl;
	return 0;
}

//header Header.h
#pragma once

class Animal
{
public:
	virtual void respira()
	{
		cout << "\nRespir!";
	}
	virtual void mananca()
	{
		cout << "\nMananc!";
	}
	virtual void doarme()
	{
		cout << "\nDorm!";
	}
};
```

```
class Caine :public Animal
{
public:
	virtual void respira()
	{
		cout << "\nRespir!(Caine)";
	}
	virtual void mananca()
	{
		cout << "\nMananc!(Caine)";
	}
};

class Peste :public Animal
{
public:
	virtual void respira()
	{
		cout << "\nRespir!(Peste)";
	}
	virtual void doarme()
	{
		cout << "\nDorm!(Peste)";
	}
};
```

```
/*  Lab 11  Popa Larisa-Ancuta  Prob 5

-definiti o clasa abstracta care contine 3 declaratii de metode pur virtuale pentru
concatenarea, intreteserea a doua siruri de caractere
   si inversarea unui sir de caractere primit ca parametru
-o subclasa implementeaza corpurile metodelor declarate in clasa de baza
-instantiati clasa derivata si afisati rezultatele aplicarii operatiilor implementate in
clasa asupra unor siruri de caractere citite de la tastatura
-examinati eroarea data de incercarea de a instantia clasa de baza

*/
//main
#define _CRT_SECURE_NO_WARNINGS
#include <iostream>

using namespace std;

#include "Header.h"

#define DIM 100

int main()
{
	Subclasa o;

	char s1[DIM], s2[DIM];

	cout << "Introduceti primul sir de caractere: ";
	cin >> s1;
```

```
	cout << "Introduceti al doilea sir de caractere: ";
	cin >> s2;

	cout << "\nConcatenarea sirurilor: " << o.concatenare(s1, s2);

	o.intersectie(s1, s2);

	o.invers(s1);
	o.invers(s2);
	cout << "\n\nPrimul sir inversat: " << s1 << "\nAl doilea sir inversat: " << s2 <<
endl << endl;

	return 0;
}

//header Header.h
#pragma once

class Clasa
{
public:
	virtual char* concatenare(char*, char*) = 0;
	virtual void intersectie(char*, char*) = 0;
	virtual void invers(char*) = 0;
};

class Subclasa
{
public:
	char* concatenare(char *s1, char *s2)
	{
		int dim = strlen(s1) + strlen(s2);
		char *s_new = new char[dim];

		strcpy(s_new, "");
		strcat(s_new, s1);
		strcat(s_new, s2);

		return s_new;
	}

	void intersectie(char *s1, char *s2)
	{

	}

	void invers(char *s)
	{
		int i;
		char *aux = new char[strlen(s)];

		strcpy(aux, s);

		for (i = strlen(aux) - 1;i >= 0; i--)
			*(s + strlen(s) - 1 - i) = *(aux + i);
	}
};
```

```
/* Lab 11  Popa Larisa-Ancuta  Prob 6

-definiti o clasa numita Record care stocheaza informatiile aferente unei melodii
(artist, titlu, durata)
-o clasa abstracta (Playlist) contine ca atribut privat un pointer spre un sir de obiecte
de tip inregistrare
-in constructor se aloca memorie pentru un numar de inregistrari definit de utilizator
-clasa contine metode accesor si mutator pentru datele componente ale unei inregistrari
si o metoda pur virtuala cu un parametru (abstracta)
   care poate ordona sirul de inregistrari dupa un anumit criteriu codat in valoarea
intreaga primita ca parametru
   (1=ordonare dupa titlu, 2=ordonare dupa artist, 3=ordonare dupa durata) I
-intr-o alta clasa (PlaylistImplementation) derivata din Playlist se implementeaza corpul
metodei abstracte de sortare
-in functia main( ), sa se instantieze un obiect din clasa PlaylistImplementation si apoi
sa se foloseasca datele si metodele aferente

*/

//main
#define _CRT_SECURE_NO_WARNINGS
#include <iostream>

using namespace std;

#include "Header.h"

int main()
{
	int i, n, d, optiune;
	char a[DIM], t[DIM];

	cout << "\nIntroduceti numarul de piese: ";
	cin >> n;

	PlaylistImplementation p(n);

	for (i = 0;i < n;i++)
	{
		cout << "\nIntroduceti numele artistului: ";
		cin >> a;
		cout << "Introduceti titlul piesei: ";
		cin >> t;
		cout << "Introduceti durata piesei: ";
		cin >> d;

		p.setTitlu(i, t);
		p.setArtist(i, a);
		p.setDurata(i, d);
	}

	cout << "\n\nOrdonare: \n  1 -> dupa titlu \n  2 -> dupa artist \n  3 -> dupa
durata \nSelectati varianta dorita: ";
	cin >> optiune;

	p.ordonare(n, optiune);

	for ( i = 0; i < 3; i++)
```

```
	{
		cout << "\nArtist: " << p.getArtist(i);
		cout << "\nPiesa: " << p.getTitlu(i);
		cout << "\nDurata: " << p.getDurata(i);
		cout << "\n\n";
	}

	return 0;
}

//header Header.h
#pragma once

const int DIM = 30;

class Record
{
public:
	char artist[DIM], titlu[DIM];
	int durata;
};

class Playlist
{
	Record *o;
public:
	Playlist(int n)
	{
		int i;

		o = new Record[n];

		for (i = 0; i < n; i++)
		{
			strcpy((o + i)->artist, "-");
			strcpy((o + i)->titlu, "-");
			(o + i)->durata = 0;
		}
	}

	void setArtist(int i, char *a) { strcpy((o + i)->artist, a); }
	void setTitlu(int i, char *t) { strcpy((o + i)->titlu, t); }
	void setDurata(int i, int val) { (o + i)->durata = val; }

	char* getArtist(int i) { return (o + i)->artist; }
	char* getTitlu(int i) { return (o + i)->titlu; }
	int getDurata(int i) { return (o + i)->durata; }

	virtual void ordonare(int n, int optiune) = 0;

};

class PlaylistImplementation:public Playlist
{
public:
	PlaylistImplementation(int n) :Playlist(n) { ; }

	void ordonare(int n, int optiune)
```

```
{
	int i, j;

	if (optiune == 1) //ordonare dupa titlu
	{
		for (i = 0; i < n - 1; i++)
		{
			for (j = i + 1; j < n; j++)
			{
				if (strcmp(getTitlu(i), getTitlu(j)) > 0)
				{
					Record aux;

					strcpy(aux.artist, getArtist(i));
					strcpy(aux.titlu, getTitlu(i));
					aux.durata = getDurata(i);

					setArtist(i, getArtist(j));
					setTitlu(i, getTitlu(j));
					setDurata(i, getDurata(j));

					setArtist(j, aux.artist);
					setTitlu(j, aux.titlu);
					setDurata(j, aux.durata);
				}
			}
		}
	}
	else if (optiune == 2) //ordonare dupa artist
	{
		for (i = 0; i < n - 1; i++)
		{
			for (j = i + 1; j < n; j++)
			{
				if (strcmp(getArtist(i), getArtist(j)) > 0)
				{
					Record aux;

					strcpy(aux.artist, getArtist(i));
					strcpy(aux.titlu, getTitlu(i));
					aux.durata = getDurata(i);

					setArtist(i, getArtist(j));
					setTitlu(i, getTitlu(j));
					setDurata(i, getDurata(j));

					setArtist(j, aux.artist);
					setTitlu(j, aux.titlu);
					setDurata(j, aux.durata);
				}
			}
		}
	}
	else if (optiune == 3) //ordonare dupa durata
	{
		for (i = 0; i < n - 1; i++)
		{
			for (j = i + 1; j < n; j++)
```

```
                        {
                                if (getDurata(i) > getDurata(j))
                                {
                                        Record aux;

                                        strcpy(aux.artist, getArtist(i));
                                        strcpy(aux.titlu, getTitlu(i));
                                        aux.durata = getDurata(i);

                                        setArtist(i, getArtist(j));
                                        setTitlu(i, getTitlu(j));
                                        setDurata(i, getDurata(j));

                                        setArtist(j, aux.artist);
                                        setTitlu(j, aux.titlu);
                                        setDurata(j, aux.durata);
                                }
                        }
                }
        }
        else
                cout << "\nValoarea introdusa nu este corecta!\n\n";
    }
};
```

```
/* Lab 11  Popa Larisa-Ancuta  Prob 7

-scrieți o aplicație C/C++ în care să implementați clasa de bază abstracta
PatrulaterAbstract avand ca atribute protected patru instante ale
   clasei de baza Punct (o pereche de coordonate x si y, accesori si mutatori)
reprezentand coordonatele colturilor patrulaterului
-declarați două metode membre pur virtuale pentru calculul ariei și perimetrului figurii
definite
-derivați clasa PatrulaterConcret care implementeaza metodele abstracte mostenite si care
contine o metoda proprie care determina daca patrulaterul
   este patrat, dreptunghi, patrulater oarecare (convex/concav)
-in programul principal instanțiați clasa derivata si apelați metodele implementate
-ariile se vor calcula functie de tipul patrulaterului
-la patrulaterul convex oarecare aria va fi data de urmatoarea formula care exprima aria
functie de laturile a, b, c, d, semiperimetrul s,
   si de diagonalele p, q: A=sqrt {(s-a)(s-b)(s-c)(s-d)-1/4(ac+bd+pq)(ac+bd-pq)}
-la patrulaterul concav se va determina doar perimetrul

*/

//main
#define _CRT_SECURE_NO_WARNINGS
#include <iostream>

using namespace std;

#include "Header.h"

int main()
{
       int x, y;
```

```
	cout << "\nIntroduceti coordonatele celor 4 puncte: \n";

	cout << "  Punct P1\nx= ";
	cin >> x;
	cout << "y= ";
	cin >> y;
	Punct p1;
	p1.setX(x);
	p1.setY(y);

	cout << "  Punct P2\nx= ";
	cin >> x;
	cout << "y= ";
	cin >> y;
	Punct p2;
	p2.setX(x);
	p2.setY(y);

	cout << "  Punct P3\nx= ";
	cin >> x;
	cout << "y= ";
	cin >> y;
	Punct p3;
	p3.setX(x);
	p3.setY(y);

	cout << "  Punct P4\nx= ";
	cin >> x;
	cout << "y= ";
	cin >> y;
	Punct p4;
	p4.setX(x);
	p4.setY(y);

	PatrulaterConcret p(p1, p2, p3, p4);

	cout << "\n\nPerimetru= " << p.perimetru();
	cout << "\n\nArie= " << p.arie();

	return 0;
}

//header Header.h
#pragma once

class Punct
{
	int x, y;
public:
	Punct()
	{
		x = 0;
		y = 0;
	}

	void setX(int val) { x = val; }
	void setY(int val) { y = val; }
```

```
	int getX() { return x; }
	int getY() { return y; }
};

class PatrulaterAbstract
{
protected:
	Punct p1, p2, p3, p4;
public:
	void setP1(Punct p) { p1.setX(p.getX()); p1.setY(p.getY()); }
	void setP2(Punct p) { p2.setX(p.getX()); p2.setY(p.getY()); }
	void setP3(Punct p) { p3.setX(p.getX()); p3.setY(p.getY()); }
	void setP4(Punct p) { p4.setX(p.getX()); p4.setY(p.getY()); }

	Punct getP1() { return p1; }
	Punct getP2() { return p2; }
	Punct getP3() { return p3; }
	Punct getP4() { return p4; }

	virtual double perimetru() = 0;
	virtual double arie() = 0;
};

class PatrulaterConcret :public PatrulaterAbstract
{
public:
	PatrulaterConcret() { ; }
	PatrulaterConcret(Punct p1, Punct p2, Punct p3, Punct p4)
	{
		setP1(p1);
		setP2(p2);
		setP3(p3);
		setP4(p4);
	}

	double distanta(Punct p1, Punct p2)
	{
		return sqrt((p1.getX()*p2.getX() + p1.getY()*p2.getY()));
	}

	double perimetru()
	{
		return distanta(p1, p2) + distanta(p2, p3) + distanta(p3, p4) +
distanta(p4, p1);
	}

	double arie()
	{
		double l1 = distanta(p1, p2), l2 = distanta(p2, p3), l3 = distanta(p3, p4),
l4 = distanta(p4, p1);

		if (l1 == l3 && l2 == l4)
		{
			cout << "\n\ndreptunghi";

			return l1*l2;
		}
		else if (l1 == l2 && l2 == l3 && l3 == l4 && l4 == l1)
```

```
                {
                        cout << "\n\npatrat";

                        return l1*l1;
                }
                else
                {
                        cout << "\n\npatrulater convex";
                        return 0;
                }

                return 0;
        }

};
```