Министерство образования Республики Беларусь
Учреждение образования
«Брестский государственный технический университет»
Кафедра ИИТ

Лабораторная работа №4
по дисциплине: **ООТПиСТ**
Тема: НАСЛЕДОВАНИЕ И ВИРТУАЛЬНЫЕ ФУНКЦИИ

Выполнил
студент 2 курса
Корнасевич И. Д.

Проверил
Миндер А. В.

Брест 2021

Задание

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 (смотри пункт 6 следующего раздела)

У нас есть 2 класса: Person и Student. Student наследуется от Person.

Person.h

```
1  //
2  // Created by ivan on 28/4/21.
3  //
4
5  #ifndef SRC_PERSON_H
6  #define SRC_PERSON_H
7
8
9  #include <string>
10 #include <vector>
11 #include <ostream>
12
13 class Person {
14 private:
15     static std::vector<Person*> list;
16 protected:
17     std::string name;
18     int age = 0;
19 public:
20     Person(std::string name, int age);
21
22     const std::string &getName() const;
23
24     void setName(const std::string &name);
25
26     int getAge() const;
27
28     void setAge(int age);
29
30     virtual std::string toString() = 0;
31
32     static std::vector<Person*> getList();
33 };
34
35 #endif //SRC_PERSON_H
```

Person.cpp

```cpp
//
// Created by ivan on 28/4/21.
//

#include "Person.h"

#include <utility>

std::vector<Person*> Person::list= std::vector<Person*>();

const std::string &Person::getName() const {
    return name;
}

void Person::setName(const std::string &name) {
    Person::name = name;
}

int Person::getAge() const {
    return age;
}

void Person::setAge(int age) {
    Person::age = age;
}

std::vector<Person *> Person::getList() {
    return Person::list;
}

Person::Person(std::string name, int age) : name(std::move(name)),
   age(age) {
    Person::list.push_back(this);
}
```

Student.h

```cpp
//
// Created by ivan on 28/4/21.
//

#ifndef SRC_STUDENT_H
#define SRC_STUDENT_H


#include <ostream>
#include "Person.h"

class Student : Person{
private:
    double grade;
public:
    Student(const std::string &name, int age, double grade);

    double getGrade() const;
```

```
19
20      void setGrade(double grade);
21
22      std::string toString() override;
23  };
24
25
26  #endif //SRC_STUDENT_H
```

Student.cpp

```
1   //
2   // Created by ivan on 28/4/21.
3   //
4
5   #include <sstream>
6   #include "Student.h"
7
8   double Student::getGrade() const {
9       return grade;
10  }
11
12  void Student::setGrade(double grade) {
13      Student::grade = grade;
14  }
15
16  std::string Student::toString() {
17      std::stringstream ss;
18      ss << "Student name: " << this->name << " age: " << this->age <<
        " grade: " << this->grade << std::endl;
19      return ss.str();
20  }
21
22  Student::Student(const std::string &name, int age, double grade) :
        Person(name, age), grade(grade) {}
```

main.cpp

```
1   #include <iostream>
2   #include "Person.h"
3   #include "Student.h"
4   #include <sstream>
5
6   int main() {
7       Student a("name", 20, 5.6);
8       Student b("name", 19, 7.8);
9       Student c("name", 18, 9.0);
10      Student d("name", 31, 5.1);
11      for (auto q : Person::getList()){
12          std::cout << q->toString();
13      }
14  }
```

main.cpp

```
1   #include <iostream>
2   #include "Person.h"
```

```
3  #include "Student.h"
4  #include <sstream>
5
6  int main() {
7      Student a("name", 20, 5.6);
8      Student b("name", 19, 7.8);
9      Student c("name", 18, 9.0);
10     Student d("name", 31, 5.1);
11     for (auto q : Person::getList()){
12         std::cout << q->toString();
13     }
14 }
```

Вывод: В работе я построил небольшую иерархию классов (Person и Student). Я сделал класс Person абстрактным, определив один из его методов как чисто виртуальный. Я использовал static std::vector как private static компоненту класса.