

Принципы объектно-ориентированного проектирования

POP3

Assembler

POSIX

C/C++

HTTP

CSS

SQL

XML

SMTP

Java

WinAPI

ActionScript

FTP

PHP

VCL

JavaScript

HTML

VBA

POP3

Assembler

POSIX

HTTP

CSS

C/C++

SQL

XI

ITP

Java

WinAPI



ActionScript

PHP

VCL

FTP

JavaScript

VBA

HTML

**императивное
программирование**

**объектно-ориентированное
программирование**

**функциональное
программирование**

**структурное
программирование**

**процедурное
программирование**



императивное программирование



структурное
программирование



процедурное
программирование

е
ие

Принципы объектно-ориентированного программирования

- Принцип единственной ответственности
- Принцип открытости / закрытости
- Принцип подстановки Лисков
- Принцип изоляции интерфейса
- Принцип инверсии зависимостей

Принцип единственной ОТВЕТСТВЕННОСТИ



Только потому, что Вы можете, еще не значит, что вы должны это сделать

Пример

```
class Network {  
    byte[] receiveByTcp() {  
    }  
  
    byte[] receiveByUdp() {  
    }  
  
    void sendByTcp(byte[] data) {  
    }  
  
    void sendByUdp(byte[] data) {  
    }  
}
```


Пример

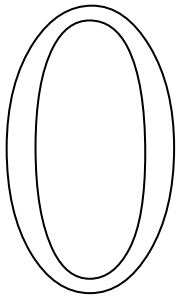
```
class TcpNetwork {  
    byte[] receive() {  
    }  
  
    void send(byte[] data) {  
    }  
}
```

```
class UdpNetwork {  
    byte[] receive() {  
    }  
  
    void send(byte[] data) {  
    }  
}
```

Пример

```
class TcpNetworkReceiver {  
    byte[] receive() {  
    }  
}  
class TcpNetworkSender {  
    void send(byte[] data) {  
    }  
}  
class UdpNetworkReceiver {  
    byte[] receive() {  
    }  
}  
class UdpNetworkSender {  
    void send(byte[] data) {  
    }  
}
```

Принцип открытости\закрытости



Для того, чтобы одеть пальто, не нужно вскрывать грудную клетку

Пример

```
class TcpNetwork {  
  
    byte[] receive() {  
    }  
  
    void send(byte[] data) {  
    }  
  
}
```

Пример

```
class TcpNetwork {  
    byte[] receive() {  
    }  
  
    void send(byte[] data) {  
    }  
  
    String receiveString() {  
        return new String(receive(), "UTF-8");  
    }  
  
    void send(String data) {  
        send(data.getBytes("UTF-8"));  
    }  
}
```

Пример

```
class StringTcpNetwork extends TcpNetwork {  
  
    String receiveString() {  
        return new String(receive(), "UTF-8");  
    }  
  
    void send(String data) {  
        send(data.getBytes("UTF-8"));  
    }  
  
}
```

Принцип подстановки Лисков



L

Если оно выглядит, как утка, квакает, как утка, но требует батарейки, возможно, у Вас проблема с абстракцией

Пример

```
public class Fibonacci {
```

```
    int a = 0;
```

```
    int b = 1;
```

```
    int getNumber() {
```

```
        int c = a + b;
```

```
        a = b;
```

```
        b = c;
```

```
        return b;
```

```
    }
```

```
}
```


Пример

```
public class Main {  
  
    public static void main(String[] args) {  
  
        Fibonacci f = new Fibonacci();  
  
        for(int n = 1; n <= 50; n++) {  
            int m = f.getNumber();  
            System.out.println(m);  
        }  
    }  
}
```

Пример

```
public class Fibonacci implements Iterator {
```

```
int a = 0;
```

```
int b = 1;
```

```
int next() {
```

```
int c = a + b;
```

```
a = b;
```

```
b = c;
```

```
return b;
```

```
}
```

```
}
```

Пример

```
public class Main {  
  
    public static void main(String[] args) {  
  
        Fibonacci f = new Fibonacci();  
  
        while(f.hasNext()) {  
            int m = f.next();  
            System.out.println(m);  
        }  
  
    }  
  
}
```

Пример

```
public class Fibonacci implements Iterator {  
    /*...*/  
    int next() { /*...*/ }  
    boolean hasNext() { /*...*/ }  
    void setAmount(int n) { /*...*/ }  
}
```

Пример

```
public class Main {  
    public static void main(String[] args) {  
  
        Fibonacci f = new Fibonacci();  
        f.setAmount(50);  
  
        while(f.hasNext()) {  
            int m = f.next();  
            System.out.println(m);  
        }  
    }  
}
```

Пример

```
public class Fibonacci implements Iterator,  
                                Iterable {  
    /*...*/  
    int next() { /*...*/ }  
    boolean hasNext() { /*...*/ }  
    void setAmount(int n) { /*...*/ }  
  
    Iterator iterator() {  
        return this;  
    }  
}
```

Пример

```
public class Main {  
    public static void main(String[] args) {  
  
        Fibonacci f = new Fibonacci();  
        f.setAmount(50);  
  
        for(int m : f) {  
            System.out.println(m);  
        }  
    }  
}
```

Пример

```
public class Fibonacci implements Iterator,  
                           Iterable {
```

```
    Fibonacci(int n) { /*...*/ }
```

```
    /*...*/
```

```
    int next() { /*...*/ }
```

```
    boolean hasNext() { /*...*/ }
```

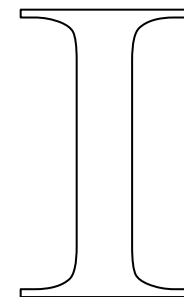
```
    Iterator iterator() {
```

```
        return this;
```

```
    }
```

```
}
```


Принцип изоляции интерфейса



Вы хотите чтобы я подключил это? Куда?

Пример

```
public class Fibonacci
    implements Iterator<Integer> {

    public boolean hasNext() {
        return false;
    }

    public Integer next() {
        return null;
    }

    public void remove() {
    }

}
```

Пример

```
interface Matrix {  
    public int size();  
    public double get(int i, int j);  
    public void set(int i, int j, double value);  
}
```

Пример

```
public class UsualMatrix implements Matrix {  
    private double a[][];  
    public UsualMatrix(int size) {  
        a = new double[size][size];  
    }  
    public int size() {  
        return a.length;  
    }  
    public double get(int i, int j) {  
        return a[i][j];  
    }  
    public void set(int i, int j, double value) {  
        a[i][j] = value;  
    }  
}
```

Пример

```
public class SimmMatrix implements Matrix {  
    private double a[][];  
    public SimmMatrix(int size) {  
        a = new double[size][];  
        for(int i = 0; i < size; i++) {  
            a[i] = new double[i+1];  
        }  
    }  
    public int size() {  
        return a.length;  
    }  
}
```

Пример

```
public double get(int i, int j) {  
    if(i < j) {  
        return a[i][j];  
    } else {  
        return a[j][i];  
    }  
}  
  
public void set(int i, int j, double value) {  
    if(i < j) {  
        a[i][j] = value;  
    } else {  
        a[j][i] = value;  
    }  
}  
}
```

Пример

```
public class EMatrix implements Matrix {  
    private int size;  
    public EMatrix(int size) {  
        this.size = size;  
    }  
    public int size() {  
        return size;  
    }  
    public double get(int i, int j) {  
        return i == j ? 1 : 0;  
    }  
    public void set(int i, int j, double value) {  
    }  
}
```

Пример

```
public class SummMatrix implements Matrix {
    private Matrix a, b;
    public SummMatrix(Matrix a, Matrix b) {
        this.a = a;
        this.b = b;
    }
    public int size() {
        return a.size;
    }
    public double get(int i, int j) {
        return a.get(i, j) + b.get(i, j);
    }
    public void set(int i, int j, double value) {
    }
}
```


Пример

```
interface Matrix {
```

```
    public int size();
```

```
    public double get(int i, int j);
```

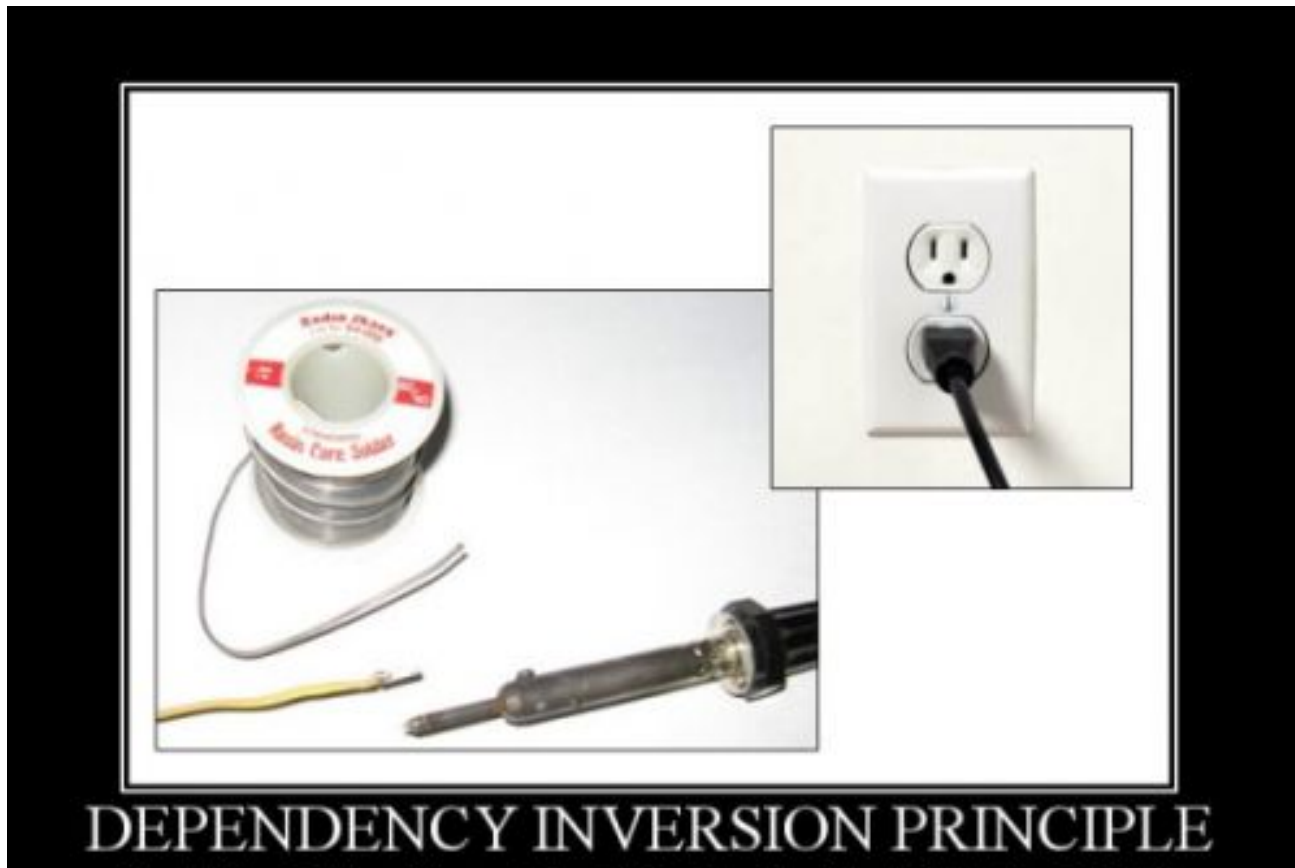
```
}
```

```
interface MutableMatrix extends Matrix {
```

```
    public void set(int i, int j, double value);
```

```
}
```

Принцип инверсии зависимостей



D

Будете ли Вы подключать лампочку непосредственно к электропроводке в стене?

Пример

```
public class MyApplet extends JApplet {  
    public paint(Graphics g) {  
        /*...*/  
    }  
}
```

Пример

```
public class MyMatrix {  
    private int r[][];  
    private int g[][];  
    private int b[][];  
    public MyMatrix(int w, int h) {  
        r = new int[h][w];  
        g = new int[h][w];  
        b = new int[h][w];  
    }  
    /*...*/  
}
```

Пример

```
public class Point {  
    public int r, g, b;  
}  
public class PictureMatrix {  
    private Point p[][];  
    public PictureMatrix(int w, int h) {  
        p = new Point[h][w];  
    }  
    /*...*/  
}
```

Пример

```
public class Point {
    public int r, g, b;
}
interface PictureMatrix {
    Point get(int i, int j);
    int width();
    int height();
}
public class PictureMatrixImpl1
    implements PictureMatrix {
    /*...*/
}
public class PictureMatrixImpl2
    implements PictureMatrix {
    /*...*/
}
```

Пример

```
public class MyApplet extends JApplet {
    private PictureMatrix pm = null;
    public void paint(Graphics g) {
        for(int i = 0; i < pm.height(); i++) {
            for(int j = 0; j < pm.height(); j++) {
                out(pm.get(i, j).r,
                    pm.get(i, j).g,
                    pm.get(i, j).b);
            }
        }
    }
    public void init() {
        pm = new PictureMatrixImpl1();
    }
}
```