MINISTERUL EDUCAȚIEI, CULTURII ȘI CERCETĂRII
AL REPUBLICII MOLDOVA
Universitatea Tehnică a Moldovei
Facultatea Calculatoare, Informatică și Microelectronică
Departamentul Informatică și Ingineria Sistemelor

Raport

pentru lucrarea de laborator Nr.5

la cursul de " Programarea orientata pe obiecte"

Efectuat: Studentul gr. SI-191 **Comanac Artiom**
Verificat: **Mititelu Vitalie**

Chișinău – 2020

LUCRARE DE LABORATOR NR. 5

Tema: Moștenirea multiplă

Scopul lucrării:

- studierea regulilor de determinare a moștenirii multiple
- studierea avantajelor și neajunsurilor moștenirii multiple
- probleme legate de utilizarea moștenirii multiple
- studierea rezolvării problemelor

Varianta 9

a) Să se creeze, o ierarhie de moștenire: hârtie, valori – acțiuni.

b) Să se creeze, o ierarhie de moștenire: obiect - hârtie, valori – acțiuni.

Realizarea punctului *a*

main.cpp

```
#include <iostream>

#include "head_a.h"

/*
	Comanac Artiom		SI-191
	LAB #5, a
*/


void main() {
	Stock google("Google", 300);
	Stock tesla("Tesla", 150);

	cout << "Title: " << google.getTitle() << endl;
	cout << "Value: " << google.getPrice() << endl;
	cout << "Paper size: " << google.getWidth() << "x" << google.getHeight() << endl
<< endl;

	cout << "Title: " << tesla.getTitle() << endl;
	cout << "Value: " << tesla.getPrice() << endl;
	cout << "Paper size: " << tesla.getWidth() << "x" << tesla.getHeight() << endl;
}
```

head_a.h

```
#pragma once
#include <string>

using namespace std;
```

```
//hartie
class Paper {
private:
	int width = 0;
	int height = 0;

public:
	Paper(int, int);

	void setSize(int, int);

	int getWidth();
	int getHeight();
};

//valori
class Value {
private:
	int price = 0;

public:
	Value(int);

	void setPrice(int);
	int getPrice();
};

//actiuni
class Stock : public Paper, public Value {
private:
	string title = "";

public:
	Stock(string title, int price);

	void setTitle(string);
	string getTitle();
};
```

functions_a.cpp

```
#include "head_a.h"

/*
	*Clasa Hartie*

	Constructor general
*/
Paper::Paper(int w, int h) {
	this->width = w;
	this->height = h;
}

void Paper::setSize(int w, int h) {
	this->width = w;
	this->height = h;
}

int Paper::getWidth() {
	return this->width;
```

```
}

int Paper::getHeight() {
	return this->height;
}

/*
	*Clasa Valori*

	Constructor general
*/
Value::Value(int price) {
	this->price = price;
}

void Value::setPrice(int price) {
	this->price = price;
}

int Value::getPrice() {
	return this->price;
}


/*
	*Clasa Actiuni*

	Constructor general (cu apelul constructori Harie si Valori)
*/
Stock::Stock(string title, int price): Paper(20, 10), Value(price) {
	this->title = title;
}

void Stock::setTitle(string title) {
	this->title = title;
}

string Stock::getTitle() {
	return this->title;
}
```

Demonstrarea și testarea

```
void main() {
    Stock google("Google", 300);
    Stock tesla("Tesla", 150);

    cout << "Title: " << google.getTitle() << endl;
    cout << "Value: " << google.getPrice() << endl;
    cout << "Paper size: " << google.getWidth() << "x" <

    cout << "Title: " << tesla.getTitle() << endl;
    cout << "Value: " << tesla.getPrice() << endl;
    cout << "Paper size: " << tesla.getWidth() << "x" <<
}
```

```
Консоль отладки Microsoft Visual Studio
Title: Google
Value: 300
Paper size: 20x10

Title: Tesla
Value: 150
Paper size: 20x10
```

Realizarea punctului *b*

main.cpp

```
#include <iostream>

#include "head_b.h"

/*
	Comanac Artiom	SI-191
	LAB #5, b
*/


void main() {
	Stock google("Google", 300);
	Stock tesla("Tesla", 150);

	cout << "Title: " << google.getTitle() << endl;
	cout << "Value: " << google.getPrice() << endl;
	cout << "Paper size: " << google.getWidth() << "x" << google.getHeight() << endl;
	cout << "Object [Paper] (Type / Physical): " << google.Paper::getType() << " / "
<< google.Paper::getPhysical() << endl;
	cout << "Object [Value] (Type / Physical): " << google.Value::getType() << " / "
<< google.Value::getPhysical() << endl << endl;

	cout << "Title: " << tesla.getTitle() << endl;
	cout << "Value: " << tesla.getPrice() << endl;
	cout << "Paper size: " << tesla.getWidth() << "x" << tesla.getHeight() << endl;
	cout << "Object [Paper] (Type / Physical): " << tesla.Paper::getType() << " / " <<
tesla.Paper::getPhysical() << endl;
	cout << "Object [Value] (Type / Physical): " << tesla.Value::getType() << " / " <<
tesla.Value::getPhysical() << endl;
}
```

head_b.h

```
#pragma once

#include <string>

using namespace std;

//obiect
class Object {
private:
	string type = "";
	bool physical = false;

public:
	Object(string, bool);

	void setType(string);
	string getType();

	void setPhysical(bool);
	bool isPhysical();
	string getPhysical();
};
```

```
//hartie
class Paper: public Object {
private:
	int width = 0;
	int height = 0;

public:
	Paper(int, int);

	void setSize(int, int);

	int getWidth();
	int getHeight();
};

//valori
class Value: public Object {
private:
	int price = 0;

public:
	Value(int);

	void setPrice(int);
	int getPrice();
};

//actiuni
class Stock : public Paper, public Value {
private:
	string title = "";

public:
	Stock(string title, int price);

	void setTitle(string);
	string getTitle();
};
```

functions_b.cpp

```
#include "head_b.h"

/*
	*Clasa Object*

	Constructor general
*/
Object::Object(string type, bool physical) {
	this->type = type;
	this->physical = physical;
}

void Object::setType(string type) {
	this->type = type;
}

string Object::getType() {
	return this->type;
}
```

```
void Object::setPhysical(bool physical) {
	this->physical = physical;
}

bool Object::isPhysical() {
	return this->physical;
}

string Object::getPhysical() {
	if (this->physical)
		return "Yes";
	else
		return "No";
}


/*
	*Clasa Hartie*

	Constructor general
*/
Paper::Paper(int w, int h): Object("hartie", true) {
	this->width = w;
	this->height = h;
}

void Paper::setSize(int w, int h) {
	this->width = w;
	this->height = h;
}

int Paper::getWidth() {
	return this->width;
}

int Paper::getHeight() {
	return this->height;
}


/*
	*Clasa Valori*

	Constructor general
*/
Value::Value(int price): Object("valori", false) {
	this->price = price;
}

void Value::setPrice(int price) {
	this->price = price;
}

int Value::getPrice() {
	return this->price;
}


/*
	*Clasa Actiuni*

	Constructor general (cu apelul constructori Harie si Valori)
*/
```

```
Stock::Stock(string title, int price) : Paper(20, 10), Value(price) {
	this->title = title;
}

void Stock::setTitle(string title) {
	this->title = title;
}

string Stock::getTitle() {
	return this->title;
}
```

Demonstrarea și testarea

```
int main() {
	Stock google("Google", 300);
	Stock tesla("Tesla", 150);

	cout << "Title: " << google.getTitle() << endl;
	cout << "Value: " << google.getPrice() << endl;
	cout << "Paper size: " << google.getWidth() << "x" <<
	cout << "Object [Paper] (Type / Physical): " << goog
	cout << "Object [Value] (Type / Physical): " << goog

	cout << "Title: " << tesla.getTitle() << endl;
	cout << "Value: " << tesla.getPrice() << endl;
	cout << "Paper size: " << tesla.getWidth() << "x" <<
	cout << "Object [Paper] (Type / Physical): " << tesl
	cout << "Object [Value] (Type / Physical): " << tesl
```

Консоль отладки Microsoft Visual Studio

```
Title: Google
Value: 300
Paper size: 20x10
Object [Paper] (Type / Physical): hartie / Yes
Object [Value] (Type / Physical): valori / No

Title: Tesla
Value: 150
Paper size: 20x10
Object [Paper] (Type / Physical): hartie / Yes
Object [Value] (Type / Physical): valori / No
```

Concluzii

Efectuând aceasta lucrarea de laborator au fost obținute cunoștințele in diferite domenii limbajului C++: moștenirea claselor, moștenirea multipla, reguli de determinare a moștenirilor multiple, avantajele si neajunsele moștenirii multiple, utilizarea constructorilor in clasele moștenite.