|  |  |
| --- | --- |
| Gerb-BMSTU_01 | **Министерство науки и высшего образования Российской Федерации**  Калужский филиал  федерального государственного бюджетного  образовательного учреждения высшего образования  ***«Московский государственный технический университет имени Н.Э. Баумана (национальный исследовательский университет)»***  ***(КФ МГТУ им. Н.Э. Баумана)*** |

|  |  |
| --- | --- |
| **ФАКУЛЬТЕТ** | **ИУК «Информатика и управление»** |
| **КАФЕДРА** | **ИУК4 «Программное обеспечение ЭВМ,** |
| **информационные технологии»** | |

**ЛАБОРАТОРНАЯ РАБОТА №2**

**«Наследование и иерархия классов»**

**ДИСЦИПЛИНА: «Высокоуровневое программирование»**



|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Выполнил: студент гр. ИУК4-21Б | |  |  | ( | Суриков Н.С | ) |
|  |  |  | (подпись) |  | (Ф.И.О.) |  |
| Проверил: | |  |  | ( | Пчелинцева Н. И. | ) |
|  |  |  | (подпись) |  | (Ф.И.О.) |  |

|  |  |
| --- | --- |
| Дата сдачи (защиты):  Результаты сдачи (защиты): | |
|  | - Балльная оценка:  - Оценка: |

**Цель:** приобретение практических навыков проектирования классов и иерархии структур данных.

**Задачи:**

1. Изучить понятие иерархии структур данных;
2. Познакомиться со способами наследования классов;
3. Изучить понятие интерфейса;
4. Научиться выполнять инициализацию объекта разными способами.

**Условие задачи:**

*Задача 1*

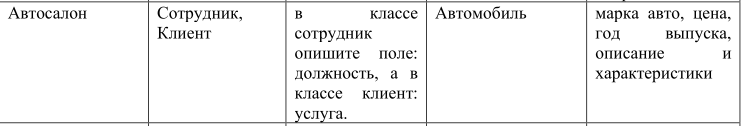
Создайте абстрактный класс “пользователь” с полями: имя, фамилия, возраст, логин и пароль. А также с чистыми виртуальными функциями, поведение которых определено по умолчанию.

*Задача 2*

Создайте папку для самостоятельной сущности (см. таблицу вариантов) и реализуйте в ней класс со следующими полями (см. столбец — Свойства). Добавьте папки для наследуемых сущностей (см. таблицу вариантов). В соответствующих папках опишите классы сущностей, унаследовав их от абстрактного класса “пользователь”. Дополнительно (см. столбец — Дополнительно) Реализуйте необходимости. Конструкторы классов, геттер/сеттер методы по

*Задача 3*

Реализуйте тестовую логику классов в файле main, чтобы оценить работоспособность приложения. Добавьте эту функциональность в меню.



**Листинг программы:**

Main.cpp:

1 *#include* *"models/CMenu/CMenu.h"*

2 *#include* *"models/CMenuItem/CMenuItem.h"*

3 *#include* *"models/Car/Car.h"*

4 *#include* *"models/Client/Client.h"*

5 *#include* *"models/Employee/Employee.h"*

6

7 using namespace std;

8

9 *#pragma region функции-заглушки*

10

11 int hello()

12 {

13 std::cout << std::endl;

14 std::cout << "Hello world!" << std::endl;

15 std::cout << std::endl;

16

17 return 3;

18 }

19

20 int smileFace()

21 {

22 std::cout << std::endl;

23 std::cout << " \*\*\*\*\* " << std::endl;

24 std::cout << " \* \* " << std::endl;

25 std::cout << "\* o o \*" << std::endl;

26 std::cout << "\* v \*" << std::endl;

27 std::cout << " \* \* " << std::endl;

28 std::cout << " \*\*\*\*\* " << std::endl;

29 std::cout << std::endl;

30

31 return 1;

32 }

33

34 int squareOf5()

35 {

36 std::cout << std::endl;

37 std::cout << "Square: " << 5 \* 5 << std::endl;

38 std::cout << std::endl;

39

40 return 2;

41 }

42

43 int testCar()

44 {

45 using namespace SNS;

46 Car car("Toyota", 25000.0, 2022, "Sedan", "Bluetooth, Backup Camera");

47 car.displayInfo();

48 return 1;

49 }

50

51 int testEmployee(){

52 using namespace SNS;

53 Employee employee("John", "Doe", 25, "johndoe", "12345", "Manager");

54 employee.displayPublicInfo();

55 employee.displayPrivateInfo();

56 return 1;

57 }

58

59 int testClient(){

60 using namespace SNS;

61 Client client("John", "Doe", 25, "johndoe", "12345", "Service");

62 client.displayPublicInfo();

63 client.displayPrivateInfo();

64 return 1;

65 }

66 *#pragma endregion*

67

68 const int ITEMS\_NUMBER = 6;

69

70 int main()

71 {

72 using namespace SNS;

73

74 CMenuItem items[ITEMS\_NUMBER]{

75 CMenuItem{"Смайлик)", smileFace},

76 CMenuItem("Квадрат 5", squareOf5),

77 CMenuItem("Привет мир", hello),

78 CMenuItem("Тест машины", testCar),

79 CMenuItem("Тест сотрудника", testEmployee),

80 CMenuItem("Тест клиента", testClient)};

81

82 CMenu menu("Главное меню", items, ITEMS\_NUMBER);

83 while (menu.isRun())

84 {

85 menu.runCommand();

86 };

87

88 return 0;

89 }

90

CMenuItem.cpp:

1 *#include* *"./CMenuItem.h"*

2

3 namespace SNS

4 {

5 CMenuItem::CMenuItem(std::string name, Func func) : item\_name(name), func(func)

6 {

7 }

8

9 std::string CMenuItem::getName()

10 {

11 return item\_name;

12 }

13

14 void CMenuItem::print()

15 {

16 std::cout << item\_name;

17 }

18

19 int CMenuItem::run()

20 {

21 return func();

22 }

23 } *// namespace SNS*

24

CMenu.h:

1 *#pragma once*

2

3 *#include* *"./CMenuItem.h"*

4 *#include* *<cstddef>*

5

6 namespace SNS

7 {

8 class CMenu

9 {

10 public:

11 CMenu(std::string, CMenuItem \*, std::size\_t);

12 int getSelect() const;

13 bool isRun() const;

14 std::string getTitle();

15 size\_t getCount() const;

16 CMenuItem \*getItems();

17 void print();

18 int runCommand();

19

20 private:

21 int select{-1};

22 size\_t count{};

23 bool running{};

24 std::string title{};

25 CMenuItem \*items{};

26 };

27 } *// namespace SNS*

28

CMenu.cpp:

1 *#include* *"./CMenu.h"*

2

3 namespace SNS

4 {

5 CMenu::CMenu(std::string title, CMenuItem \*items, size\_t count) : title(title), items(items), count(count)

6 {

7 }

8

9 int CMenu::getSelect() const

10 {

11 return select;

12 }

13

14 bool CMenu::isRun() const

15 {

16 return running;

17 }

18

19 size\_t CMenu::getCount() const

20 {

21 return count;

22 }

23

24 std::string CMenu::getTitle()

25 {

26 return title;

27 }

28

29 CMenuItem \*CMenu::getItems()

30 {

31 return items;

32 }

33

34 void CMenu::print()

35 {

36 for (size\_t i{}; i < count; ++i)

37 {

38 std::cout << i << ". ";

39 items[i].print();

40 std::cout << std::endl;

41 }

42 }

43

44 int CMenu::runCommand()

45 {

46 print();

47 std::cout << "\n Select >> ";

48 std::cin >> select;

49 return items[select].run();

50 }

51 } *// namespace SNS*

52

Car.h:

1 *#pragma once*

2 *#include* *<string>*

3

4 namespace SNS

5 {

6 class Car

7 {

8 private:

9 std::string brand;

10 double price;

11 int year;

12 std::string description;

13 std::string features;

14

15 public:

16 Car(const std::string, double, int, const std::string, const std::string);

17

18 void displayInfo();

19

20 std::string getBrand() const;

21 double getPrice() const;

22 int getYear() const;

23 std::string getDescription() const;

24 std::string getFeatures() const;

25

26 void setBrand(const std::string &brand);

27 void setPrice(double price);

28 void setYear(int year);

29 void setDescription(const std::string &description);

30 void setFeatures(const std::string &features);

31 };

32 } *// namespace SNS*

Car.cpp:

1 *#include* *"./Car.h"*

2 *#include* *<iostream>*

3

4 namespace SNS

5 {

6 Car::Car(const std::string brand, double price, int year, const std::string description, const std::string features)

7 : brand(brand), price(price), year(year), description(description), features(features)

8 {

9 }

10

11 void Car::displayInfo()

12 {

13 std::cout << "Brand: " << brand << std::endl;

14 std::cout << "Price: " << price << std::endl;

15 std::cout << "Year: " << year << std::endl;

16 std::cout << "Description: " << description << std::endl;

17 std::cout << "Features: " << features << std::endl;

18 }

19

20 std::string Car::getBrand() const

21 {

22 return brand;

23 }

24

25 double Car::getPrice() const

26 {

27 return price;

28 }

29

30 int Car::getYear() const

31 {

32 return year;

33 }

34

35 std::string Car::getDescription() const

36 {

37 return description;

38 }

39

40 std::string Car::getFeatures() const

41 {

42 return features;

43 }

44

45 void Car::setBrand(const std::string &brand)

46 {

47 this->brand = brand;

48 }

49

50 void Car::setPrice(double price)

51 {

52 this->price = price;

53 }

54

55 void Car::setYear(int year)

56 {

57 this->year = year;

58 }

59

60 void Car::setDescription(const std::string &description)

61 {

62 this->description = description;

63 }

64

65 void Car::setFeatures(const std::string &features)

66 {

67 this->features = features;

68 }

69

70 }

71

Client.h:

1 *#pragma once*

2 *#include* *"../User/User.h"*

3

4 namespace SNS

5 {

6 class Client : public User

7 {

8 public:

9 Client(std::string, std::string, int, std::string,

10 std::string, std::string);

11 std::string getService() const;

12 void setService(const std::string &service);

13 void displayPublicInfo();

14 void displayPrivateInfo();

15

16 protected:

17 std::string service;

18 };

19 }

Client.cpp:

1 *#include* *"Client.h"*

2

3 namespace SNS

4 {

5 Client::Client(std::string name, std::string surname, int age, std::string login,

6 std::string password, std::string service)

7 : User(name, surname, age, login, password), service(service)

8 {

9 }

10

11 void Client::displayPublicInfo(){

12 User::displayPublicInfo();

13 std::cout << "Service: " << service << std::endl;

14 }

15

16 void Client::displayPrivateInfo(){

17 User::displayPrivateInfo();

18 }

19

20 std::string Client::getService() const

21 {

22 return service;

23 }

24

25 void Client::setService(const std::string &service)

26 {

27 this->service = service;

28 }

29 }

User.h:

1 *#pragma once*

2

3 *#include* *<iostream>*

4 namespace SNS

5 {

6 class User

7 {

8 public:

9 User(std::string m\_name, std::string, int, std::string, std::string);

10 std::string m\_name;

11 std::string m\_surname;

12 int m\_age;

13 std::string m\_login;

14 std::string m\_password;

15

16 virtual void displayPublicInfo() = 0;

17 virtual void displayPrivateInfo() = 0;

18 };

19 }

User.cpp:

1 *#include* *"./User.h"*

2 *#include* *<iostream>*

3

4 namespace SNS

5 {

6 User::User(std::string name, std::string surname, int age, std::string login, std::string password) : m\_name(name), m\_surname(surname), m\_age(age), m\_login(login), m\_password(password)

7 {

8 }

9 void User::displayPublicInfo()

10 {

11 std::cout << "Name: " << m\_name << std::endl;

12 std::cout << "Surname: " << m\_surname << std::endl;

13 std::cout << "Age: " << m\_age << std::endl;

14 }

15 void User::displayPrivateInfo()

16 {

17 std::cout << "Login: " << m\_login << std::endl;

18 std::cout << "Password: " << m\_password << std::endl;

19 }

20 }

Employee.h:

1 *#pragma once*

2 *#include* *"../User/User.h"*

3

4 namespace SNS

5 {

6 class Employee : public User

7 {

8 public:

9 Employee(std::string name, std::string surname, int age, std::string login,

10 std::string password, std::string post);

11 std::string getPost() const;

12 void setPost(const std::string &post);

13 void displayPublicInfo();

14 void displayPrivateInfo();

15

16 protected:

17 std::string post;

18 };

19 }

Employee.cpp:

1 *#include* *"Employee.h"*

2

3 namespace SNS

4 {

5

6 Employee::Employee(std::string name, std::string surname, int age, std::string login,

7 std::string password, std::string post)

8 : User(name, surname, age, login, password), post(post)

9 {

10 }

11

12 std::string Employee::getPost() const

13 {

14 return post;

15 }

16

17 void Employee::setPost(const std::string &post)

18 {

19 this->post = post;

20 }

21

22 void Employee::displayPublicInfo()

23 {

24 User::displayPublicInfo();

25 std::cout << "Post: " << post << std::endl;

26 }

27

28 void Employee::displayPrivateInfo()

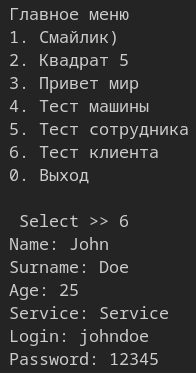
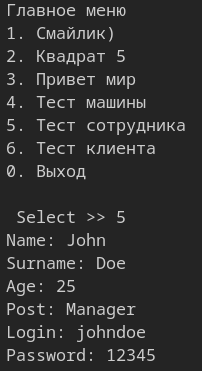
29 {

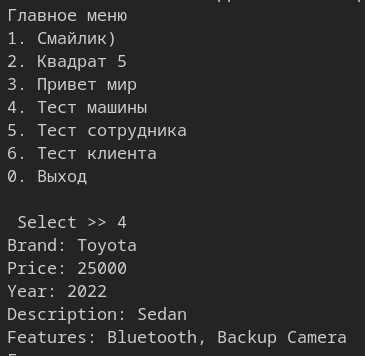
30 User::displayPrivateInfo();

31 }

32 }

**Результат работы:**

** **



**Вывод:** в ходе выполнения лабораторной работы были получены практические навыки проектирования классов и иерархии структур данных, средствами языка C++.