МИНЕСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

УЧЕРЕЖДЕНИЕ ОБРАЗОВАНИЯ

«Брестский государственный технический университет»

Кафедра «Интеллектуальные информационные технологии»

Лабораторная работа №6

По дисциплине «Объектно-ориентированное программирование и проектирование»

За 4 семестр

Тема: «Наследование и виртуальные функции»

Выполнила:

студентка 2 курса

группы АС-56

Карпенко М.В.

Проверил:

Давидюк Ю.И.

Брест 2021

Цель работы: Получить практические навыки создания иерархии классов и использования статических компонентов класса.

Вариант 4 - деталь, механизм, изделие, узел.

1. Определить иерархию классов (в соответствии с вариантом).

2. Определить в классе статическую компоненту - указатель на начало связанного списка объектов и статическую функцию для просмотра списка.

3. Реализовать классы.

4. Написать демонстрационную программу, в которой создаются объекты различных классов и помещаются в список, после чего список просматривается.

5. Сделать соответствующие методы не виртуальными и посмотреть, что будет.

6. Реализовать вариант, когда объект добавляется в список при создании, т.е. в конструкторе

Код программы:

```
#include <iostream>
#include <Windows.h>
#include <vector>

using namespace std;

//4) деталь, механизм, изделие, узел;

class Part
{
private:
	string name;
public:
	static Part* begin;
	static Part* end;
	Part* next;
	static void ShowList();
	Part();
	Part(string name);
	Part(const Part& part);
	string getName() const;
	friend ostream& operator<<(ostream& out, const Part& PN);
	void AddToList();
protected:
	virtual void printInformation(ostream& out)const = 0;
	virtual ~Part() {};
};

class Mechanism : public Part
{
private:
	int weight;
public:
	Mechanism() :Part()
	{
		weight = 0;
		AddToList();
	}
	Mechanism(string name, int age) : Part(name)
```

```
	{
		this->weight = age;
		AddToList();
	}
	Mechanism(const Mechanism& ST) : Part(ST.getName())
	{
		this->weight = ST.weight;
		AddToList();
	}
	~Mechanism() {};
protected:
	void printInformation(ostream& out) const;
};

void Mechanism::printInformation(ostream& out)const
{
	Part::printInformation(out);
	out << "Вес: " << weight;
}

class Unit :public Part
{
public:
	Unit() :Part()
	{
		consistOf = 0;
		AddToList();
	};
	Unit(string name, double partsNum) :Part(name)
	{
		this->consistOf = partsNum;
		AddToList();
	}
	Unit(const Unit& ST) :Part(ST.getName())
	{
		this->consistOf = ST.consistOf;
		AddToList();
	}
	~Unit() {};
	double getExperience() const;
protected:
	void printInformation(ostream& out) const;
	double consistOf;
};

void Unit::printInformation(ostream& out)const
{
	Part::printInformation(out);
	out << "Состоит из  " << consistOf << " деталей\n";
}

double Unit::getExperience() const
{
	return consistOf;
}

class Product :public Unit
{
private:
	int cost;
public:
	Product() : Unit()
	{
		cost = 0;
		AddToList();
	};
	Product(string name, int cost, double partsnum) :Unit(name, partsnum)
	{
```

```
                this->cost = cost;
                AddToList();
        }
        Product(Product& ST) :Unit(ST.getName(), ST.getExperience())
        {
                this->cost = ST.cost;
                AddToList();
        }
        ~Product() {};
protected:
        void printInformation(ostream& out) const;
};

void Product::printInformation(ostream& out)const
{
        Unit::printInformation(out);
        out << "Цена: " << cost << "$" << "\n";
}

ostream& operator<<(ostream& out, const Part& SH) {
        SH.printInformation(out);
        return out;
}

Part* Part::begin = nullptr;
Part* Part::end = nullptr;

void Part::AddToList()
{
        if (begin == nullptr)
        {
                begin = this;
                end = this;
        }
        else
        {
                end->next = this;
                end = this;
        }
}

Part::Part()
{
        name = "";
}

Part::Part(string name)
{
        this->name = name;
}

Part::Part(const Part& part)
{
        this->name = part.name;
}

string Part::getName() const
{
        return name;
}

void Part::printInformation(ostream& out)const
{
        out << "\n Название: " << name << "\n";
}

void Part::ShowList()
{
```

```
	Part* element = begin;
	for (int i = 0; i < 3; i++)
	{
		cout << *element << "\n";
		element = element->next;
	}
	cout << endl;
}

int main()
{
	setlocale(0, "");
	Part* arr[3];
	Mechanism something1("Болт", 20);
	Unit something2("Центральный узел", 32);
	Product something3("Карданный вал", 600, 22);
	arr[0] = &something1;
	arr[1] = &something2;
	arr[2] = &something3;
	Part::ShowList();
}
```

Результат работы программы:

```
 Название: Болт
Вес: 20

 Название: Центральный узел
Состоит из  32 деталей


 Название: Карданный вал
Состоит из  22 деталей
Цена: 600$
```

Вывод: Получила практические навыки создания иерархии классов и использования статических компонентов класса.