Учреждение образования
«БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ»

Кафедра интеллектуальных информационных технологий

Отчет по лабораторной работе №2 по курсу «Проектирование программ в интеллектуальных системах»

Выполнил студент группы 921703 **Кравцов М.С.**

Проверил: **Садовский М.Е.**

МИНСК 2020

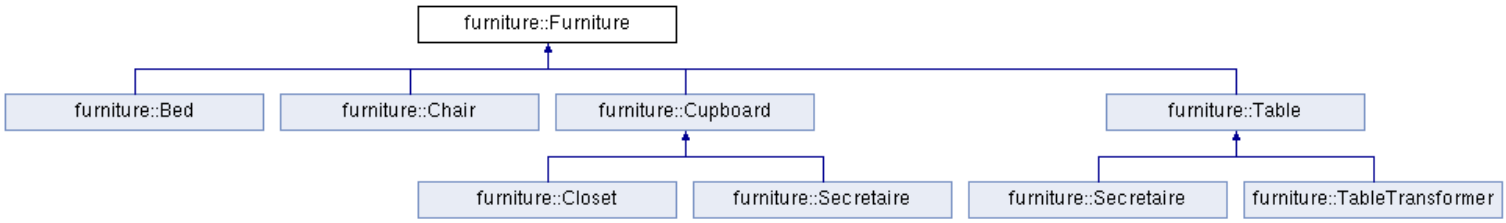
Тема:
Изучение возможных принципов наследования при объектно-ориентированного программирования с использованием языка C++

Цель работы:
Изучить принцип наследования ООП, используя возможности языка C++;

Задание: выбрать одну из представленных предметных областей. В рамках выбранной области необходимо построить иерархию классов с применением следующих концепций ООП:
• Закрытое
• Открытое
• Защищённое
• Виртуальное методы
• Разрешение имен с помощью using-директивы
• множественное наследование
• виртуальное наследование

Вариант 8:
Мебель;

Выполнение задания:



1)Furniture

```
class Furniture
{
public:
    Furniture(const double& price, const string& material, const string& name,
        const string& manufacturer, const double& length, const double& widht, const double& height);

    virtual ~Furniture() = default;

    Furniture& setPrice(const double& price);

    Furniture& setMaterial(const string& material);

    Furniture& setHeight(const double& height);

    Furniture& setName(const string& name);

    Furniture& setManufacturer(const string& manufacturer);

    Furniture& setWidth(const double& width);

    Furniture& setLenght(const double& length);

    double getPrice() const;

    string getMaterial() const;

    string getName() const;

    string getManufacturer() const;

    double getHeight() const;

    double getWidth() const;

    double getLenght() const;

    virtual double usefulSpace() const = 0;

    virtual void info() const;

protected:
    double check(const double& num, const string& name, const double& max) const;
    virtual void specificInfo() const = 0;

private:
    double price;
    string material;
    string name;
    string manufacturer;
    double height;
    double width;
    double length;
};
```

2)Bed

```
class Bed :virtual public Furniture
{
public:
    Bed(const double& price, const string& material, const string& name,
        const string& manufacturer, const double& length, const double& widht, const double& height,
        const int& numberOfLegs, const int& numberOfSleepingPlaces);

    ~Bed()override = default;

    Bed& setNumberOfLegs(const int& numberOfLegs);

    Bed& setNumberOfSleepingPlaces(const int& numberOfSleepingPlaces);

    int getNumberOfLegs() const;

    int getNumberOfSleepingPlaces() const;

    double usefulSpace() const override;

    void info() const override;

protected:
    void specificInfo() const override;

private:
    int numberOfLegs;
    int numberOfSleepingPlaces;
};
```

3)Chair

```
class Chair :virtual public Furniture
{
public:
    Chair(const double& price, const string& material, const string& name,
        const string& manufacturer, const double& length, const double& widht, const double& height,
        const int& numberOfLegs, const string& materialOfSeatPad, const string& materialOfBack);

    ~Chair() override = default;

    Chair& setNumberOfLegs(const int& numberOfLegs);

    Chair& setMaterialOfSeatPad(const string& materialOfSeatPad);

    Chair& setMaterialOfBack(const string& materialOfBack);

    int getNumberOfLegs() const;

    string getMaterialOfSeatPad() const;

    string getMaterialOfBack() const;

    virtual double usefulSpace() const override;

    virtual void info() const override;

protected:
    virtual void specificInfo() const override;

private:
    int numberOfLegs;
    string materialOfSeatPad;
    string materialOfBack;
};
```

4)Cupboard

```
class Cupboard :virtual public Furniture
{
public:
    Cupboard(const double& price, const string& material, const string& name,
        const string& manufacturer, const double& length, const double& widht,
        const double& height, const int& numberOfShelves, const double& lengthOfShelves);

    ~Cupboard() override = default;

    Cupboard& setNumberOfShelves(const int& numberOfShelves);

    Cupboard& setLengthOfShelve(const double& lengthOfShelve);

    int getNumberOfShelves() const;

    double getLengthOfShelve() const;

    double usefulSpace() const override;

    void info() const override;

protected:
    void specificInfo() const override;

private:
    int numberOfShelves;
    double lengthOfShelve;
};
```

5)Table

```
class Table :virtual public Furniture
{
public:
    Table(const double& price, const string& material, const string& name,
        const string& manufacturer, const double& length, const double& widht,
        const double& height, const int& numberOfLegs, const int& numberOfShuffles);

    ~Table()override = default;

    Table& setNumberOfLegs(const int& numberOfLegs);

    Table& setNumberOfShuffles(const int& numberOfShuffels);

    int getNumberOfLegs() const;

    int getNumberOfShuffles() const;

    double usefulSpace() const override;

    void info() const override;

protected:
    void specificInfo() const override;

private:
    int numberOfLegs;
    int numberOfShuffles;
};
```

6)Closet

```
class Closet : virtual public Cupboard
{
public:
    Closet(const double& price, const string& material, const string& name,
        const string& manufacturer, const double& length, const double& widht, const double& height,
        const int& numberOfShelves, const double& lengthOfShelve,
        const double& capacity, const int& numberOfDoors);

    ~Closet() override = default;

    Closet& setCapacity(const double& capacity);

    Closet& setNumberOfDoors(const int& numberOfDoors);

    double getCapacity() const;

    int getNumberOFDoors() const;

    double usefulSpace() const override;

    void info() const override;

protected:
    void specificInfo() const override;

private:
    double capacity;
    int numberOfDoors;
};
```

7)Secretaire

```
class Secretaire :public Cupboard, public Table
{
public:
    Secretaire(const double& price, const string& material, const string& name,
        const string& manufacturer, const double& length, const double& widht, const double& height,
        const int& numberOfShelves, const double& lengthOfShelves,
        const int& numberOfLegs, const int& numberOfShuffles);

    ~Secretaire()override = default;

    double usefulSpace() const override;

    void info() const override;

protected:
    void specificInfo() const override {};
};
```

8)TableTransformer

```
class TableTransformer :virtual public Table
{

public:
    TableTransformer(const double& price, const string& material, const string& name,
        const string& manufacturer, const double& length, const double& widht,
        const double& height, const int& numberOfLegs, const int& numberOfShuffles,
        const int& decompositionCoefficient);

    ~TableTransformer()override = default;

    TableTransformer& setDecompositionCoefficient(const int& decompositionCoefficient);

    int getDecompositionCoefficient() const;

    double usefulSpace() const override;

    void info() const override;

protected:
    void specificInfo() const override;

private:
    int decompositionCoefficient;

};
```

9)Printer (для печати информации о каждом типе мебели, используя указатель на базовый класс)

```
class Printer
{
public:
    void printInfo(const vector<const Furniture*>& furniture) const;
};
```

Вывод: в процессе выполнения данной лабораторной работы я изучил основы наследования на примере языка C++: виды наследования, using-директива. Результатом выполнения данной работы является программа, которая создает фрагмент иерархии мебели средствами языка C++.