Лабораторная работа №1 по предмету: «Объектно-ориентированные технологии программирования и стандарты проектирования

Сделал: студент группы 351001 Будников Михаил.

Задание: Построить иерархию классов на выбранную тематику (не менее 6), вывести информацию о них в консоль. Распределить классы по модулям. Создать список объектов в виде отдельного класса. В главном модуле визуализировать классы.

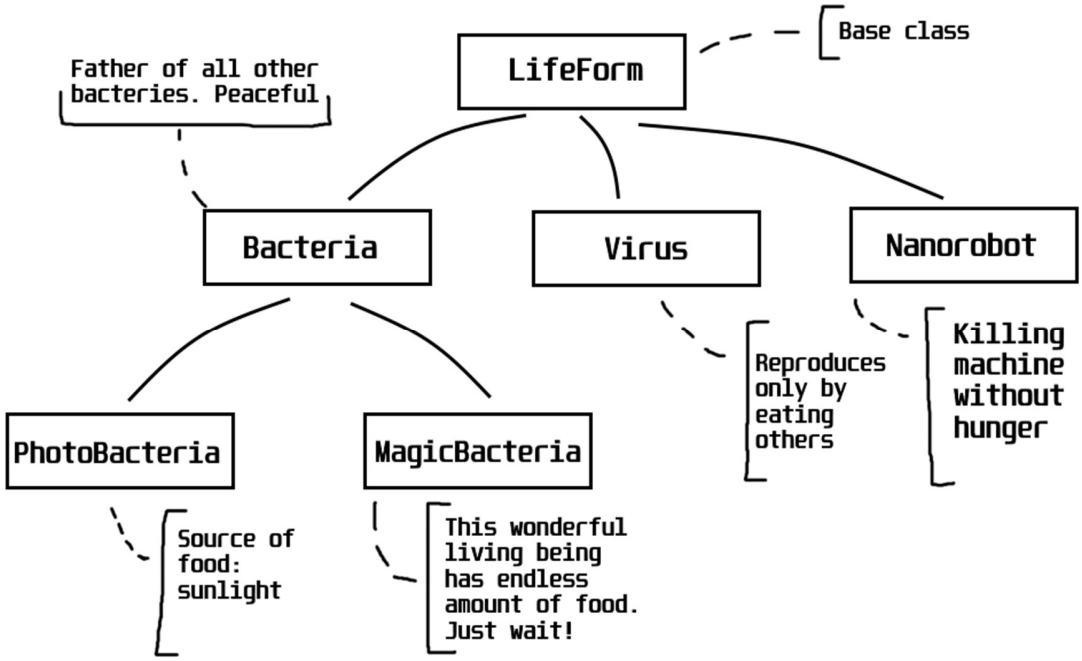

Рисунок 1 – Иерархия классов

Объявление базового класса **LifeForm**:

```
class LifeForm
{
public:
    LifeForm();

    bool isAlive();
    std::string & name();
    int maxHp();
    int hp();

    virtual void Draw(QPainter & painter) = 0;

    // necessary method in future for every inheritant to implement
    // virtual void onTick() = 0; // Some EnvironmentInfo variable will be
passed here

protected:
    std::string name_;
```

```
    QPointF coord_;
    int maxHp_;
    int hp_;
    int size_;
    int speed_;
    bool isAlive_;
};
```

Реализация:

```
#include "lifeform.h"

LifeForm::LifeForm() {}

bool LifeForm::isAlive()
{
    return isAlive_;
}
std::string &LifeForm::name()
{
    return name_;
}
int LifeForm::maxHp()
{
    return maxHp_;
}
int LifeForm::hp()
{
    return hp_;
}
```

Объявление класса-наследника **Bacteria:**

```
class Bacteria : public LifeForm
{
public:
    Bacteria(float x, float y);

    virtual void Draw(QPainter & painter) override;
    virtual void Reproduce();
    virtual void Eat();

protected:
    Bacteria() {} // for future descendants

    int hunger_;
};
```

Реализация:

```
#include "bacteria.h"
#include "random.h"

Bacteria::Bacteria(float x, float y)
{
```

```
    name_ = "Bacteria";
    coord_ = QPoint(x, y);
    size_ = rnd::Randint(10,13);
    maxHp_ = 10;
    hp_ = 10;
    hunger_ = 100;
}

void Bacteria::Draw(QPainter &painter)
{
    painter.save();

    painter.translate(coord_);
    painter.rotate(45); // implement in future when there will be movement
vector
    painter.setBrush(Qt::gray);
    painter.drawEllipse(QPointF(0, 0), 2 * size_, size_);

    painter.restore();
}

void Bacteria::Reproduce() {} // In Future
void Bacteria::Eat() {}
```