

Язык Java и разработка Java-приложений

Владимир Юрьевич Романов,
Московский Государственный Университет им. М.В.Ломоносова
Факультет Вычислительной Математики и Кибернетики
vromanov@cs.msu.su,
romanov.rvy@yandex.ru

Литература

МГУ им. М.В.Ломоносова. Факультет ВМК.

Романов Владимир Юрьевич ©2025

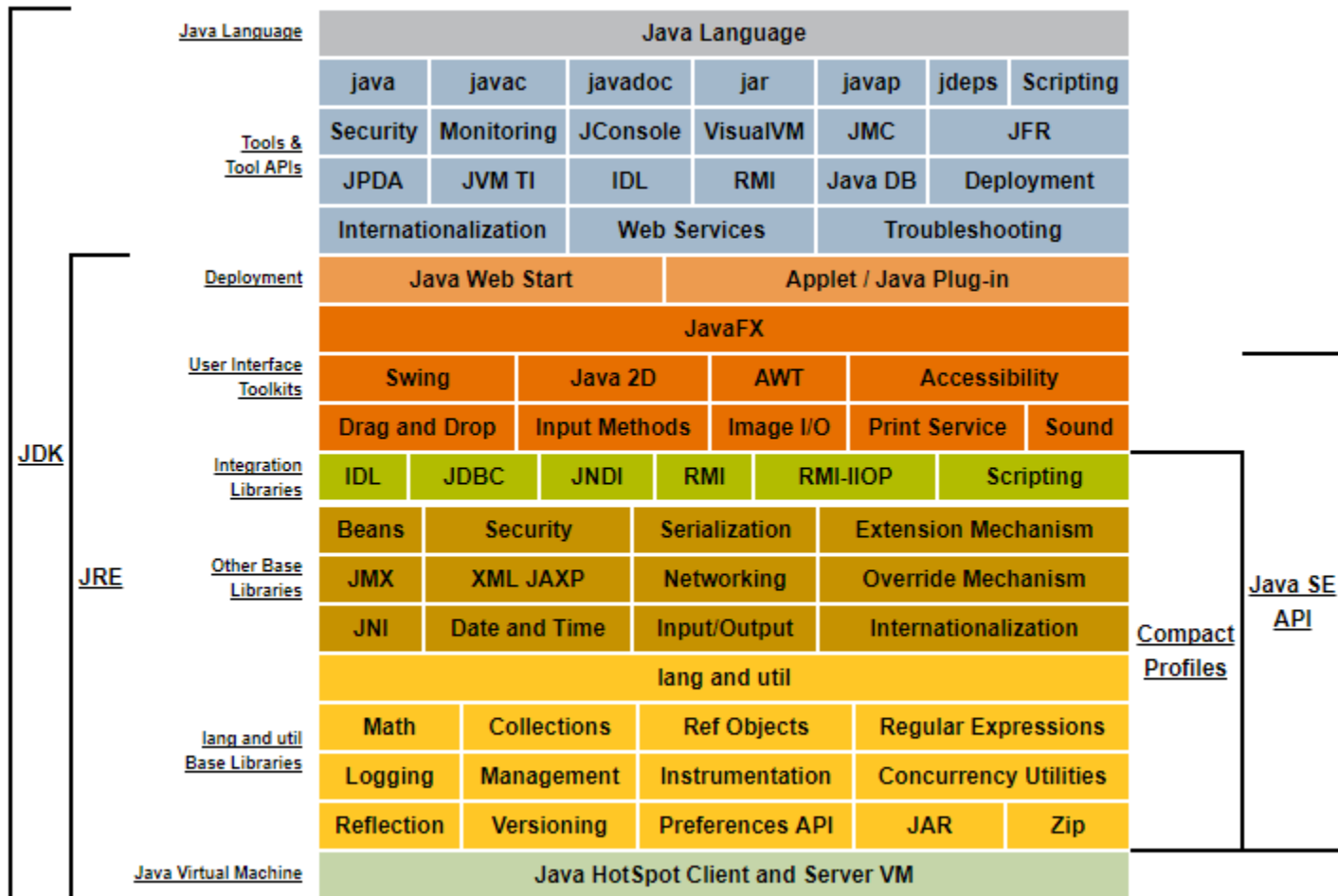
1. James Gosling, Bill Joy, Guy Steele, Gilad Bracha, Alex Buckley
Java(TM) Programming Language, Java SE 8 Edition
<https://docs.oracle.com/javase/specs/jvms/se8/jvms8.pdf>
2. Tim Lindholm, Frank Yellin, Gilad Bracha, Alex Buckley
The Java™ Virtual Machine Specification, Java SE 8 Edition
<https://docs.oracle.com/javase/specs/jvms/se8/jvms8.pdf>
3. Eclipse web sites: **<http://www.eclipse.org>**
4. Oracle web site:
<https://docs.oracle.com/javase/8/docs/>
5. Брюс Эккель. Философия Java. 4-е издание.
Издательство «Питер». Петербург 2016.
ISBN 5-88782-105-1

Документация по Java

МГУ им. М.В.Ломоносова. Факультет ВМК.

Романов Владимир Юрьевич ©2025

<https://docs.oracle.com/javase/8/docs/>



Цели курса

МГУ им. М.В.Ломоносова. Факультет ВМК.

Романов Владимир Юрьевич ©2025

- Изучение языка Java перед изучением технологий основанных на языке Java
- Изучение среды разработки Eclipse распространяемой фирмой IBM в исходных текстах
- Подготовка к изучению курса:
“Разработка объектно-ориентированных систем программирования в среде Eclipse”

О курсе

“Разработка объектно-ориентированных систем программирования в среде Eclipse”

МГУ им. М.В.Ломоносова. Факультет ВМК.

Романов Владимир Юрьевич ©2025

- Разработка расширений (plug-in) для среды Eclipse
- Разработка расширений среды для объектно-ориентированных систем программирования
- Разработка распознавателей для объектно-ориентированных языков программирования
- Разработка промежуточного представления компилятора с помощью Eclipse Modeling Framework

О курсе

“Разработка объектно-ориентированных систем программирования в среде Eclipse”

МГУ им. М.В.Ломоносова. Факультет ВМК.

Романов Владимир Юрьевич ©2025

- Визуализация промежуточного представления компилятора с помощью *Graphic Editing Framework*
- Моделирование программ с помощью реализованной в среде Eclipse *метамодел* языка *UML 2.0*
- Генерация кода для виртуальной машины Java
- Разработка в среде Eclipse отладчиков для языков программирования

Язык программирования Java

МГУ им. М.В.Ломоносова. Факультет ВМК.

Романов Владимир Юрьевич ©2025

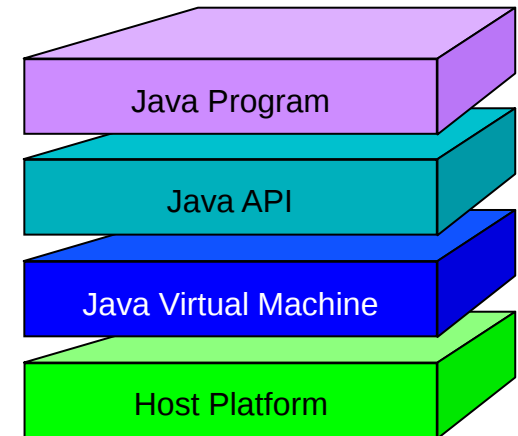
- Простота
- Язык высокого уровня
- Объектно-ориентированный
- Независим от архитектуры
- Переносим в исходных текстах и в двоичном коде
- Интерпретируемый и компилируемый
- Многопоточный

Платформа Java

МГУ им. М.В.Ломоносова. Факультет ВМК.

Романов Владимир Юрьевич ©2025

- Платформа – окружение разработки где работает программа
- Java – платформа работает на разных операционных системах
- Java – платформа состоит из виртуальной машины Java и интерфейса приложений



Компилятор и виртуальная машина Java

МГУ им. М.В.Ломоносова. Факультет ВМК.

Романов Владимир Юрьевич ©2025



Установка Eclipse. Версии среды

МГУ им. М.В.Ломоносова. Факультет ВМК.

Романов Владимир Юрьевич ©2025

- Среда Eclipse бесплатна
- Дистрибутив Eclipse загружается с сайта:
<https://www.eclipse.org/downloads/packages/release/2020-06/r>

Eclipse Modeling Tools

444 MB

9,497 DOWNLOADS



The Modeling package provides tools and runtimes for building model-based applications. You can use it to graphically design domain models, to leverage those models at design time by creating and editing dynamic instances, to collaborate via Eclipse's team support with facilities for comparing and merging models and model instances structurally, and finally to generate Java code from those models to produce complete applications. In addition, via the package's discover catalog, you can easily install a wide range of additional powerful, model-based tools and runtimes to suit your specific needs.



Windows x86_64
macOS x86_64
Linux x86_64

Установка Eclipse. Eclipse Modeling Tools

МГУ им



Eclipse IDE for JavaScript and Web Developers

156 MB - Downloaded 29,419 Times

Windows **32-bit 64-bit**
Mac Cocoa **64-bit**
Linux **32-bit 64-bit**



Eclipse IDE for Java and DSL Developers

309 MB - Downloaded 19,275 Times

Windows **32-bit 64-bit**
Mac Cocoa **64-bit**
Linux **32-bit 64-bit**



Eclipse for Parallel Application Developers

227 MB - Downloaded 18,259 Times

Windows **32-bit 64-bit**
Mac Cocoa **64-bit**
Linux **32-bit 64-bit**



Eclipse for RCP and RAP Developers

254 MB - Downloaded 13,260 Times

Windows **32-bit 64-bit**
Mac Cocoa **64-bit**
Linux **32-bit 64-bit**



Eclipse Modeling Tools

392 MB - Downloaded 8,603 Times

Windows **32-bit 64-bit**
Mac Cocoa **64-bit**
Linux **32-bit 64-bit**



Eclipse IDE for Java and Report Developers

315 MB - Downloaded 4,941 Times

Windows **32-bit 64-bit**
Mac Cocoa **64-bit**
Linux **32-bit 64-bit**



Eclipse for Testers

131 MB - Downloaded 4,115 Times

Windows **32-bit 64-bit**
Mac Cocoa **64-bit**
Linux **32-bit 64-bit**

Eclipse Modeling Tools

МГУ им. М.В.Ломоносова. Факультет ВМК.

Романов Владимир Юрьевич ©2025

<https://www.eclipse.org/downloads/packages/release/2020-06/r>

Eclipse Modeling Tools

444 MB

9,578 DOWNLOADS



The Modeling package provides tools and runtimes for building model-based applications. You can use it to graphically design domain models, to leverage those models at design time by creating and editing dynamic instances, to collaborate via Eclipse's team support with facilities for comparing and merging models and model instances structurally, and finally to generate Java code from those models to produce complete applications. In addition, via the package's discover catalog, you can easily install a wide range of additional powerful, model-based tools and runtimes to suit your specific needs.



Windows x86_64

macOS x86_64

Linux x86_64



IntelliJ IDEA Community Edition

МГУ им. М.В.Ломоносова. Факультет ВМК.

Романов Владимир Юрьевич ©2025

<https://www.jetbrains.com/idea/download/#section=windows>

Download IntelliJ IDEA

Windows

macOS

Linux

Ultimate

For web and enterprise development

Download

.exe ▼

Free 30-day trial

Community

For JVM and Android development

Download

.exe ▼

Free, built on open source

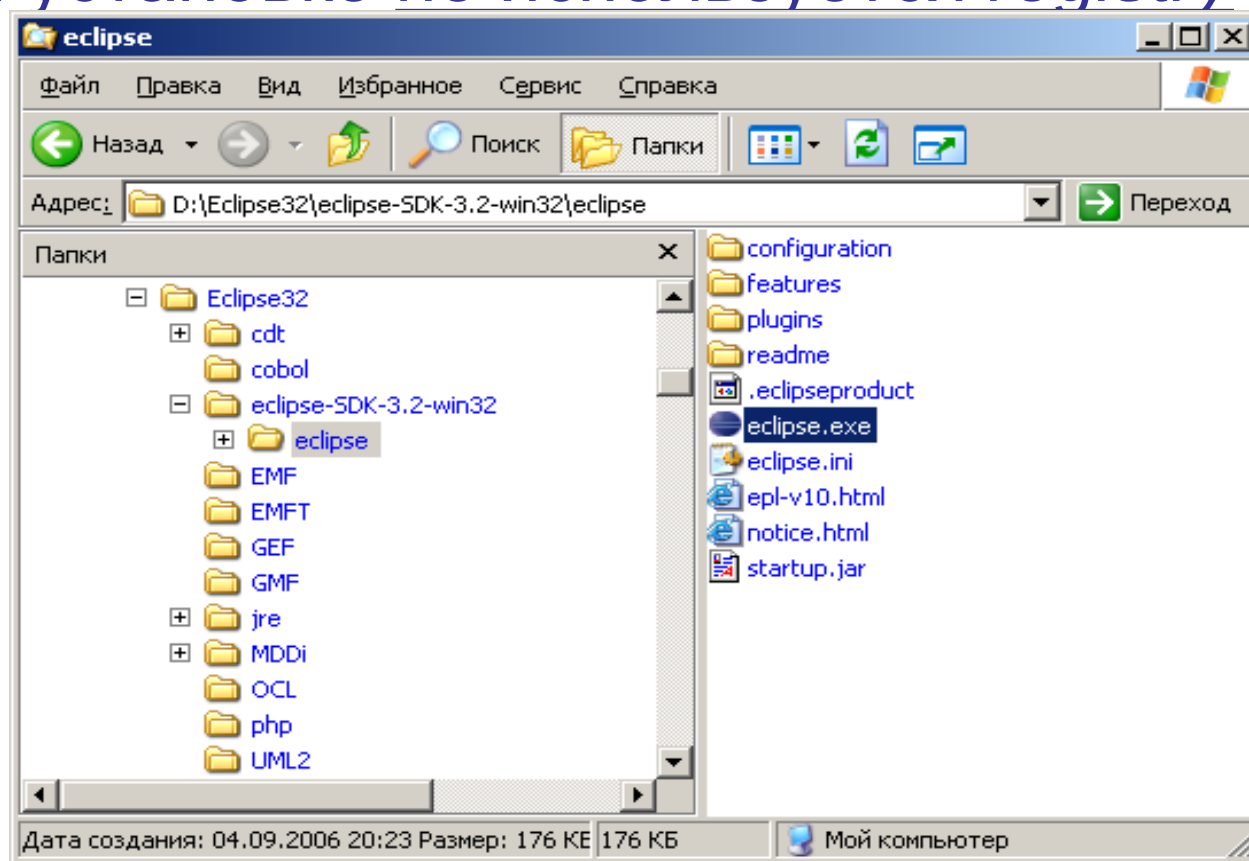
Краткий обзор среды Eclipse для разработки на языке Java

Запуск среды Eclipse на исполнение

МГУ им. М.В.Ломоносова. Факультет ВМК.

Романов Владимир Юрьевич ©2025

- Установка – просто распаковка Zip-архива.
При установке не используется *registry*



Обзор среды Eclipse

МГУ им. М.В.Ломоносова. Факультет ВМК.

Романов Владимир Юрьевич ©2025

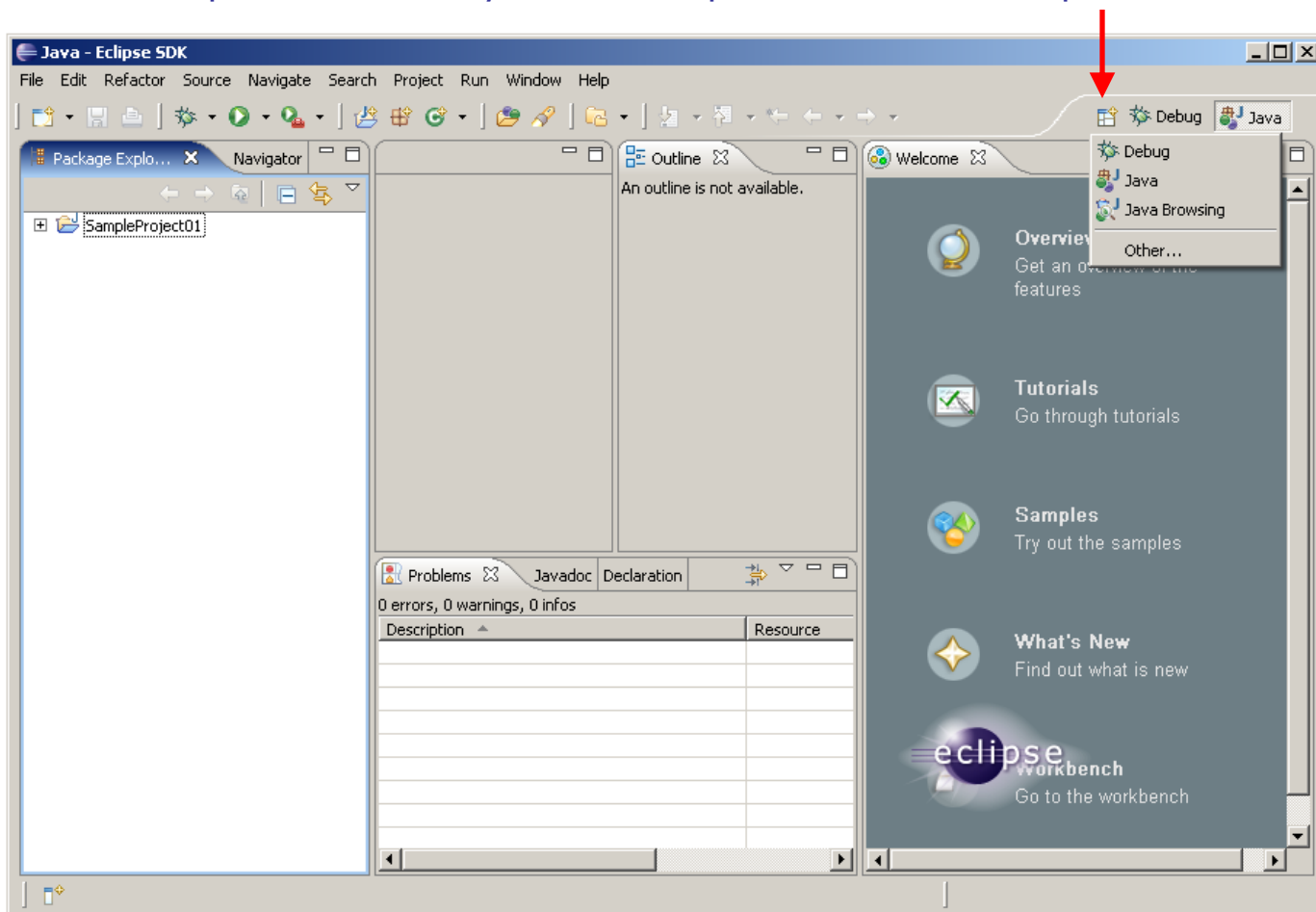
- Перспективы (Perspectives)
- Виды (Views)
- Редакторы (Editors)
- Работа с файлами
- Навигация по рабочему месту (Workbench)
- Консоль (Console)

Обзор среды Eclipse. Перспективы (Java)

МГУ им. М.В.Ломоносова. Факультет ВМК.

Романов Владимир Юрьевич ©2025

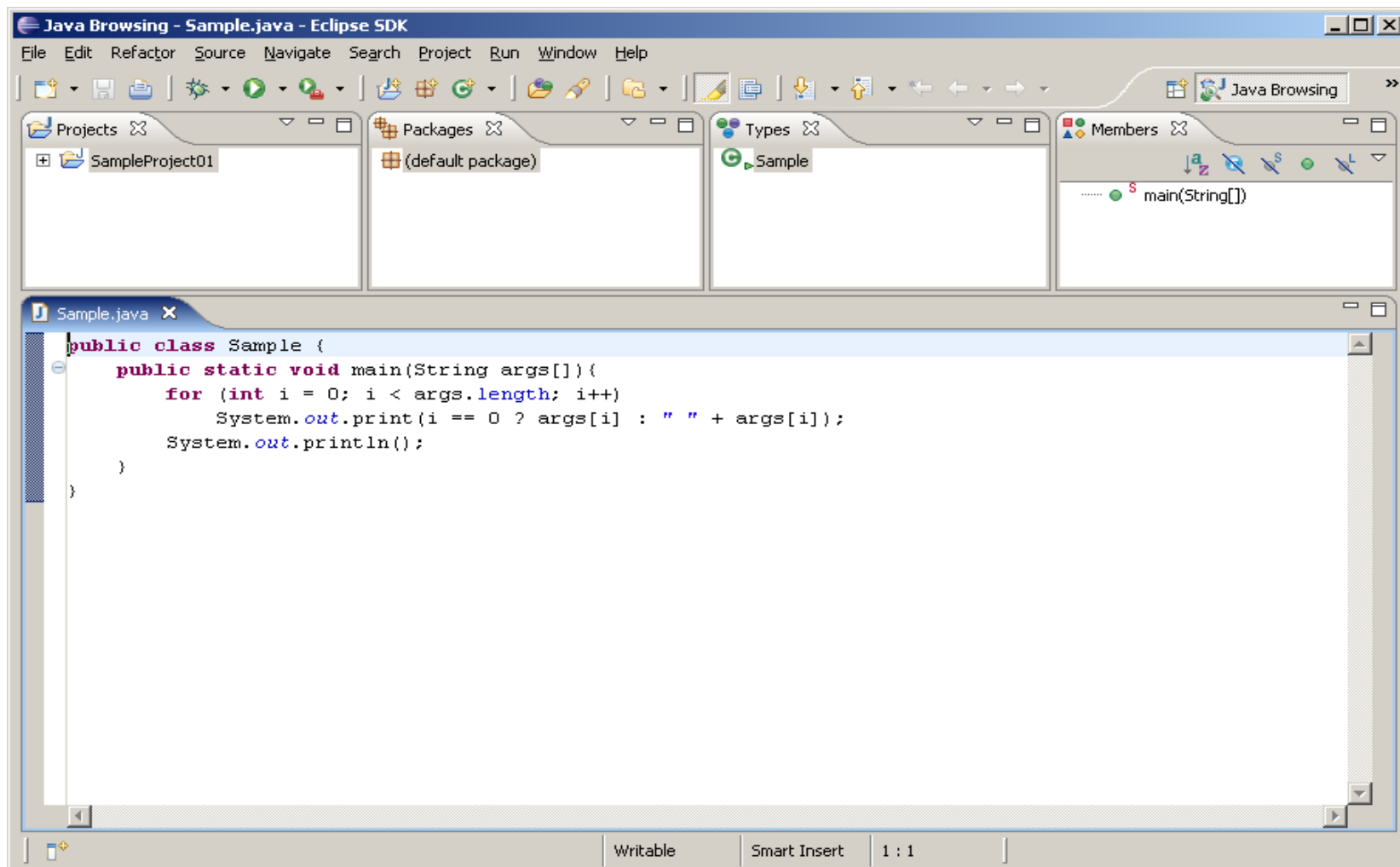
- На рабочем месте (workbench) может быть открыто несколько перспектив
- Различные перспективы могут быть выбраны из панели перспектив



Обзор среды Eclipse. Перспективы (Java Browsing)

МГУ им. М.В.Ломоносова. Факультет ВМК.

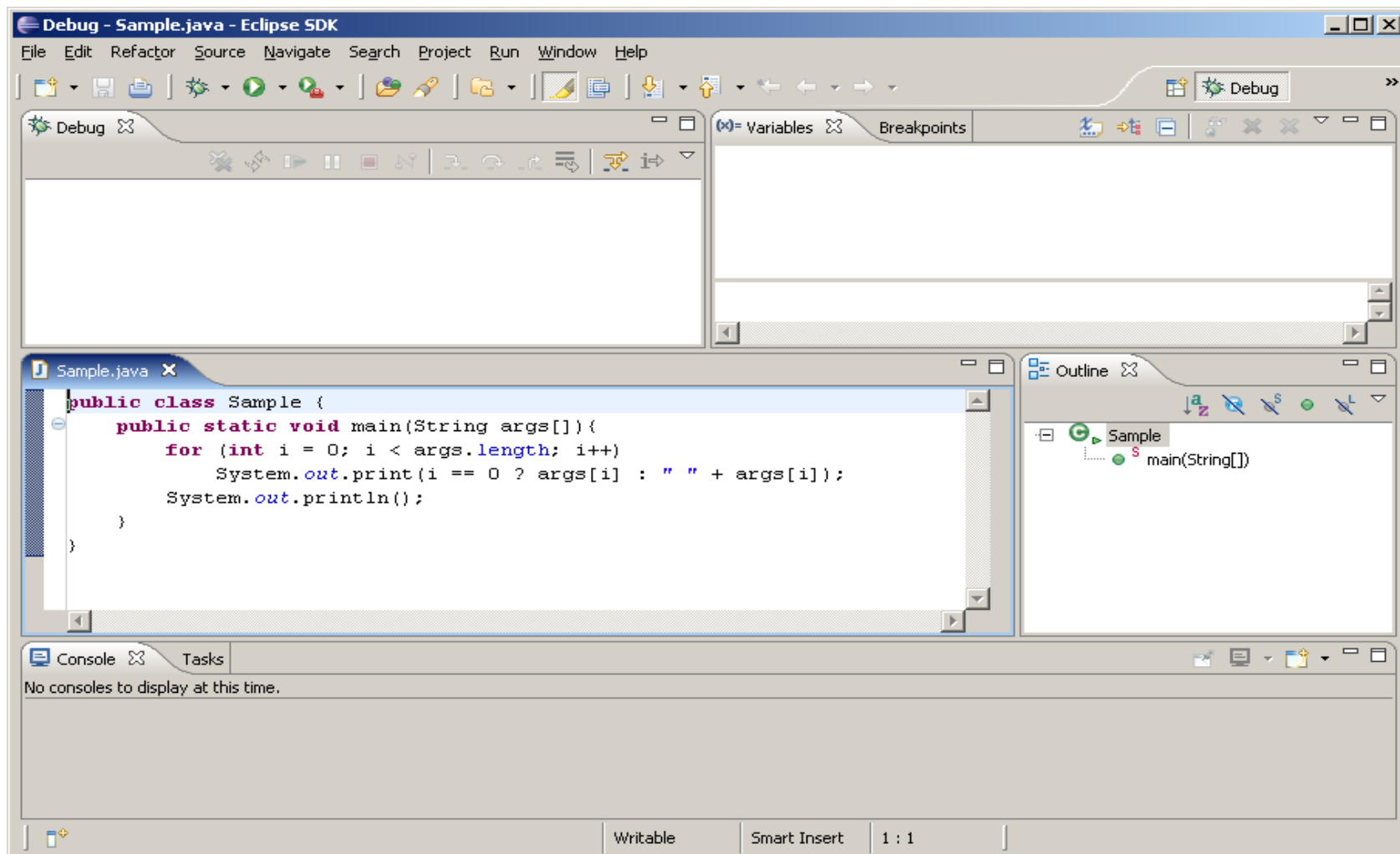
Романов Владимир Юрьевич ©2025



Обзор среды Eclipse. Перспективы (Debug)

МГУ им. М.В.Ломоносова. Факультет ВМК.

Романов Владимир Юрьевич ©2025



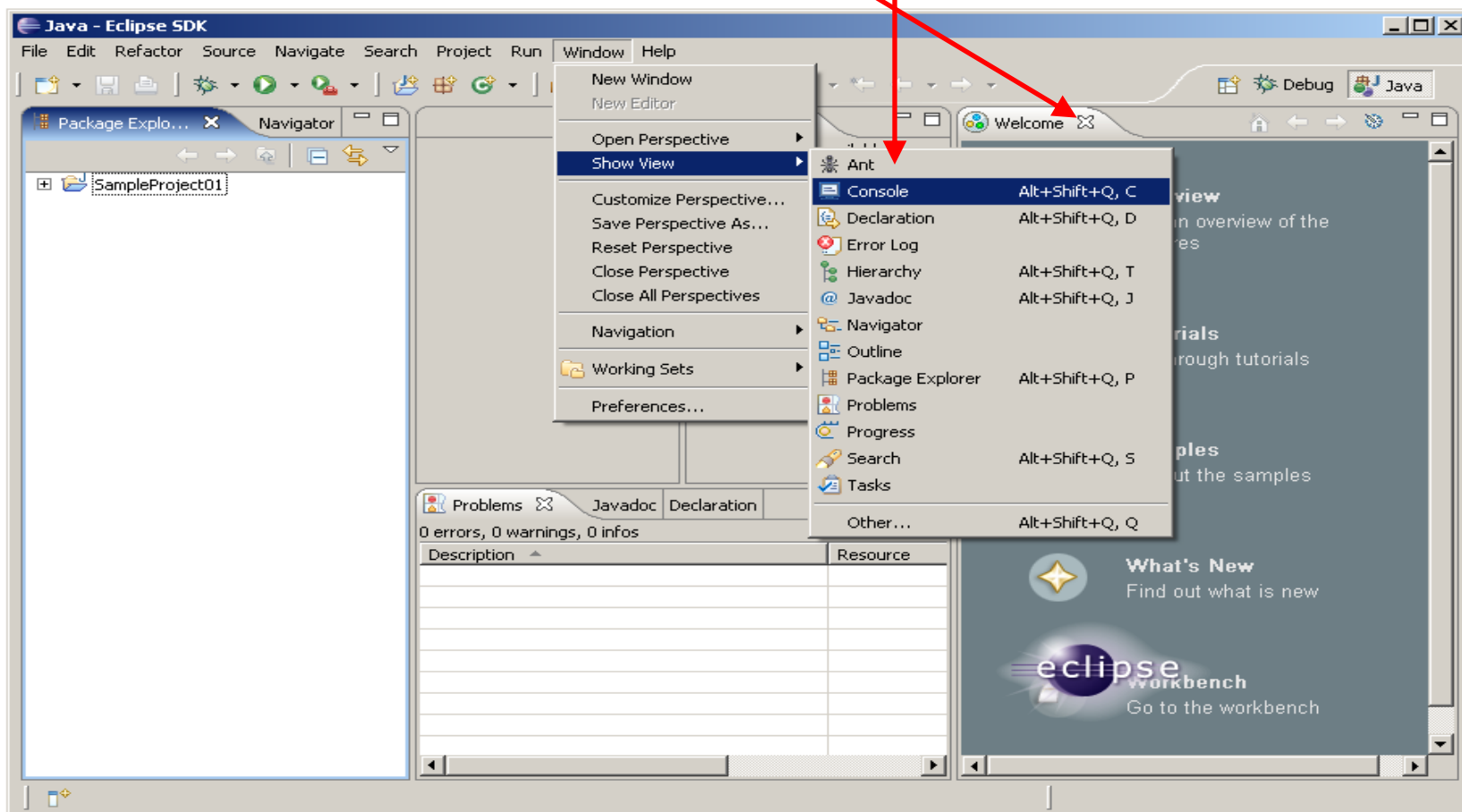
Обзор среды Eclipse.

Добавление видов к перспективам

МГУ им. М.В.Ломоносова. Факультет ВМК.

Романов Владимир Юрьевич ©2025

- Добавление к перспективе нового вида (Console)
- Удаление вида Welcome из перспективы



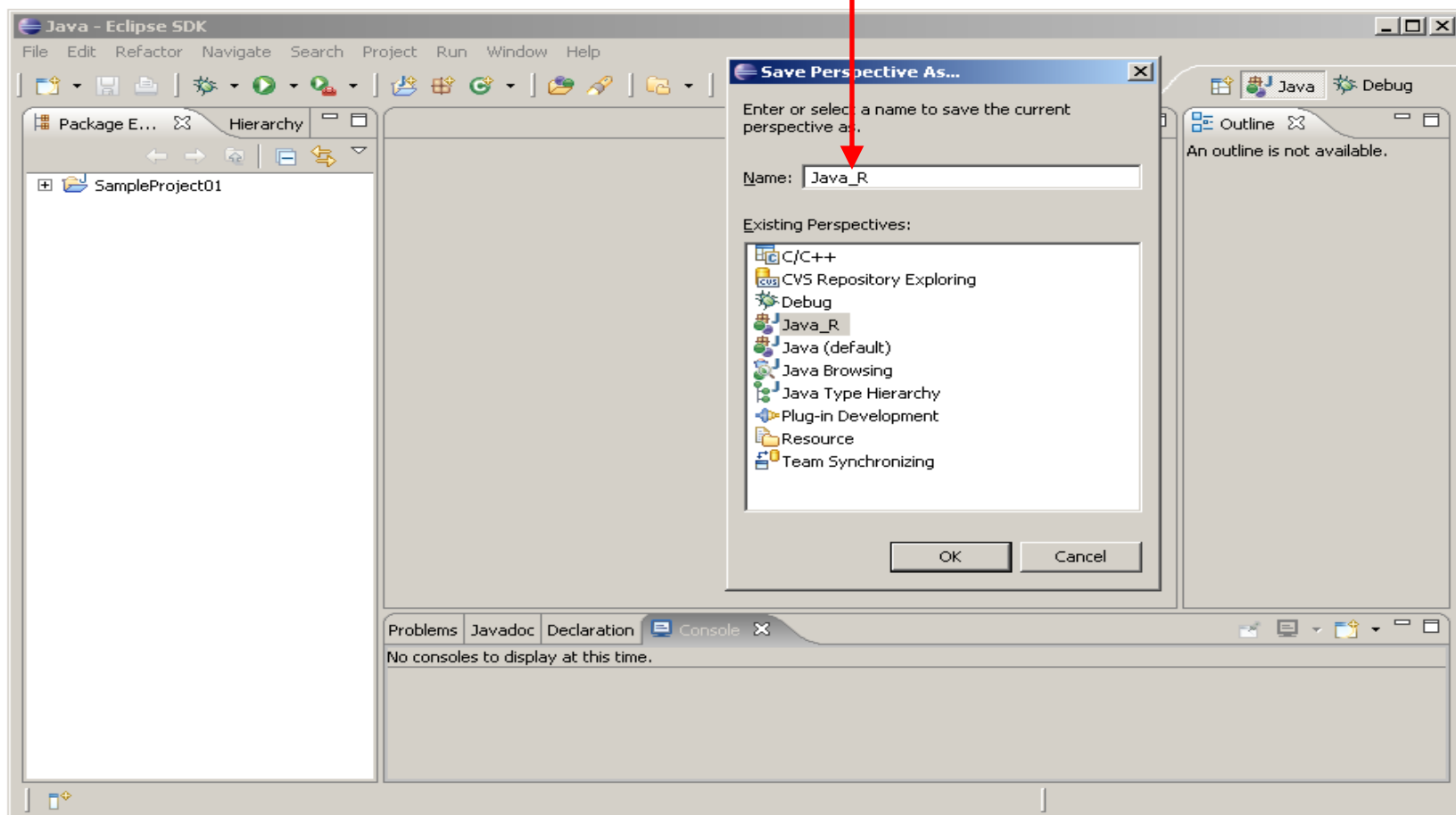
Обзор среды Eclipse.

Сохранение перспективы

МГУ им. М.В.Ломоносова. Факультет ВМК.

Романов Владимир Юрьевич ©2025

- Сохранение перспективы под новым именем



Обзор среды Eclipse.

Виды

МГУ им. М.В.Ломоносова. Факультет ВМК.

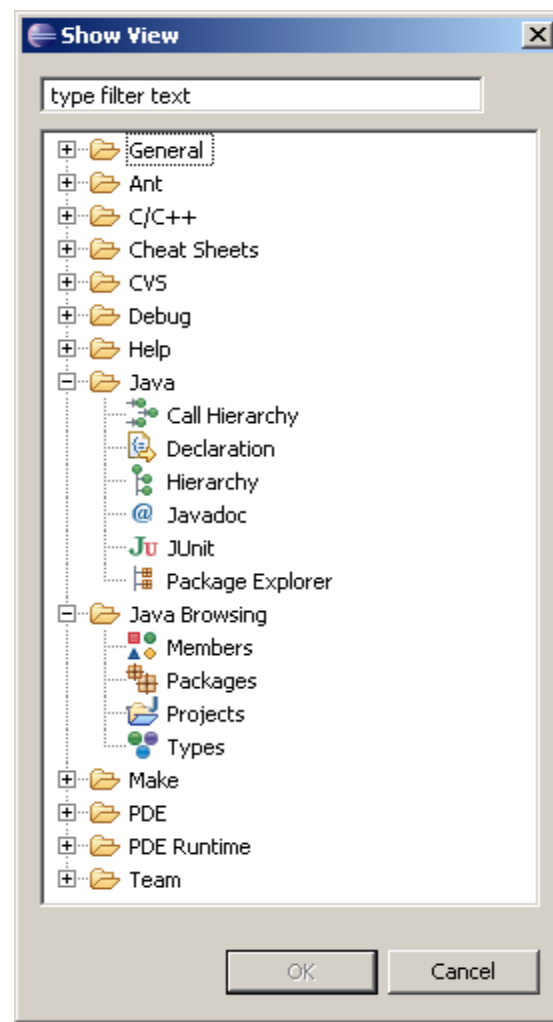
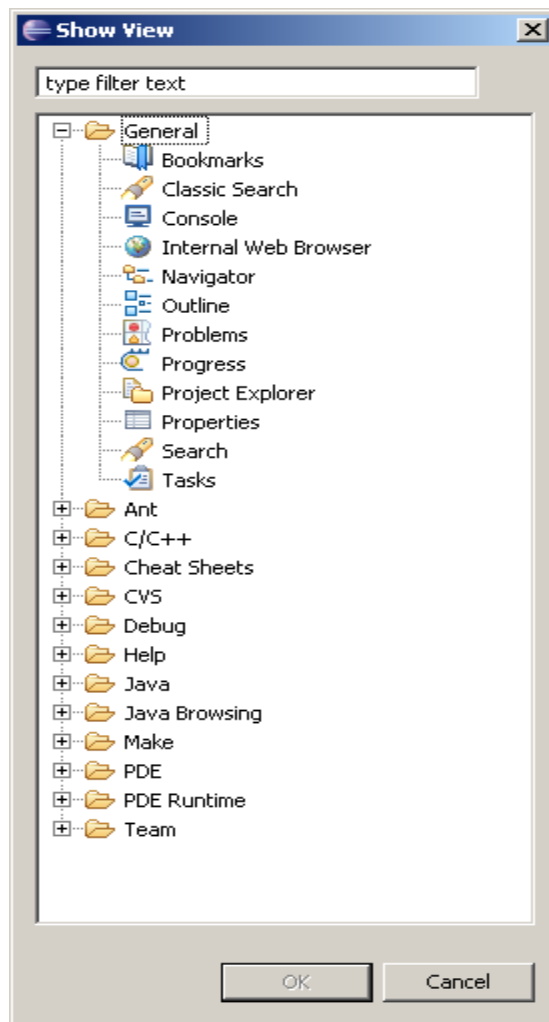
Романов Владимир Юрьевич ©2025

Java Perspective

- Declaration
- Javadoc
- Outline
- Package Explorer
- Problems
- Welcome

Debug Perspective

- Console
- Debug
- Outline
- Tasks
- Variables
- Welcome

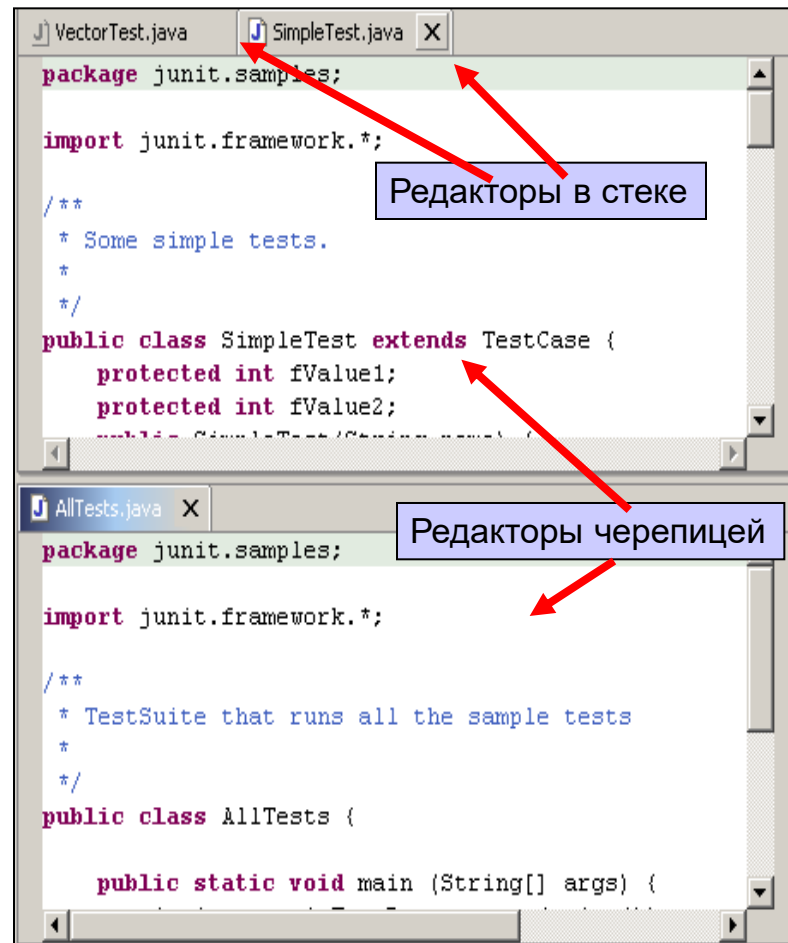


Обзор среды Eclipse. Редакторы

МГУ им. М.В.Ломоносова. Факультет ВМК.

Романов Владимир Юрьевич ©2025

- Редакторы могут быть открыты для большинства ресурсов
- Несколько редакторов может быть открыто в стеке
- Если содержимое редактора было модифицировано, но не сохранено, то перед именем файла *
- Если редактор активен, то в меню и панели инструментов содержатся операции применимые к активному редактору



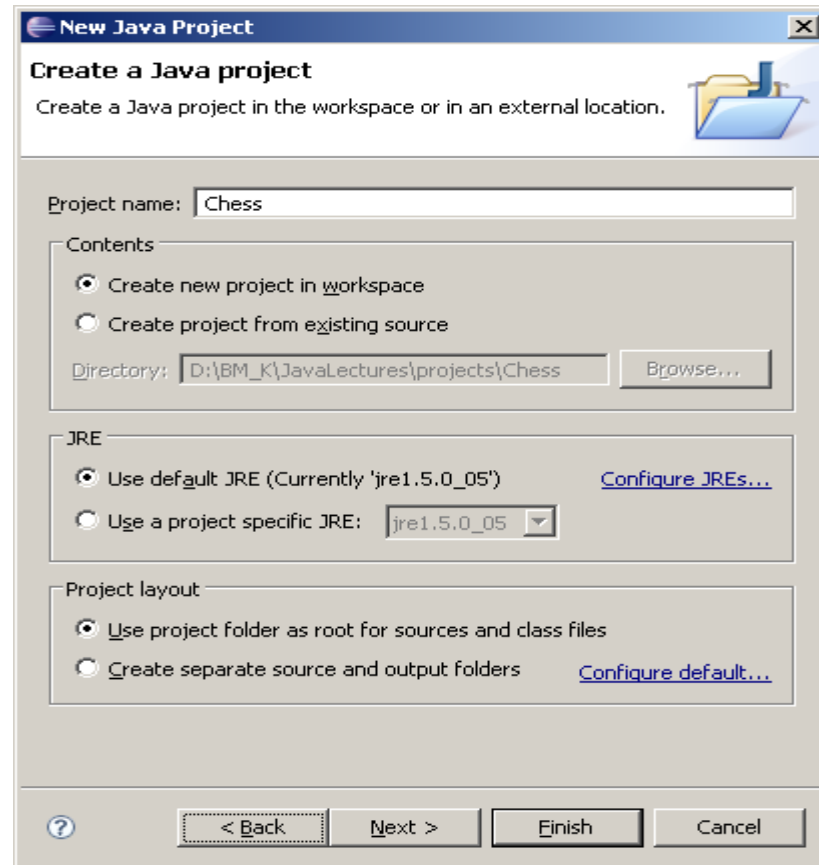
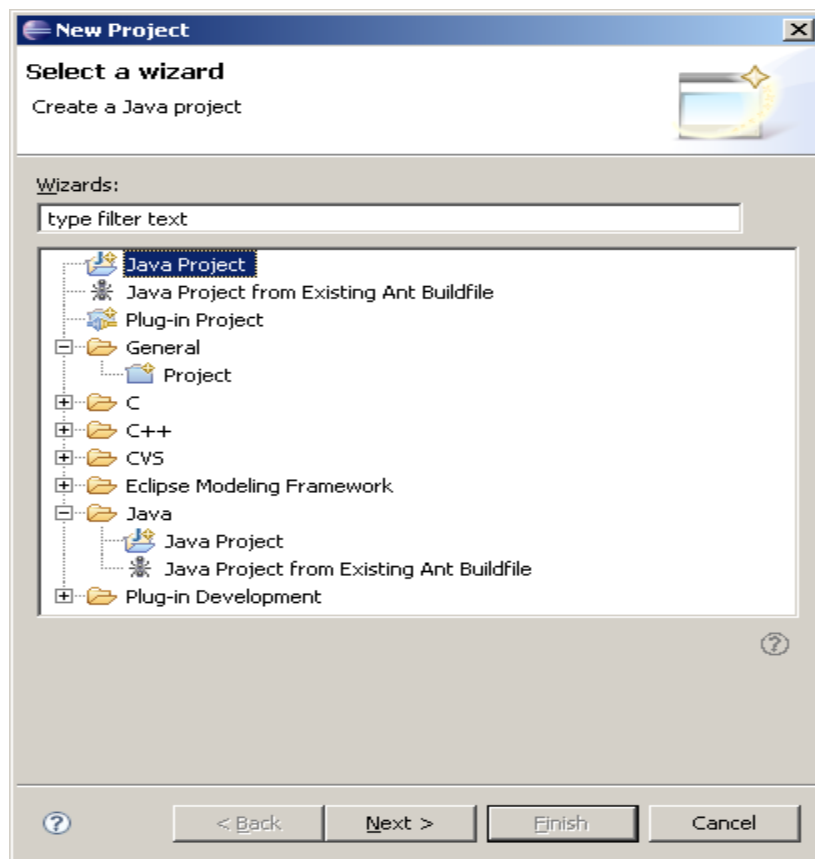
Обзор среды Eclipse.

Создание нового проекта

МГУ им. М.В.Ломоносова. Факультет ВМК.

Романов Владимир Юрьевич ©2025

- File | New | Project

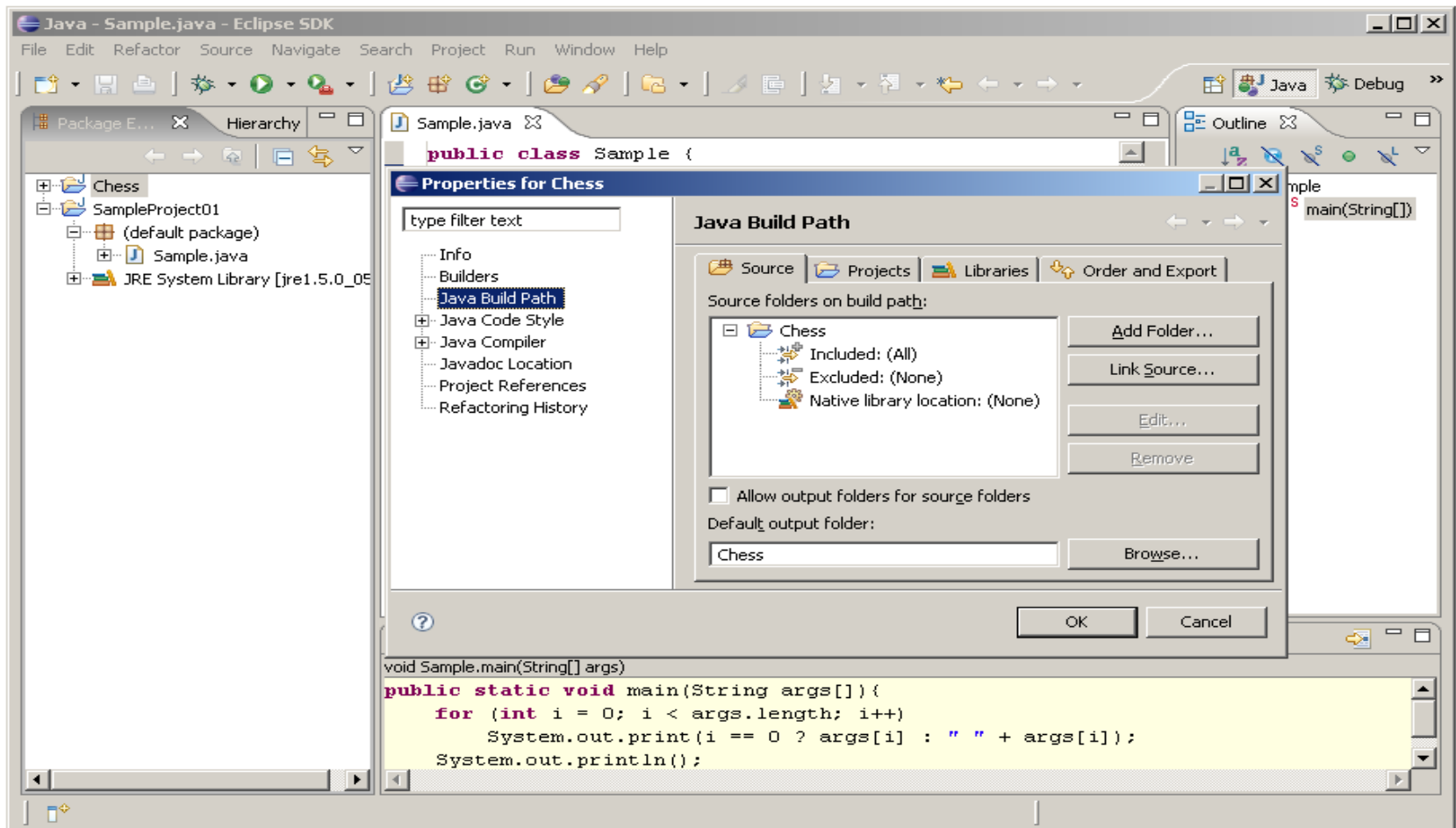


Обзор среды Eclipse. Свойства

МГУ им. М.В.Ломоносова. Факультет ВМК.

Романов Владимир Юрьевич ©2025

Все ресурсы имеют свойства.

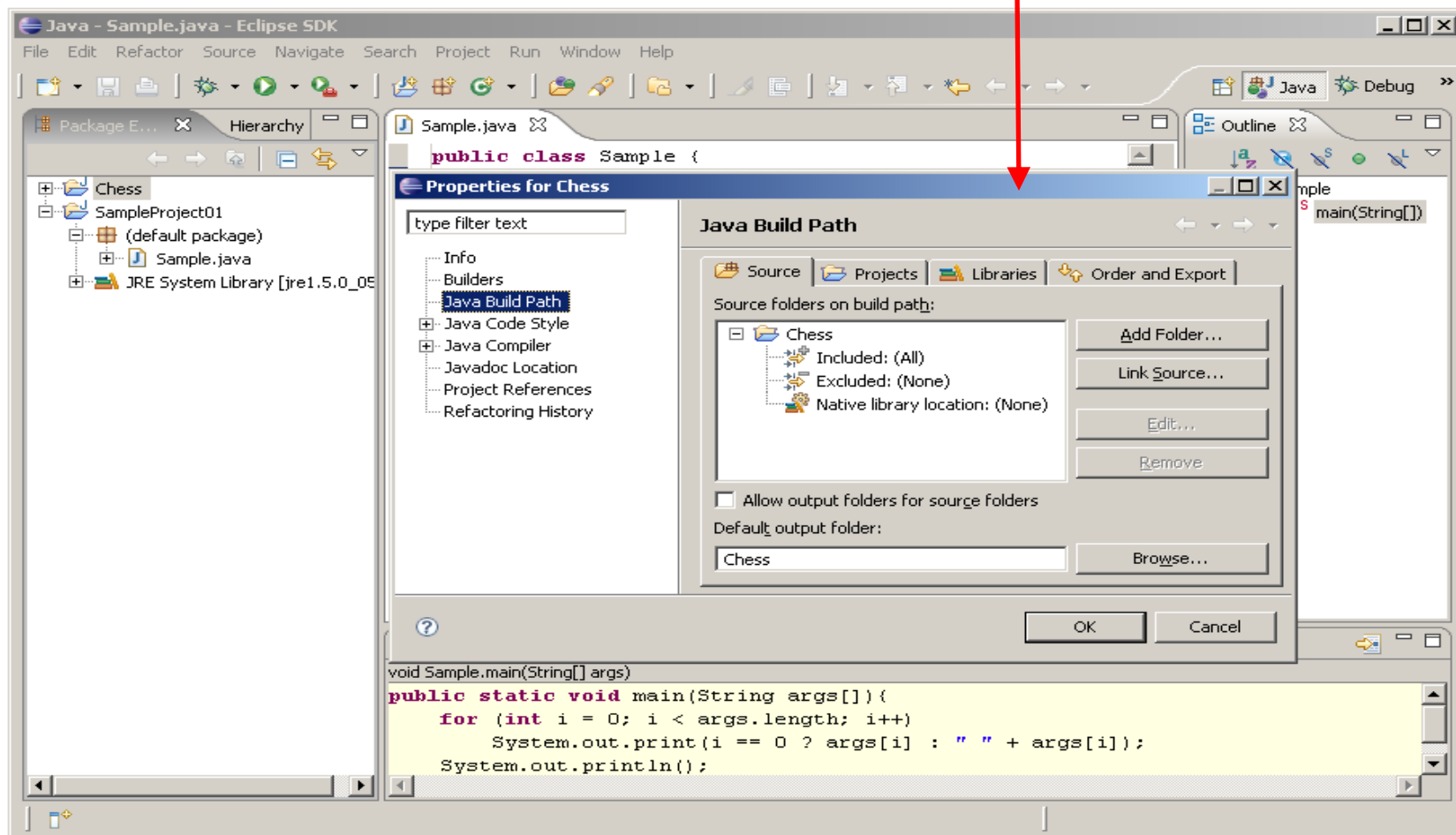


Обзор среды Eclipse. Свойства

МГУ им. М.В.Ломоносова. Факультет ВМК.

Романов Владимир Юрьевич ©2025

Все ресурсы имеют свойства. Свойства проекта *Chess*:



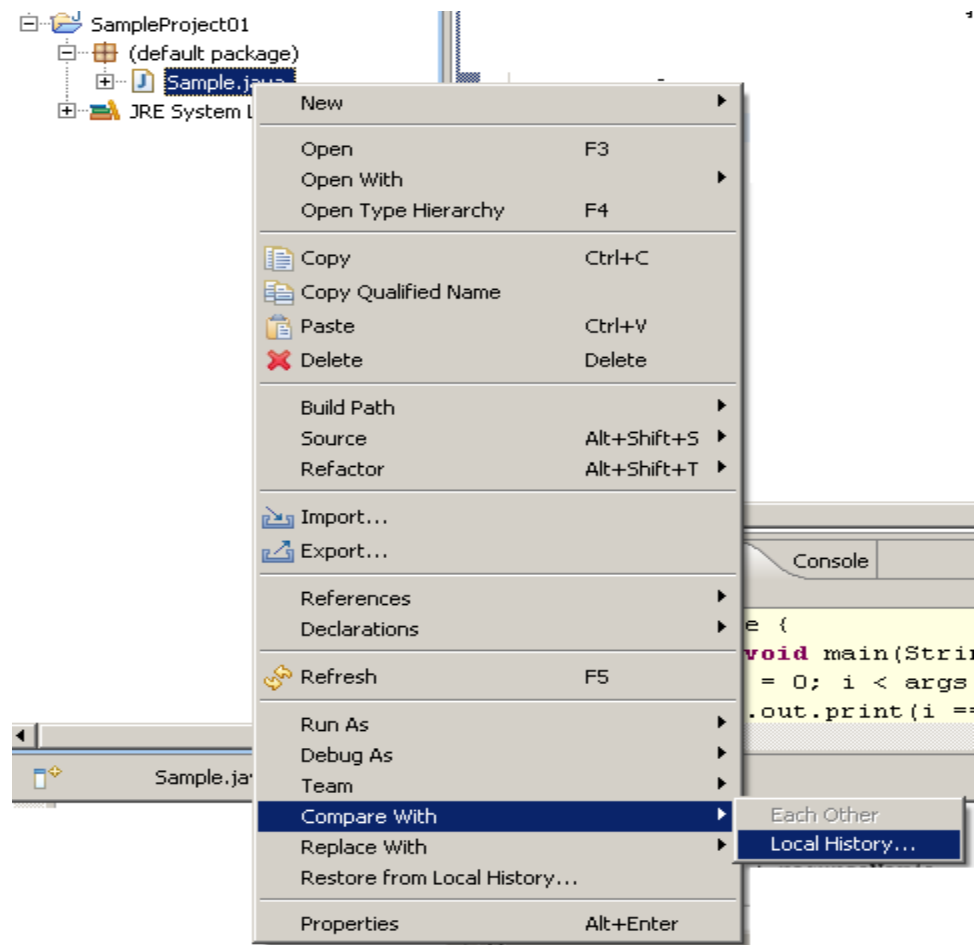
Обзор среды Eclipse.

Локальная история (1)

МГУ им. М.В.Ломоносова. Факультет ВМК.

Романов Владимир Юрьевич ©2025

Для каждого файла хранится *локальная история* файла



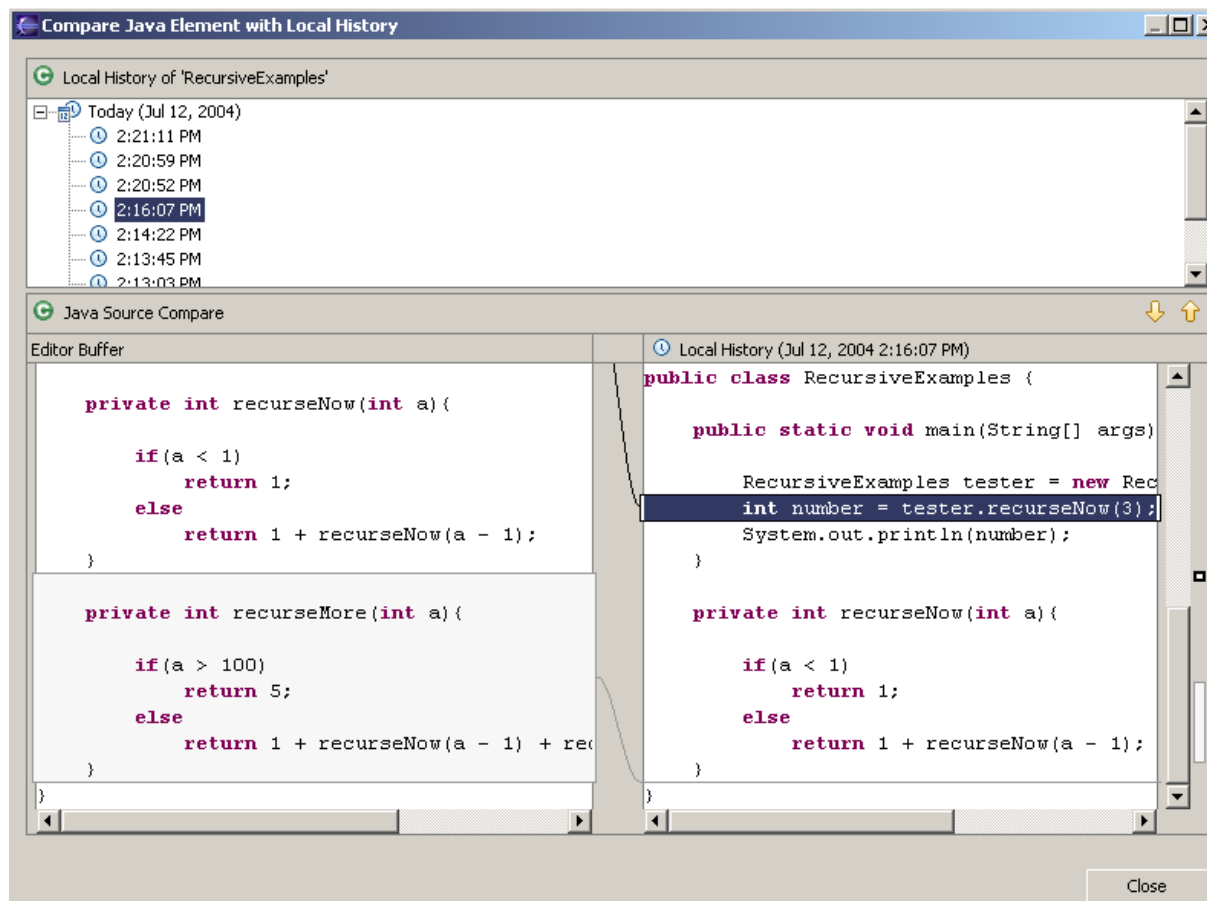
Обзор среды Eclipse.

Локальная история (2)

МГУ им. М.В.Ломоносова. Факультет ВМК.

Романов Владимир Юрьевич ©2025

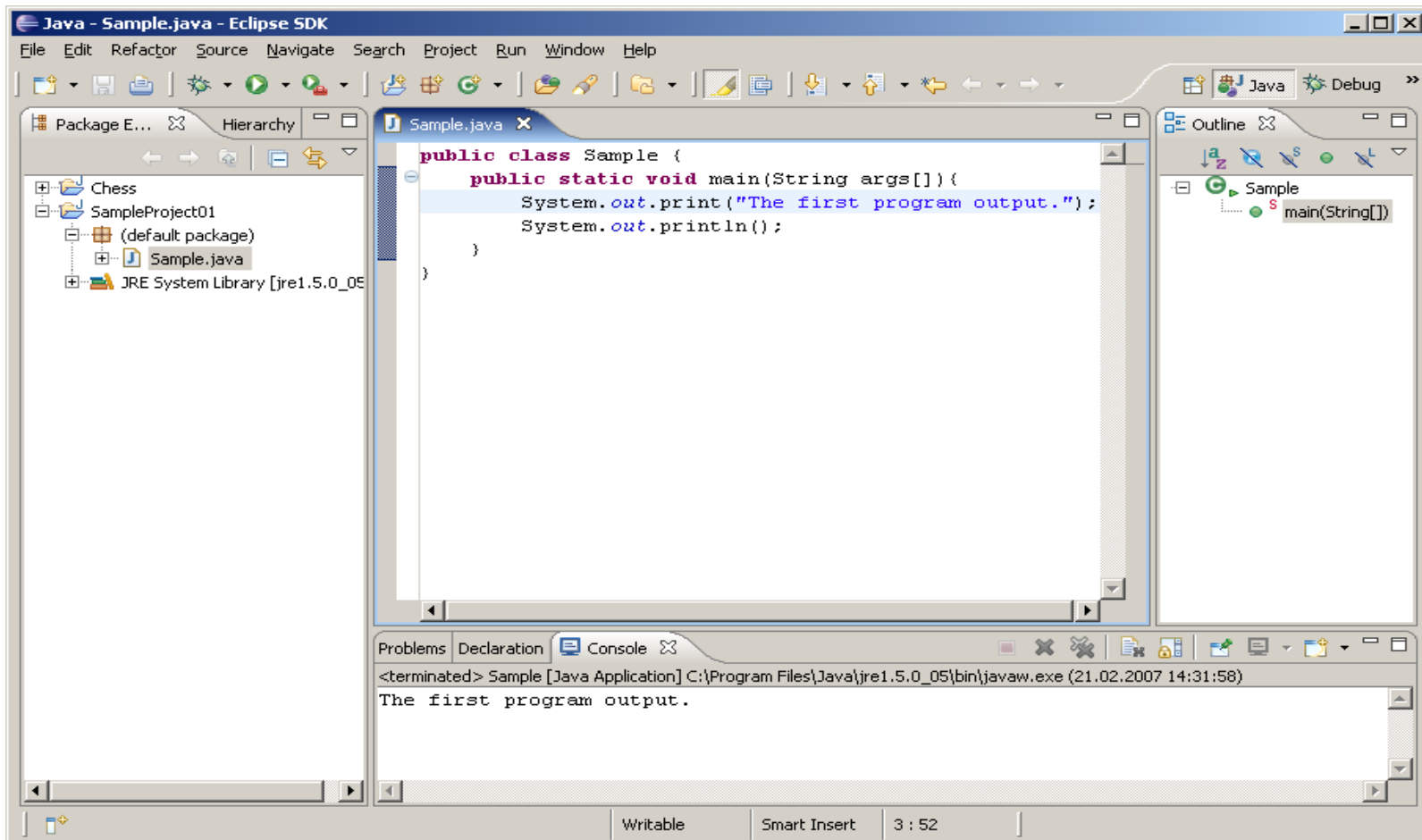
- Каждое использование команды **Save** сохраняет редакцию файла
- Редакции файлов можно сравнить



Вывод программы на консоль

МГУ им. М.В.Ломоносова. Факультет ВМК.

Романов Владимир Юрьевич ©2025



Введение в язык Java

Классы Java и их синтаксис

Члены класса

МГУ им. М.В.Ломоносова. Факультет ВМК.

Романов Владимир Юрьевич ©2025

- Классы инкапсулируют атрибуты (поля) и поведение (методы)
- Поля и методы являются *членами* класса
- Члены класса могут принадлежать всем экземплярам класса. В этом случае поля и методы помечаются ключевым словом *static*
- Члены класса могут принадлежать конкретным экземплярам класса. В этом случае они называются полями и методами экземпляров класса
- В одном файле с расширением **.java* не может быть более одного публичного класса

Классы Java и их синтаксис.

Члены классов

МГУ им. М.В.Ломоносова. Факультет ВМК.

Романов Владимир Юрьевич ©2025

- Классы инкапсулируют атрибуты (поля) и поведение (методы)
- Атрибуты и методы являются *членами* класса

```
class Square {  
    int h;  
    int v;  
    boolean isNear(Square s) {  
        return  
            Math.abs(h - s.h) <= 1 &&  
            Math.abs(v - s.v) <= 1;  
    }  
}
```

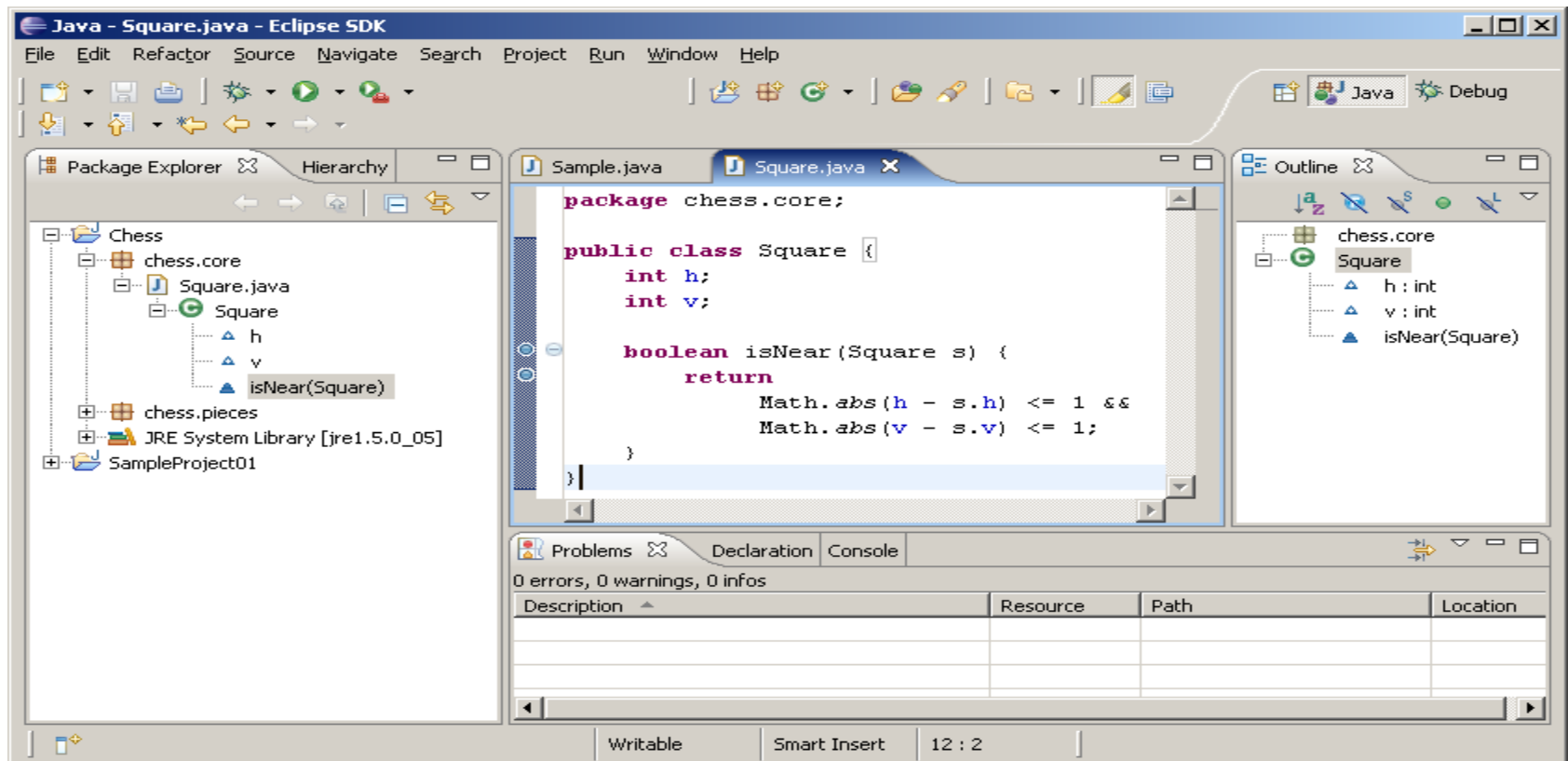
Классы Java и их синтаксис.

Члены классов. Eclipse

МГУ им. М.В.Ломоносова. Факультет ВМК.

Романов Владимир Юрьевич ©2025

- Классы инкапсулируют атрибуты (поля) и поведение (методы)
- Поля и методы являются *членами* класса



Классы Java и их синтаксис.

Объявление полей класса и экземпляра

МГУ им. М.В.Ломоносова. Факультет ВМК.

Романов Владимир Юрьевич ©2025

- Поля класса могут принадлежать всем экземплярам класса. В этом случае поля помечаются ключевым словом *static*.

```
class Point {  
    public int x, y;  
  
    static public final  
    Point ORIGIN = new Point(0,0);  
  
    public Point(int newX, int newY)  
        { x = newX; y = newY; }  
  
    public Point(Point p)  
        { x = p.x; y = p.y; }  
}
```

Поле экземпляра класса

Поле всех экземпляров класса

Классы Java и их синтаксис.

Использование полей класса

МГУ им. М.В.Ломоносова. Факультет ВМК.

Романов Владимир Юрьевич ©2025

- Для доступа к статическому полю (класса) используется имя класса, для доступа к полю экземпляра - имя экземпляра.

```
// Использование поля класса.  
Point p1 = Point.ORIGIN;  
  
// Создание экземпляра класса.  
Point p2 = new Point(3, 4);  
Point p3 = new Point(5, 6);  
  
// Использование поля экземпляра.  
p2.y = 100;  
p3.y = p2.y;  
}
```

Классы Java и их синтаксис.

Объявление методов класса и экземпляра

МГУ им. М.В.Ломоносова. Факультет ВМК.

Романов Владимир Юрьевич ©2025

- Методы класса принадлежат всем экземплярам класса. В этом случае методы помечаются ключевым словом *static*

```
class Point {  
    static  
    public int distance(Point p1, Point p2) {  
        int dx = p1.x - p2.x;  
        int dy = p1.y - p2.y;  
        return Math.sqrt(dx * dx + dy * dy);  
    }  
    public int distance(Point p) {  
        int dx = p.x - x;  
        int dy = p.y - y;  
        return Math.sqrt(dx * dx + dy * dy);  
    }  
}
```

Классы Java и их синтаксис.

Использование методов

МГУ им. М.В.Ломоносова. Факультет ВМК.

Романов Владимир Юрьевич ©2025

- Для доступа к статическому методу (класса) используется имя класса, для доступа к методу экземпляра - имя экземпляра класса.

```
Point p1 = new Point(3, 4);  
Point p2 = new Point(5, 6);  
  
// Использование метода класса.  
int d1 = Point.distanse(p1, p2);  
  
// Использование метода экземпляра.  
int d2 = p2.distanse(p1);  
}
```

Классы Java и их синтаксис.

Область видимости статических методов

МГУ им. М.В.Ломоносова. Факультет ВМК.

Романов Владимир Юрьевич ©2025

- В статических методах класса можно использовать только статические поля класса и вызывать только статические методы класса.

```
class Test {  
    public int p;  
  
    public void process() {  
    }  
    static public void main(String s[]) {  
        p = 1; // Ошибка  
        process(); // Ошибка  
  
        Test test = new Test();  
        test.p = 1;  
        test.process();  
    }  
}
```

Классы Java и их синтаксис.

Наследование классов

МГУ им. М.В.Ломоносова. Факультет ВМК.

Романов Владимир Юрьевич ©2025

- Классы могут быть независимы друг от друга
- Классы могут быть связаны *отношением наследования*. (суперкласс/подкласс).

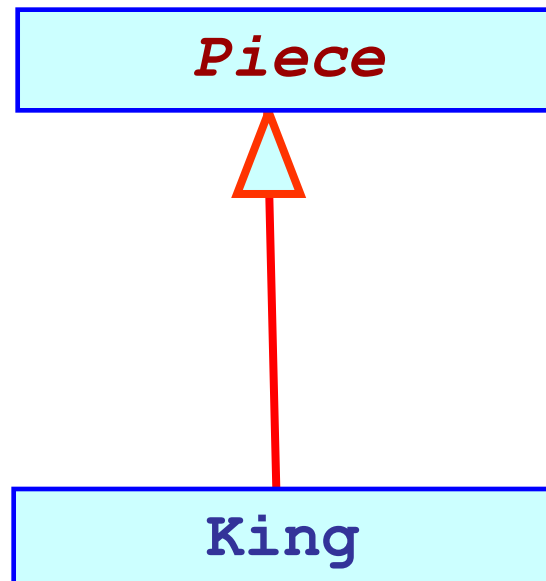
```
public class King extends Piece {  
    void move(Square s) {  
    }  
}
```

Изображение наследования классов Java на UML-диаграммах

МГУ им. М.В.Ломоносова. Факультет ВМК.

Романов Владимир Юрьевич ©2025

```
public class King extends Piece {  
    void move(Square s) {  
    }  
}
```



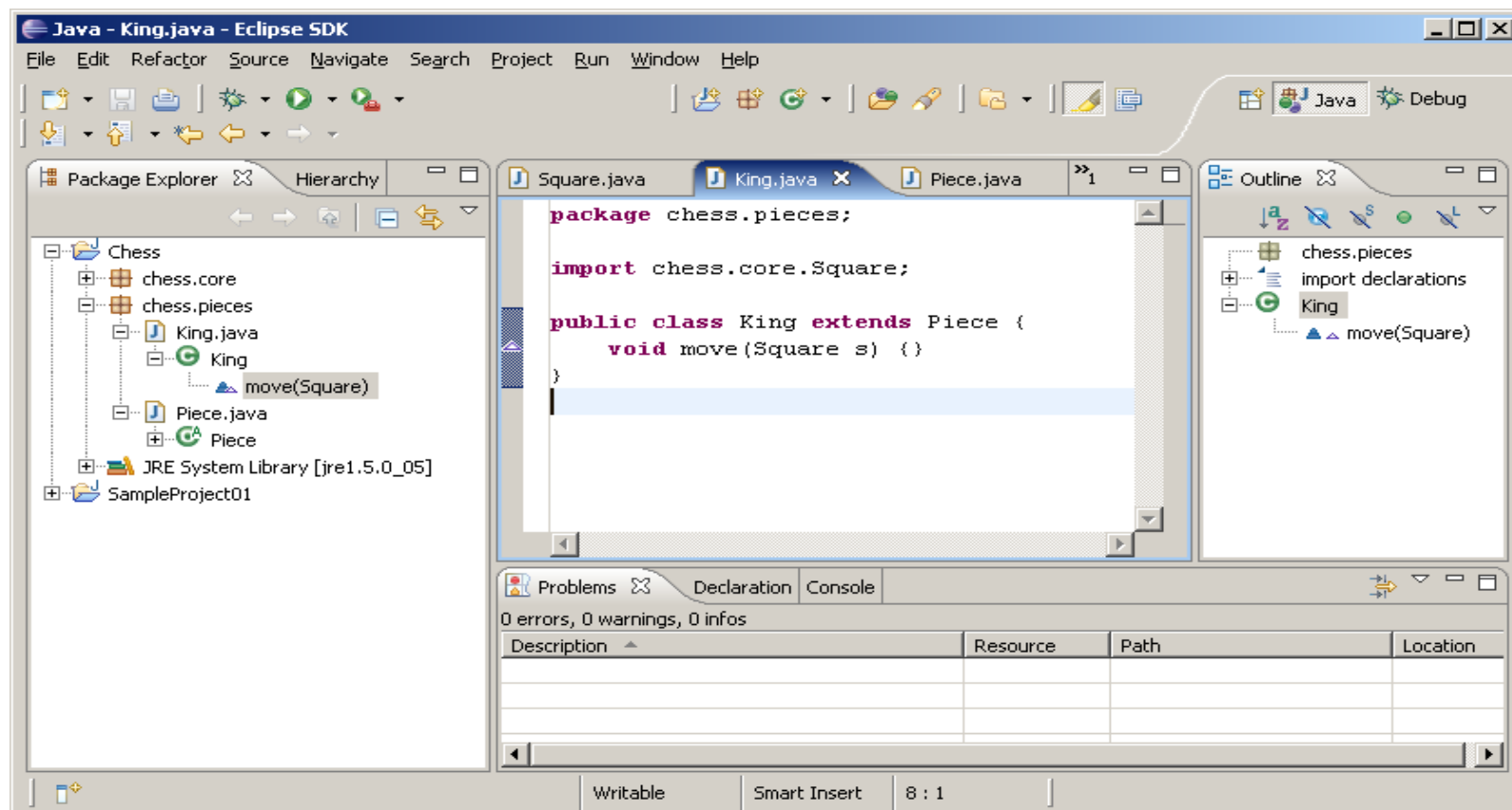
Классы Java и их синтаксис.

Наследование классов

МГУ им. М.В.Ломоносова. Факультет ВМК.

Романов Владимир Юрьевич ©2025

- Классы могут быть связаны отношением наследования (суперкласс/подкласс).



Интерфейсы в Java и их синтаксис.

Объявление интерфейсов

МГУ им. М.В.Ломоносова. Факультет ВМК.

Романов Владимир Юрьевич ©2025

- Интерфейсы описывают методы, которые должны быть реализованы в классах
- Могут описывать неизменяемые (**final**) переменные
- Метафора – что должен уметь делать класс

```
// Интерфейс для классов представляющих
// шахматные ходы
public interface Move {
    // Передвинуть фигуры на доске.
    void doMove();

    // Вернуть фигуры в исходное состояние.
    void undoMove();
}
```

Интерфейсы в Java и их синтаксис.

Реализация интерфейсов классами

МГУ им. М.В.Ломоносова. Факультет ВМК.

Романов Владимир Юрьевич ©2025

- Классы реализуют методы интерфейса
- Абстрактные классы могут реализовать не все методы интерфейса. Остальные реализуют потомки класса.

```
// Шахматный ход – рокировка.  
public class Castling implements Move {  
    void doMove() {  
        // Передвинуть короля и ладью  
    }  
  
    void undoMove() {  
        // Вернуть короля и ладью  
    }  
}
```

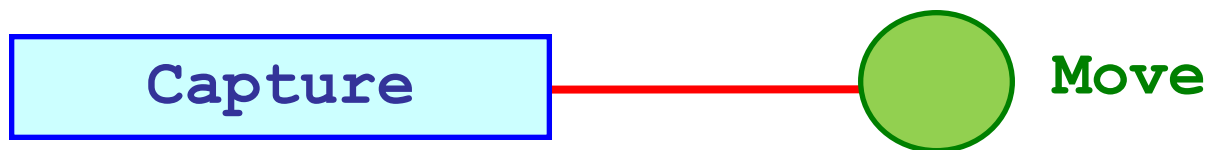
Интерфейсы в Java и их синтаксис.

Реализация интерфейсов классами

МГУ им. М.В.Ломоносова. Факультет ВМК.

Романов Владимир Юрьевич ©2025

```
// Шахматный ход – захват чужой фигуры
public class Capture implements Move {
    void doMove() {
        // Убрать чужую фигуру.
        // Передвинуть на ее место свою.
    }
    void undoMove() {
        // Передвинуть свою фигуру на прежнее место.
        // Поставить чужую фигуру.
    }
}
```



Интерфейсы в Java и их синтаксис.

Не изменяемые поля интерфейса.

МГУ им. М.В.Ломоносова. Факультет ВМК.

Романов Владимир Юрьевич ©2025

```
public interface BaseColors {  
    int RED = 1, GREEN = 2, BLUE = 4;  
}
```

```
int color = BaseColors.RED;
```

Интерфейсы в Java и их синтаксис.

Статические методы интерфейсов в Java 8.

МГУ им. М.В.Ломоносова. Факультет ВМК.

Романов Владимир Юрьевич ©2025

```
// Фигура ходит по диагонали.  
public interface DiagonalPiece {  
  
    static boolean isDiagonalPined(Square target) {  
        // Может ли фигура пойти на поле target?  
        // Откроется ли королю шах если диагональная  
        // фигура пойдет на клетку target  
    }  
  
}
```

Интерфейсы в Java и их синтаксис.

Статические методы интерфейсов в Java 8.

МГУ им. М.В.Ломоносова. Факультет ВМК.

Романов Владимир Юрьевич ©2025

```
// Фигура ходит по вертикали или горизонтали.  
public interface LinePiece {  
    static  
    boolean isVerticalPined(Square target) {  
        // Может ли фигура пойти на поле target?  
        // Откроется ли королю шах если  
        // фигура пойдет на клетку target  
    }  
  
    static  
    boolean isHorizontalPined(Square target) {  
        // Может ли фигура пойти на поле target?  
        // Откроется ли королю шах если  
        // фигура пойдет на клетку target  
    }  
}
```

Интерфейсы в Java и их синтаксис

Использование статических методов в Java 8

МГУ им. М.В.Ломоносова. Факультет ВМК.

Романов Владимир Юрьевич ©2025

```
public class Queen extends Piece
    implements DiagonalPiece, LinePiece
{
    boolean isCorrectMove(Square s) {
        if (isHorizontalPined(s))
            return false;
        if (isVerticalPined(s))
            return false;
        if (isDiagonalPined(s))
            return false;
        // ...
    }
    // ...
}
```

Интерфейсы в Java и их синтаксис.

Использование статических методов в Java 8

МГУ им. М.В.Ломоносова. Факультет ВМК.

Романов Владимир Юрьевич ©2025

```
public class Bishop extends Piece
    implements DiagonalPiece
{
    void isCorrectMove (Square s) {
        if (isDiagonalPined(s))
            return false;

        // ...
    }
    // ...
}
```

Интерфейсы в Java и их синтаксис

Использование статических методов в Java 8

МГУ им. М.В.Ломоносова. Факультет ВМК.

Романов Владимир Юрьевич ©2025

```
public class Rook extends Piece
    implements LinePiece
{
    boolean isCorrectMove(Square s) {
        if (isHorizontalPined(s))
            return false;
        if (isVerticalPined(s))
            return false;
        // ...
    }
    // ...
}
```

Интерфейсы в Java и их синтаксис

Расширение интерфейсов

МГУ им. М.В.Ломоносова. Факультет ВМК.

Романов Владимир Юрьевич ©2025

```
// Интерфейс для классов представляющих  
// шахматные ходы  
public interface Move {  
    void doMove();  
    void undoMove();  
  
    Board getBoard(); // Ошибка!!!  
}
```

Классы в уже оттранслированных
библиотеках не смогут реализовать
новый метод интерфейса

Интерфейсы в Java и их синтаксис

Использование умалчиваемых методов в Java 8

МГУ им. М.В.Ломоносова. Факультет ВМК.

Романов Владимир Юрьевич ©2025

```
// Интерфейс для классов представляющих  
// шахматные ходы  
public interface Move {  
    void doMove();  
    void undoMove();  
  
    default Board getBoard() {  
        // ...  
    }  
}
```

Для классов в уже
оттранслированных библиотеках
используется умалчиваемая
реализация метода интерфейса

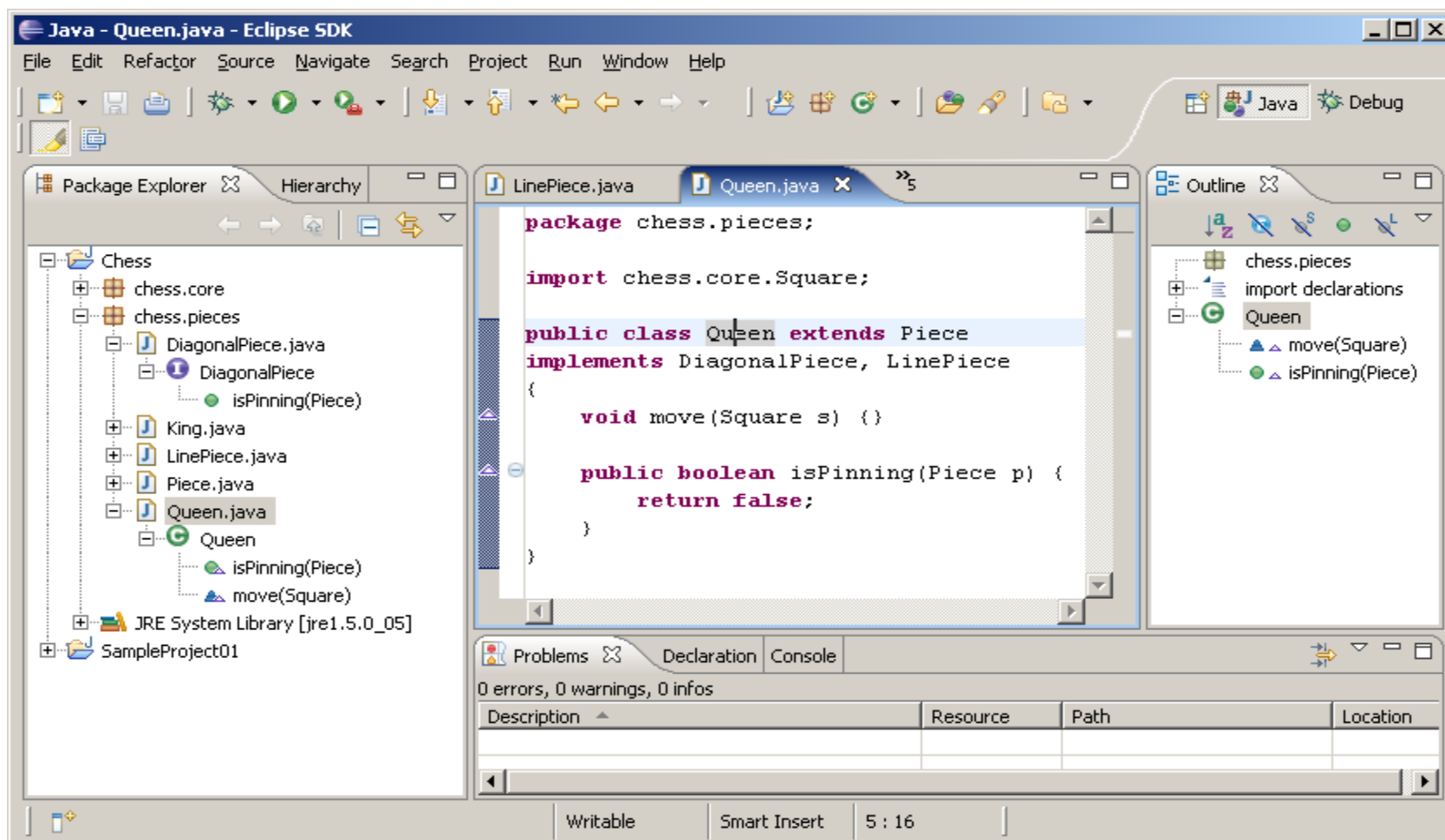
Классы Java и их синтаксис.

Реализация интерфейсов. Eclipse

МГУ им. М.В.Ломоносова. Факультет ВМК.

Романов Владимир Юрьевич ©2025

- Классы могут реализовывать один и более *интерфейсов*



Классы Java и их синтаксис.

Группирование классов в пакеты

МГУ им. М.В.Ломоносова. Факультет ВМК.

Романов Владимир Юрьевич ©2025

- Логически связанные классы группируются в *пакеты*

```
package chess.pieces;

import chess.core.Square;

public class Queen extends Piece
    implements DiagonalPiece, LinePiece
{
    void move(Square s) {}

    public boolean isPinning(Piece p) {
        return false;
    }
}
```

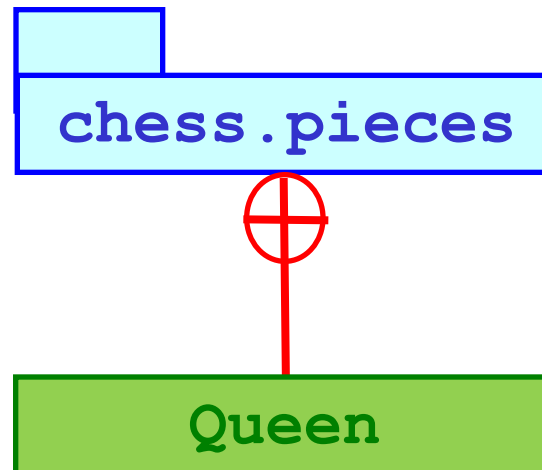
Изображение пакетов на UML-диаграммах

МГУ им. М.В.Ломоносова. Факультет ВМК.

Романов Владимир Юрьевич ©2025

```
package chess.pieces;
```

```
public class Queen extends Piece  
implements DiagonalPiece, LinePiece  
{  
}
```



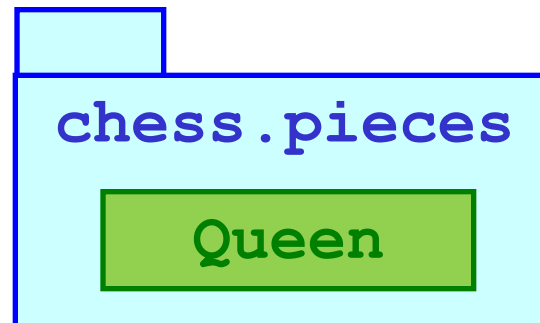
Изображение пакетов на UML-диаграммах

МГУ им. М.В.Ломоносова. Факультет ВМК.

Романов Владимир Юрьевич ©2025

```
package chess.pieces;
```

```
public class Queen extends Piece  
implements DiagonalPiece, LinePiece  
{  
}
```



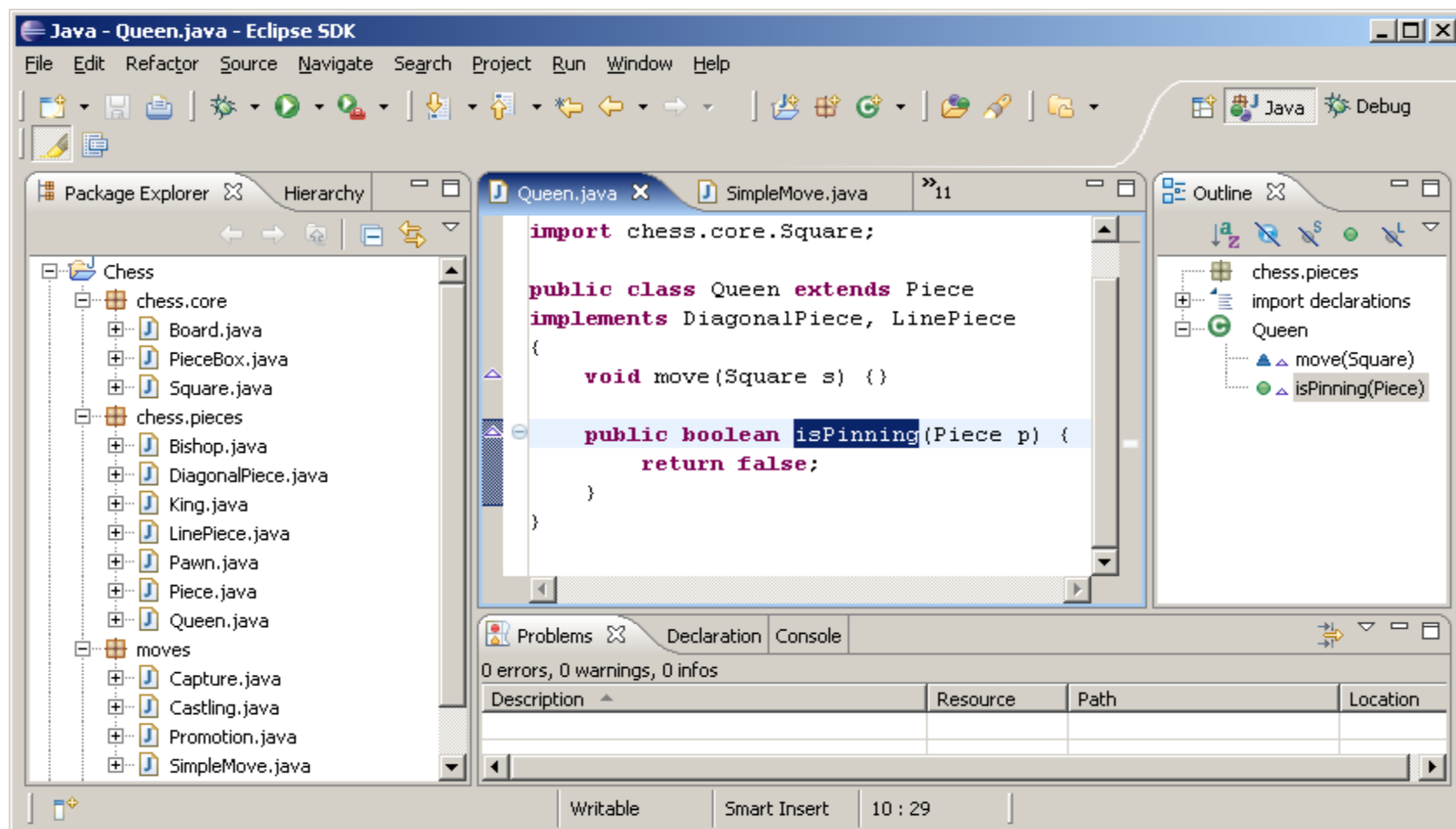
Классы Java и их синтаксис.

Группирование классов в пакеты

МГУ им. М.В.Ломоносова. Факультет ВМК.

Романов Владимир Юрьевич ©2025

- Логически связанные классы группируются в пакеты



Классы Java и их синтаксис.

Модификаторы классов

МГУ им. М.В.Ломоносова. Факультет ВМК.

Романов Владимир Юрьевич ©2025

- Модификатор *public*

Класс доступен для всех других классов. Отсутствие модификатора означает доступность класса только классам внутри пакета содержащего данный класс.

- Модификатор *private*

Этот модификатор допустим только для классов, которые *вложены (nested)* в другие классы

- Модификатор *abstract*

Запрет на создание экземпляров класса

- Модификатор *final*

Запрет на создание подклассов данного класса

Изображение модификаторов класса на UML

МГУ им. М.В.Ломоносова. Факультет ВМК.

Романов Владимир Юрьевич ©2025

```
abstract public class Piece  
{  
}
```

Piece
{*abstract*}

Конструкторы классов

МГУ им. М.В.Ломоносова. Факультет ВМК.

Романов Владимир Юрьевич ©2025

- *Конструктор* – метод который создает экземпляр класса. Имя конструктора совпадает с именем класса. Допустимы несколько конструкторов с различными параметрами
- Конструктор используется для инициализации объектов
- Тело класса содержит по меньшей мере один конструктор
- Конструктор возвращает указатель на созданный объект. Оператор **return** в конструкторе отсутствует
- Для создания *экземпляров* класса используется ключевое слово **new** с именем конструктора

```
King piece = new King();
```

Умалчиваемые конструкторы классов

МГУ им. М.В.Ломоносова. Факультет ВМК.

Романов Владимир Юрьевич ©2025

- Умалчиваемый конструктор не имеет аргументов.

Предоставляется по умолчанию платформой Java если нет ни одного явно определенного конструктора

- При определении хотя бы одного явного конструктора необходимо объявить явно и умалчиваемый конструктор

```
class King {}  
King piece = new King();
```

```
class King {  
    King(Square s) {}  
}  
King piece = new King(); // Ошибка компиляции!!!
```

```
class King {  
    King() {}  
    King(Square s) {}  
}  
King piece = new King(); // Ошибки компиляции теперь нет
```

Цепочка конструкторов

МГУ им. М.В.Ломоносова. Факультет ВМК.

Романов Владимир Юрьевич ©2025

- *Цепочка конструкторов* - вызов внутри класса одного конструктора другим конструктором. Цепочки используются для разделения общего кода между конструкторами
- Вызов цепочки конструкторов:
this (список аргументов)

```
class Piece {  
    boolean isWhite;  
    Square square;  
  
    Piece(Square square, boolean isWhite)  
        { this.square = square; this.isWhite = isWhite; }  
    Piece(Square square)  
        { this(square, true); }  
    Piece()  
        { this(null, true); }  
}
```

Конструкторы суперкласса

МГУ им. М.В.Ломоносова. Факультет ВМК.

Романов Владимир Юрьевич ©2025

- Объекты суперкласса строятся до построения объекта-подкласса
- Для инициализации членов суперкласса используется вызов:
 - **super**(список-аргументов)
- Первой строкой конструктора могут быть:
 - **super**(список-аргументов)
 - **this**(список-аргументов)
- В одном конструкторе нельзя использовать одновременно и **super** и **this**
- Компилятор предоставляет *неявный вызов конструктора* **super()** для всех конструкторов подклассов

Освобождение памяти объектов

МГУ им. М.В.Ломоносова. Факультет ВМК.

Романов Владимир Юрьевич ©2025

- В Java отсутствует понятие *деструктора* для неиспользуемых объектов
- Освобождение памяти выполняется автоматически виртуальной машиной Java
- Сборщик мусора освобождает память объектов, на которые нет ссылок
- Связь между объектом и ссылкой на объект уничтожается при задании нового значения ссылке на объект

`objectReference = null;`

- Объект без ссылок – кандидат на освобождение при сборке мусора

Сборка мусора

МГУ им. М.В.Ломоносова. Факультет ВМК.

Романов Владимир Юрьевич ©2025

- Можно явно запросить сборку мусора:

```
System.gc()
```

- Метод объекта *finalize* будет выполняться непосредственно перед сборкой мусора.

```
@Override  
protected void finalize() {  
    // TODO Здесь освободить захваченные ресурсы  
}
```

Используется при:

- освобождении памяти выделенной с помощью *native*-методов
- закрытии открытых объектом файлов перед тем, как память объекта будет освобождена.

Поля классов

МГУ им. М.В.Ломоносова. Факультет ВМК.

Романов Владимир Юрьевич ©2025

- *Поля*
 - Часть определения классов
 - Состояние объекта хранится в полях
 - Каждый экземпляр получает собственную копию переменных экземпляра
- В месте объявления поля могут быть *инициализированы*
- Если поля не инициализируются явно, то используются *умалчиваемые значения*

```
public class Piece {  
    public boolean isWhite;  
    protected Square square;  
}
```

модификатор
доступа

ТИП

ИМЯ

Методы

МГУ им. М.В.Ломоносова. Факультет ВМК.

Романов Владимир Юрьевич ©2025

- *Методы* определяют
 - Как объект отвечает на сообщение
 - Поведение класса
 - Все методы принадлежат классу

```
public class Square {  
    private boolean isNear(Square s) { ... }  
}
```

модификатор
доступа

тип

имя

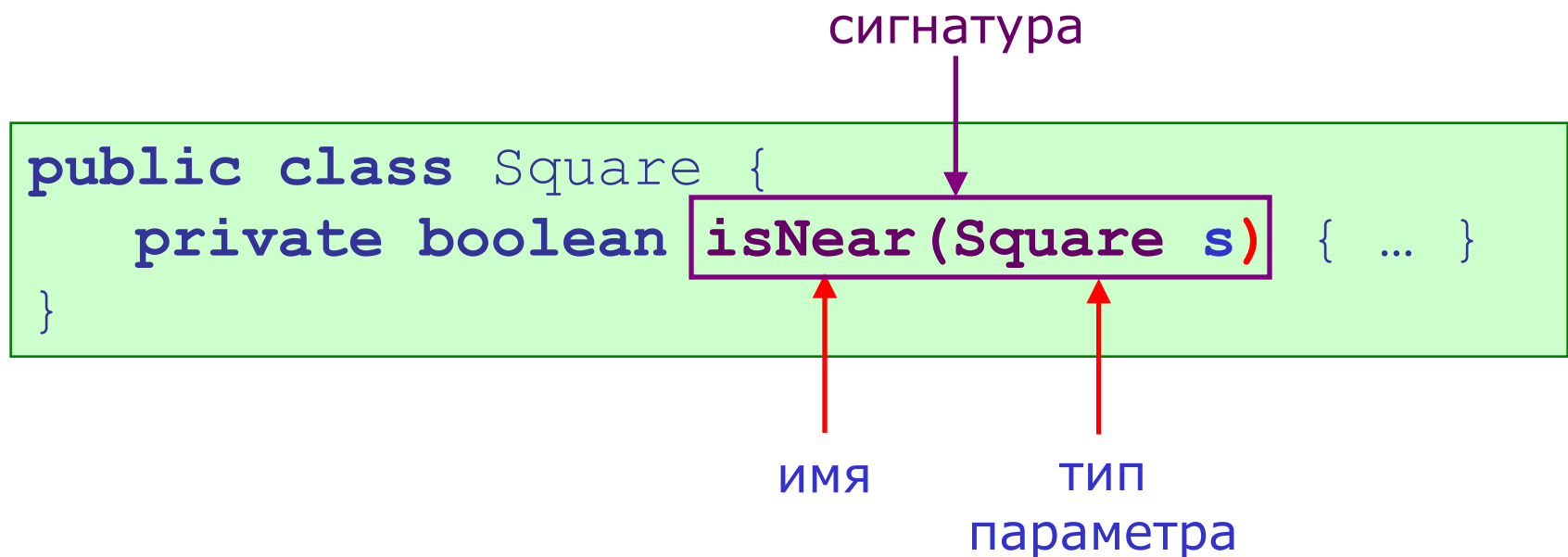
список
параметров

Сигнатура метода

МГУ им. М.В.Ломоносова. Факультет ВМК.

Романов Владимир Юрьевич ©2025

- Класс может иметь множество методов с одинаковыми именами
- Каждый метод должен иметь другую сигнатуру
- Сигнатура – количество аргументов и типы аргументов



Параметры метода

МГУ им. М.В.Ломоносова. Факультет ВМК.

Романов Владимир Юрьевич ©2025

- Аргументы (параметры) пересылаются:
 - примитивные типы по значению
 - ссылки на объекты для ссылочных типов
- Примитивные значения не могут быть модифицированы при пересылке в качестве аргументов

```
public void method1 () {  
    int a = 0;  
    System.out.println(a); // вывод 0  
    method2(a);  
    System.out.println(a); // вывод 0  
}  
  
void method2(int a) {  
    a = a + 1;  
}
```

Вызов метода и возврат из метода

МГУ им. М.В.Ломоносова. Факультет ВМК.

Романов Владимир Юрьевич ©2025

- И для вызова метода используется оператор точка
 - для методов экземпляра и для методов класса
 - если вызываемый метод в том же классе, то оператор точка не требуется

Вызов через оператор точка метода класса

```
public class King {  
    boolean wasCastling() { ... }  
  
    boolean isCorrectMove(Square newSquare) {  
        Square s = King.oppositeKing().square;  
        return !s.isNear(newSquare) && !wasCastling();  
    }  
}
```

Вызов через оператор точка
метода экземпляра

метод определен в
том же классе

Перекрытие метода

МГУ им. М.В.Ломоносова. Факультет ВМК.

Романов Владимир Юрьевич ©2025

- Метод суперкласса перекрывается методом подкласса с той же сигнатурой

Перекрытый метод

```
public class Piece {  
    boolean isCorrectMove (Square newSquare) {...}  
}  
  
public class King extends Piece {  
    boolean isCorrectMove (Square newSquare) {  
        if (!super.isCorrectMove (newSquare))  
            return false;  
  
        Square s = King.oppositeKing().square;  
        return !s.isNear(square) && !wasCastling();  
    }  
}
```

Перекрывающий метод

Вызов перекрытого метода

Метод *Main*

МГУ им. М.В.Ломоносова. Факультет ВМК.

Романов Владимир Юрьевич ©2025

- Приложение не может быть выполнено без хотя бы одного класса с методом *main*
- Виртуальная машина Java загружает класс и начинает выполнение с метода *main*

```
public class Chess {  
    static public void main(String[] args) {  
    }  
}
```

- *public* – метод может быть вызван любым объектом
 - *public*
- *static* – нет необходимости сначала создавать объект
- *void* – этот метод ничего не возвращает

Инкапсуляция

МГУ им. М.В.Ломоносова. Факультет ВМК.

Романов Владимир Юрьевич ©2025

- Объект – содержит данные и действия, которые можно выполнить над данными.
- Принцип сокрытия информации – объект знает о себе все, другие объекты запрашивают информацию об этом объекте.
- Все объекты отличаются друг от друга и программа – это обмен сообщениями между объектами
- Для сокрытия информации используется модификатор доступа *private*

Статические члены

МГУ им. М.В.Ломоносова. Факультет ВМК.

Романов Владимир Юрьевич ©2025

- Статические поля и методы принадлежат классу
- При изменении статического значения одним из объектов данного класса изменяется значение для всех объектов данного класса
- Статические методы и поля могут быть доступны без создания экземпляров класса

Конечные (*final*) члены класса

МГУ им. М.В.Ломоносова. Факультет ВМК.

Романов Владимир Юрьевич ©2025

- Поле с модификатором *final* не может быть модифицировано. Это аналог констант в языке Java
- Константы связанные с классом обычно для простоты доступа объявляются с модификаторами *static final*
- Общепринято константы записывать большими буквами

```
public class Piece {  
    static final public KING = 1;  
    static final public QUEEN = 2;  
    static final public PAWN = 3;  
}
```

Абстрактные классы

МГУ им. М.В.Ломоносова. Факультет ВМК.

Романов Владимир Юрьевич ©2025

- Невозможно создать экземпляр абстрактного класса. Предполагается, что они будут суперклассами для других классов
- Методы с модификатором *abstract* не имеют реализации
- Если класс имеет абстрактные методы, то он должен быть объявлен абстрактным

```
abstract public class Piece {  
    boolean isCorrectMove(Square s) { ... }  
    abstract void makeMove(Square s);  
}
```

Пакеты классов

МГУ им. М.В.Ломоносова. Факультет ВМК.

Романов Владимир Юрьевич ©2025

- Классы могут быть сгруппированы в пакеты

```
package chess.pieces;  
  
abstract public class Piece {  
    boolean isCorrectMove(Square s) { ... }  
    abstract void makeMove(Square s);  
}
```

- Различные пакеты могут иметь классы с одинаковыми именами
- По соглашению имена пакетов задаются в нижнем регистре

Видимость классов

МГУ им. М.В.Ломоносова. Факультет ВМК.

Романов Владимир Юрьевич ©2025

- Для ссылки на классы в том же пакете можно использовать только имя класса
- Для ссылки на классы из других пакетов необходимо использовать полностью квалифицированное имя класса

```
package chess.movies;  
  
public class Castling extends Move {  
    void doMove(Square s) {  
        if (chess.pieces.King.wasCasling())  
            return;  
    }  
}
```

Импорт классов

МГУ им. М.В.Ломоносова. Факультет ВМК.

Романов Владимир Юрьевич ©2025

- Предложение *import* используется для того, что бы сделать классы непосредственно доступными

```
package chess.movies;

import game.core.*;
import chess.pieces.King;
import chess.pieces.Rook;

public class Castling extends Move {
    void doMove(Square s) {
        if (King.wasCasling()) return;
        if (Rook.wasMoved()) return;
    }
}
```

Импорт классов

МГУ им. М.В.Ломоносова. Факультет ВМК.

Романов Владимир Юрьевич ©2025

- Предложение *import* используется для того, что бы сделать классы непосредственно доступными

```
package chess.movies;

import chess;

public class Castling extends Move {
    void doMove(Square s) {
        if (pieces.King.wasCasling()) return;
        if (pieces.Rook.wasMoved()) return;
    }
}
```

Импорт статических методов классов в Java8

МГУ им. М.В.Ломоносова. Факультет ВМК.

Романов Владимир Юрьевич ©2025

- Пакет `java.lang` импортируется по умолчанию.

```
package samples;

//import java.lang.Math;

public class StaticImportSample {

    public static void main(String[] args) {
        Math.sin(Math.PI);
    }

}
```

Импорт статических методов классов в Java8

МГУ им. М.В.Ломоносова. Факультет ВМК.

Романов Владимир Юрьевич ©2025

- Предложение **import** используется для импорта статической функции **sin** и статического поля **PI** класса **Math**.

```
package samples;

import static java.lang.Math.sin;
import static java.lang.Math.PI;

public class StaticImportSample {

    public static void main(String[] args) {
        sin(PI);
    }

}
```

Импорт статических методов классов в Java8

МГУ им. М.В.Ломоносова. Факультет ВМК.

Романов Владимир Юрьевич ©2025

- Предложение *import* используется для импорта всех статических методов и полей класса **Math**.

```
package samples;

import static java.lang.Math.*;

public class StaticImportSample {

    public static void main(String[] args) {
        sin(PI);
    }

}
```

Пакеты ядра Java (1)

МГУ им. М.В.Ломоносова. Факультет ВМК.

Романов Владимир Юрьевич ©2025

- **java.lang**
 - неявно импортируется во все пакеты
 - предоставляет фундаментальные классы языка программирования Java (**Object**, **String**, **StringBuffer**, ...)
- **java.util**
 - библиотека классов-коллекций
 - модель для программирования событий
 - классы для работы с датами и временем
 - классы для локализации программ на различных национальных языках
- **java.io**
 - работа через потоки ввода/вывода
 - сериализация
 - работа с файловой системой

Пакеты ядра Java (2)

МГУ им. М.В.Ломоносова. Факультет ВМК.

Романов Владимир Юрьевич ©2025

- **java.math**
 - стандартные математические функции
 - работа с большими целыми числами **BigInteger**
 - работа с большими вещественными числами **BigDecimal**
- **java.sql**
 - классы для анализа структуры реляционной базы данных
 - классы для выполнения запросов к базе данных на языке SQL
- **java.text**
 - классы и интерфейсы для обработки текста, дат, чисел способом независимым от национальных языков