Введение в ООП

Алексей Владыкин

OBJECT-ORIENTED SOFTWARE CONSTRUCTION

SECOND EDITION

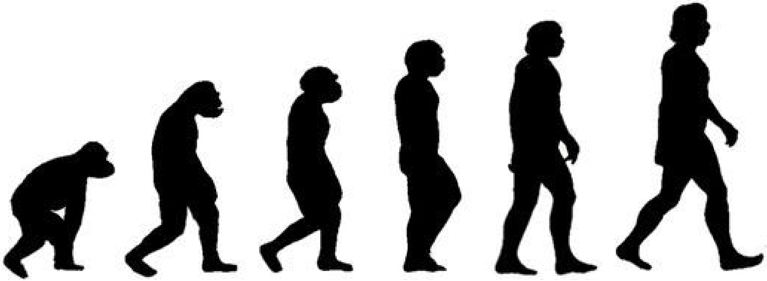
CD-ROM INCLUDED

The Most Comprehensive, Definitive O-O Reference Ever Published

An O-O *Tour de Force* by a Pioneer in the Field

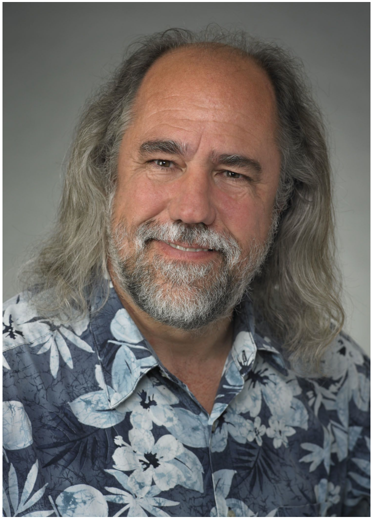
CD-ROM Includes Complete Hypertext Version of Book AND Object-Oriented Development Environment

BERTRAND MEYER



Объект — это мыслимая или реальная сущность, обладающая характерным поведением и отличительными характеристиками и являющаяся важной в предметной области.

Гради Буч



```java
Client client = new Client();

Contract contract = new Contract();

Order order = new Order();
```

```
Client client = new Client();

client.setName("Vasily Poupkine");

client.setBirthDate(new LocalDate(1990, 6, 13));

client.setAddress("Uryupinsk");

Order[] orders = client.getOrders();
```

```
Order order = getOrder();

order.paymentReceived();

order.sendToDelivery();

order.cancel();
```

```java
Client client = new VIPClient();

client.getDeliveryPrice();
// actually VIPClient.getDeliveryPrice()
// is executed
```

```java
double[][] m1 = new double[3][3];
double[][] m2 = new double[3][3];

// multiply
int m = m1.length;
int n = m2[0].length;
int o = m2.length;
double[][] res = new double[m][n];
for (int i = 0; i < m; i++) {
  for (int j = 0; j < n; j++) {
    for (int k = 0; k < o; k++) {
      res[i][j] += m1[i][k] * m2[k][j];
    }
  }
}
```

```java
Matrix m1 = new FullMatrix(new double[3][3]);

Matrix m2 = new DiagonalMatrix(new double[3]);

// multiply
int m = m1.getHeight();
int n = m2.getWidth();
int o = m2.getHeight();
double[][] res = new double[m][n];
for (int i = 0; i < m; i++) {
  for (int j = 0; j < n; j++) {
    for (int k = 0; k < o; k++) {
      res[i][j] += m1.get(i, k) * m2.get(k, j);
    }
  }
}
Matrix m3 = new FullMatrix(res);
```

Пакеты

```
package org.stepic.java;

public class HelloWorld {
    // ...
}
```

Пакеты

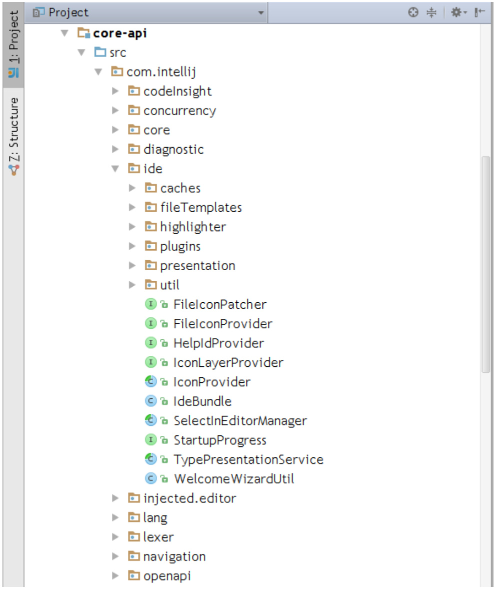
```java
package org.stepic.java.other;

import org.stepic.java.HelloWorld;

import java.util.*;



import static java.lang.Math.sqrt;

import static java.lang.System.out;


public class OtherClass {
    // ...
}
```

Пакеты стандартной библиотеки

- `java.lang`
- `java.io`
- `java.nio`
- `java.math`
- `java.time`
- `java.util`
- `java.util.regex`
- `javax.xml`
- ...

Пакеты для стороннего кода

- `org.stepic.java`
- `com.google.common`
- `org.apache.maven`
- `com.intellij.idea`
- `net.sf.json`
- `io.netty`
- ...



```
▼ core-api
  ▼ src
    ▼ com.intellij
      ▶ codeInsight
      ▶ concurrency
      ▶ core
      ▶ diagnostic
      ▼ ide
        ▶ caches
        ▶ fileTemplates
        ▶ highlighter
        ▶ plugins
        ▶ presentation
        ▶ util
          FileIconPatcher
          FileIconProvider
          HelpIdProvider
          IconLayerProvider
          IconProvider
          IdeBundle
          SelectInEditorManager
          StartupProgress
          TypePresentationService
          WelcomeWizardUtil
      ▶ injected.editor
      ▶ lang
      ▶ lexer
      ▶ navigation
      ▶ openapi
```

Модификаторы доступа

```
public class ModifiersDemo {

    public static void visibleEverywhere() {}

    protected static void inSubclasses() {}

    static void inPackage() {}

    private static void inClass() {}
}
```

Объявление класса

```java
package java.lang;

/**
 * The {@code Integer} class wraps a value of the primitive type
 * {@code int} in an object. An object of type {@code Integer}
 * contains a single field whose type is {@code int}.
 */
public final class Integer {

    // ...

}
```

Поля

```
package java.lang;

public final class Integer {

    private final int value;

    // ...

}
```

Конструкторы

```java
package java.lang;

public final class Integer {

    private final int value;

    public Integer(int value) {
        this.value = value;
    }

    // ...
}
```

Конструкторы

```java
package java.lang;

public final class Math {

    /**
     * Don't let anyone instantiate this class.
     */
    private Math() {}

    // ...
}
```

Конструкторы

```java
package java.math;

public class BigInteger {

    public BigInteger(String val) {
        this(val, 10);
    }

    public BigInteger(String val, int radix) {
        // ...
    }

}
```

```java
package java.io;

public class FileInputStream {

    protected void finalize() {
        // cleanup
    }

    public void close() {
        // cleanup
    }

    // ...
}
```

Методы

```
package java.lang;

public final class Integer {

    private final int value;



    public int intValue() {
        return value;
    }

    // ...
}
```

Методы

```java
package java.lang;

public final class String {

    public int indexOf(int ch) {
        return indexOf(ch, 0);
    }

    public int indexOf(int ch, int fromIndex) {
        // ...
    }

    // ...
}
```

Статические поля и методы

```
package java.lang;

public final class Integer {

    public static final int MIN_VALUE = 0x80000000;

    public static int rotateRight(int i, int distance) {
        return (i >>> distance) | (i << -distance);
    }

    // ...
}
```

```java
package java.math;

public class BigInteger {

    public static final BigInteger ONE = valueOf(1);

    public static BigInteger valueOf(long val) {
        // ...
    }

    // ...
}
```

Вложенные классы

```
package java.util;

public class ArrayList<E> {

    Object[] elementData;

    public Iterator<E> iterator() {
        return new Itr();
    }

    private class Itr implements Iterator<E> {
        int cursor;
        // ...
    }

    // ...
}
```

Вложенные классы

```
package java.util;

public class Collections {

    public static final List EMPTY_LIST = new EmptyList<>();

    public static final <T> List<T> emptyList() {
        return (List<T>) EMPTY_LIST;
    }

    private static class EmptyList<E> {
        // ...
    }
}
```

Перечисления

```java
public class BadExample {
    public static final int MONDAY = 1;
    public static final int TUESDAY = 2;
    public static final int WEDNESDAY = 3;
    public static final int THURSDAY = 4;
    public static final int FRIDAY = 5;
    public static final int SATURDAY = 6;
    public static final int SUNDAY = 7;
}
```

Перечисления

```java
package java.time;

public enum DayOfWeek {
    MONDAY,
    TUESDAY,
    WEDNESDAY,
    THURSDAY,
    FRIDAY,
    SATURDAY,
    SUNDAY




}
```

Перечисления

```java
for (DayOfWeek day : DayOfWeek.values()) {

    System.out.println(
        day.ordinal() + " " + day.name());

}
```

Аннотации

```java
package java.lang;

public final class Character {

    @Deprecated
    public static boolean isJavaLetter(char c) {
        // ...
    }

    @SuppressWarnings("unchecked")
    public static final Class<Character> TYPE =
        (Class<Character>) Class.getPrimitiveClass("char");

}
```

Аннотации

```
package java.lang;

import java.lang.annotation.*;
import static java.lang.annotation.ElementType.*;

@Target({TYPE, FIELD, METHOD, PARAMETER,
        CONSTRUCTOR, LOCAL_VARIABLE})
@Retention(RetentionPolicy.SOURCE)
public @interface SuppressWarnings {

    String[] value();

}
```

Наследование

```java
package java.lang;

public final class BigDecimal extends Number {

    public int intValue() {
        // ...
    }

    // no shortValue() method,
    // it's inherited from Number
}
```

Наследование

```java
package java.lang;

public final class StringBuilder
    extends AbstractStringBuilder {

    @Override
    public StringBuilder append(String str) {
        // ...
    }

    // base method in AbstractStringBuilder:
    // AbstractStringBuilder append(String str)
}
```

```java
package java.lang;

public final class StringBuilder
    extends AbstractStringBuilder {

    public StringBuilder() {
        super(16);
    }

    @Override
    public StringBuilder append(String str) {
        super.append(str);
        return this;
    }

    // ...
}
```

java.lang.Object

```
package java.lang;

public final class String /*extends Object*/ {

    // ...

}
```

```java
package java.lang;

public class Object {

    public String toString() {
        return getClass().getName() + "@"
            + Integer.toHexString(hashCode());
    }

    public boolean equals(Object obj) {
        return this == obj;
    }

    public native int hashCode();

    // ...
}
```

```java
package java.lang;

public final class String {

    @Override
    public boolean equals(Object anObject) {
        if (this == anObject) {
            return true;
        }
        if (anObject instanceof String) {
            String other = (String)anObject;
            // ...
        }
        return false;
    }
    // ...
}
```

- Liskov Substitution Principle (LSP)
- Barbara Liskov

Интерфейсы

```
public abstract class OrderService {

    public abstract Order[] getOrdersByClient(
            long clientId);
}
```

Интерфейсы

```
public interface OrderService {

    Order[] getOrdersByClient(long clientId);


}
```

Интерфейсы

```
import java.time.LocalDate;

public interface OrderService {

    Order[] getOrdersByClient(long clientId);

    default Order[] getOrdersByClient(
            long clientId, LocalDate date) {
        Order[] allOrders = getOrdersByClient(clientId);
        Order[] orders = // ... filter by date
        return orders;
    }
}
```

Интерфейсы

```java
public class OrderServiceImpl
        extends ServiceBase
        implements OrderService {

    public Order[] getOrdersByClient(
            long clientId) {
        // ...
    }
}
```

Интерфейсы

```
package java.lang;

public interface CharSequence {

    int length();

    char charAt(int index);

    CharSequence subSequence(
            int start, int end);
}
```

Интерфейсы

```java
package java.lang;

public interface Appendable {

    Appendable append(CharSequence csq);

    Appendable append(CharSequence csq,
            int start, int end);

    Appendable append(char c);
}
```

Интерфейсы

```java
package java.lang;

@FunctionalInterface
public interface Runnable {

    void run();
}
```

Интерфейсы

```
package java.lang;

@FunctionalInterface
public interface Comparator<T> {

    int compare(T o1, T o2);
}
```