Николай Прохоренок

OpenCV и Java
ОБРАБОТКА ИЗОБРАЖЕНИЙ И КОМПЬЮТЕРНОЕ ЗРЕНИЕ

Санкт-Петербург
«БХВ-Петербург»
2018

УДК 004.93
ББК 32.973.26-018.1
П84

Прохоренок Н. А.

П84 OpenCV и Java. Обработка изображений и компьютерное зрение. — СПб.: БХВ-Петербург, 2018. — 320 с.: ил. — (Профессиональное программирование)

ISBN 978-5-9775-3955-5

Книга знакомит с современными технологиями компьютерного зрения, позволяющими машинам, роботам, веб-камерам и другим устройствам распознавать изображения. Приведено описание библиотеки компьютерного зрения OpenCV применительно к языку программирования Java. Объясняется, как загружать и сохранять изображения в различных форматах, захватывать кадры с веб-камеры в режиме реального времени, выполнять обработку, трансформацию и сегментацию изображения, применять к изображению фильтры. На практических примерах рассмотрены алгоритмы компьютерного зрения, предназначенные для обнаружения, классификации и отслеживания объектов, выделения границ и контуров объектов, поиска объектов по шаблону, особым точкам, цвету или обученному классификатору.

Для программистов

УДК 004.93
ББК 32.973.26-018.1

Группа подготовки издания:

Главный редактор	*Екатерина Кондукова*
Зам. главного редактора	*Евгений Рыбаков*
Зав. редакцией	*Екатерина Капалыгина*
Редактор	*Григорий Добин*
Компьютерная верстка	*Ольги Сергиенко*
Корректор	*Зинаида Дмитриева*
Дизайн обложки	*Марины Дамбиевой*

Подписано в печать 28.02.18.
Формат 70×100 1/16. Печать офсетная. Усл. печ. л. 25,8.
Тираж 1000 экз. Заказ №
"БХВ-Петербург", 191036, Санкт-Петербург, Гончарная ул., 20.

ООО "Печатное дело",
142300, МО, г. Чехов, ул. Полиграфистов, д. 1

ISBN 978-5-9775-3955-5

Оглавление

Введение

Добро пожаловать в мир OpenCV!

OpenCV (от англ. Open Source Computer Vision Library) — это библиотека алгоритмов компьютерного зрения с открытым исходным кодом. Библиотека распространяется по лицензии BSD, следовательно, она может свободно использоваться в академических и коммерческих целях. За долгое время своего существования библиотека приобрела обширную аудиторию пользователей и на сегодняшний день OpenCV является стандартом в области компьютерного зрения. Библиотека написана на языках C/C++, имеет привязки к языкам Python, Java, Matlab и др. В этой книге мы рассмотрим привязку OpenCV к языку программирования Java SE (*SE*, Standard Edition) применительно к операционной системе Windows.

Что же такое компьютерное зрение? *Компьютерное зрение* — это теория и технология получения информации из изображений. Причем изображение может быть как отдельной фотографией, так и последовательностью кадров видео, полученной из видеофайла или с видеокамеры (например, с камеры наружного наблюдения, с веб-камеры или со стереокамеры) в режиме реального времени.

Типичными задачами компьютерного зрения являются обнаружение, отслеживание и классификация объектов. Цель *обнаружения* — найти на изображении объект. Для этого мы можем использовать границы (контуры), особые точки (например, углы), информацию о цвете и т. д. *Отслеживание* применяется в работе с камерами наружного наблюдения. При этом также можно воспользоваться контурами, особыми точками и информацией о цвете, а можно и вычесть фон из текущего кадра (при условии, что камера статична). Чтобы выполнить *классификацию*, нужно распознать обнаруженный объект. Для распознавания можно выполнить попиксельное сравнение с шаблоном, сравнить контуры или особые точки, осуществить поиск по обученному классификатору (например, с помощью каскадов Хаара) и др. В последнее время для распознавания объектов все чаще используются глубокие сверточные нейронные сети. Поддержка таких сетей была добавлена в OpenCV в версии 3.3.0.

С компьютерным зрением тесно связана еще одна технология — *обработка изображения*. Действительно, прежде чем искать объекты на изображении, необходимо выполнить обработку. В частности, в большинстве случаев надо изменить раз-

меры изображения и выполнить выравнивание гистограммы. Кроме того, нужно сгладить изображение, чтобы избавиться от цифрового шума, который возникает при использовании дешевых камер, а также при различных специфических режимах съемки (например, при высоких значениях ISO или при длительных выдержках). Многие алгоритмы очень чувствительны к шумам, и их наличие может привести, например, к неправильно найденным границам объекта или ложным особым точкам.

Ученые занимаются исследованиями в области компьютерного зрения уже очень давно, но до сих пор так и не приблизились к способностям человека распознавать образы. В чем же проблемы? Почему компьютер может с легкостью сложить очень большие числа (что человеку часто не под силу), а вот распознать образ на изображении не в состоянии (человек, заметьте, справляется с этим без проблем)?

Первая проблема заключается в том, что цвет пиксела в цифровом изображении задается числовыми значениями. Причем этих чисел в полноцветных изображениях три: первое число задает количество красного цвета, второе — зеленого и третье — синего (цветовая модель RGB). Вторая проблема состоит в том, что пиксел имеет квадратную форму, а в видео даже встречается прямоугольная форма элементов изображения. Попробуйте нарисовать прямую линию под углом, увеличить масштаб и посмотреть, что получилось. При увеличенном масштабе даже человек не сможет сказать, что это линия. Третья проблема возникает при сжатии изображения, которое используется для уменьшения размера файла. Например, при сохранении в формате JPEG появляются артефакты сжатия, с которыми нужно как-то бороться.

Существует множество и других сложностей. Например, при работе с изображениями мы имеем дело с двумерным пространством, а человек, имея два глаза, видит трехмерную картинку. Попробуйте закрыть один глаз и попасть вилкой в розетку. Согласитесь, что с двумя открытыми глазами получается гораздо лучше. При распознавании образов человек воспринимает не только контуры, точки или символы, но и контекст сцены или смысл текста в целом. Например, попробуйте прочитать этот текст:

```
Не иеемт занчнеия, в кокам пряокде рсапожолены бкувы в солве!
```

Согласитесь, что у вас не возникло никаких проблем с понятием смысла, при этом вы не распознавали буквы внутри каждого слова, иначе бы ничего не поняли.

Еще одна проблема заключается в изменчивости формы объекта. Посмотрим, например, на человека. Он может стоять, сидеть, лежать, нагнуться вперед, назад или в бок и т. д. Поскольку мы имеем дело с двумерными изображениями, то части тела могут перекрываться другими частями тела или другими объектами. Человек может быть одет в разную одежду, носить очки, иметь усы и бороду и т. д. Все это очень сильно осложняет процесс распознавания образов.

Если мы обратимся к обитателям дикой природы, то заметим, что для сохранения собственной жизни животные маскируются под окружающую обстановку. Попробуйте отличить палочника от ветки дерева. Кроме того, животные имеют различную окраску, которая очень затрудняет отделение одного животного от другого, —

например, полосы у зебр. Так что не все так просто, как может показаться на первый взгляд.

В библиотеке OpenCV реализовано очень большое количество алгоритмов для обработки изображений и компьютерного зрения. Некоторые алгоритмы реализованы в виде отдельных классов, а большинство — в виде статических методов какого-либо класса. Классы в OpenCV версии 3.3.0 в Java разделены на следующие пакеты:

- `org.opencv.imgcodecs` — включает класс `Imgcodecs`, с помощью которого можно загрузить изображение из файла или буфера, а также сохранить изображение в файл или в буфер в различных форматах (например, в JPEG);
- `org.opencv.core` — содержит основные классы библиотеки, реализующие базовые структуры (векторы, матрицы и т. д.). Кроме того, пакет включает вспомогательные классы (например, классы `Point`, `Rect` и др.). Класс `Core` из этого пакета содержит статические методы, с помощью которых можно выполнить различные операции с матрицами;
- `org.opencv.imgproc` — включает классы, предназначенные для обработки и анализа изображений;
- `org.opencv.features2d` — содержит классы, с помощью которых можно находить и сравнивать особые точки;
- `org.opencv.photo` — включает классы, предназначенные для создания HDR-изображений;
- `org.opencv.video` — содержит классы, предназначенные для работы с видеоданными (анализ движения и отслеживание объектов);
- `org.opencv.videoio` — с помощью класса `VideoCapture` из этого пакета можно загружать кадры из видеофайла или последовательности кадров, а также захватывать кадры в режиме реального времени с камер наружного видеонаблюдения, веб-камер и др.;
- `org.opencv.calib3d` — содержит классы, с помощью которых можно выполнить калибровку камеры, работать со стереокамерами и обрабатывать трехмерные данные;
- `org.opencv.objdetect` — включает классы для поиска объектов на изображении. С помощью обученных классификаторов можно искать людей, лица, глаза, нос, узнать, улыбается человек или нет, и т. д. При большом желании можно обучить собственный классификатор с произвольным назначением или загрузить уже обученный из Интернета;
- `org.opencv.ml` — содержит классы, предназначенные для машинного обучения;
- `org.opencv.dnn` — включает классы для работы с нейронными сетями. Можно загружать модели, обученные в популярных библиотеках Caffe, TensorFlow и Torch;
- `org.opencv.utils` — содержит вспомогательный класс `Converters`, который в основном используется для внутренних нужд библиотеки;

- `org.opencv.android` — включает классы для работы с ОС Android. Пакет доступен только в составе дистрибутива под Android.

В этой книге мы рассмотрим большинство классов и методов, предназначенных для обработки изображений и компьютерного зрения. При изучении материала от вас потребуется знание основ языка Java SE 8, а также математики на уровне средней школы. Сами алгоритмы мы рассматривать не станем, т. к., во-первых, исходный код библиотеки доступен для просмотра, во-вторых, частичное описание алгоритмов есть в документации, в-третьих, в документации есть ссылки на оригинальные научные статьи, которые и нужно изучать, если возникнет в этом необходимость. И, наконец, описание алгоритмов потребует для книги очень большого объема, а от вас очень глубокого знания математики. Эта книга для всех, поэтому мы будем учиться пользоваться уже реализованными алгоритмами.

Параллельно с изучением книги советую посмотреть два видеокурса по компьютерному зрению, чтобы у вас сложилось более целостное представление о предметной области. Первый видеокурс доступен на канале Антона Конушина:

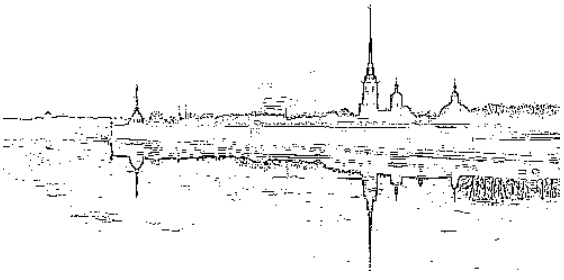
https://www.youtube.com/channel/UC9qIqkpLGA4xg6Nz6BE4aRA

Второй видеокурс доступен на канале Дмитрия Полевого:

https://www.youtube.com/channel/UChNX3pI3bTHLNge294F7yDA

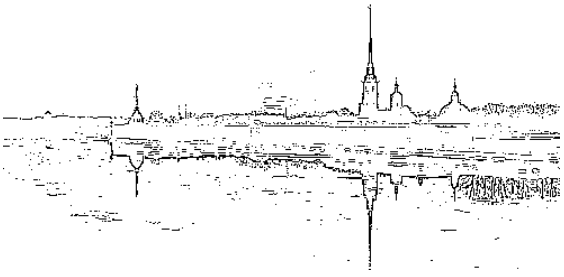
Обратите внимание: на каналах есть различные версии видеокурсов, отличающиеся годом выхода. Советую вам посмотреть их все, чтобы увидеть тенденции развития компьютерного зрения.

Все листинги из этой книги вы найдете в файле Listings.doc, электронный архив с которым можно загрузить с FTP-сервера издательства «БХВ-Петербург» по ссылке: **ftp://ftp.bhv.ru/9785977539555.zip** или со страницы книги на сайте **www.bhv.ru** (см. *приложение*).



ЧАСТЬ I

Загрузка изображения



ГЛАВА 1

Первые шаги

Прежде чем мы начнем рассматривать возможности библиотеки OpenCV, необходимо сделать одно замечание. Не забывайте, что книги по программированию нужно не только читать, но и выполнять все приведенные в них примеры, а также экспериментировать, что-либо в этих примерах изменяя. Поэтому, если вы удобно устроились на диване и настроились просто читать, у вас практически нет шансов изучить предлагаемую вам технологию! Материал в книге изложен очень компактно, без «воды», поэтому просто чтение вам быстро наскучит, и вы станете бегло просматривать листинги, так ничего из них и не усваивая. Однако, если вы сядете перед компьютером и будете не только читать книгу, но и запускать на выполнение все примеры, то процесс окажется увлекательным и запоминающимся, — ведь вы увидите результаты выполнения, но перед этим, наверняка, допустите множество ошибок (например, забудете импортировать класс или импортируете его из другого пакета). Обязательно найдите и исправьте все свои ошибки. Именно поиск и исправление ошибок научат вас очень многому и не дадут вам скучать. Чем больше вы будете делать самостоятельно, тем большему научитесь. Наберитесь терпения, и вперед!

1.1. Установка OpenCV

Вначале нужно скачать и установить библиотеку OpenCV. Для этого открываем в веб-браузере страницу **http://opencv.org/releases.html** и переходим по ссылке **Win pack** из раздела **3.3.0**. Сохраняем файл opencv-3.3.0-vc14.exe на рабочем столе и запускаем его на выполнение. Скачанный нами файл представляет собой архив, а не программу установки, поэтому открывшееся окно (рис. 1.1) просто предлагает вам выбрать место для его распаковки, — нажимаем кнопку **Extract** и распаковываем архив прямо на рабочий стол. Процесс распаковки архива показан на рис. 1.2.

Далее заходим в созданную распаковщиком папку и переименовываем вложенную папку opencv в opencv_3_3, а затем вырезаем эту папку со всем содержимым и вставляем ее в корень диска C или любого другого. В результате путь до файла opencv-330.jar должен стать таким: C:\opencv_3_3\build\java. Если вы решили выбрать другое

место, то помните, что в пути не должно быть русских букв, поэтому корень диска — самое лучшее место для установки библиотеки.

В OpenCV версии 3 некоторые популярные алгоритмы защищены патентами, поэтому они вынесены в отдельный (не свободный) модуль. Эти алгоритмы можно использовать в образовательных целях, но коммерческое их применение требует оплаты. Модуль с защищенными алгоритмами по умолчанию не доступен в Java, однако эти алгоритмы доступны в OpenCV версии 2.4.13. Считаю необходимым представить вам и эти алгоритмы, поэтому некоторые примеры в книге будут написаны для версии OpenCV 2.4.13. Поэтому советую дополнительно установить эту версию в папку C:\opencv_2_4. Путь до файла opencv-2413.jar должен быть таким: C:\opencv_2_4\build\java.

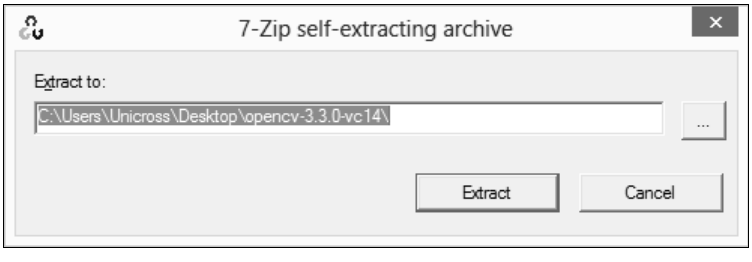


Рис. 1.1. Выбор места установки OpenCV

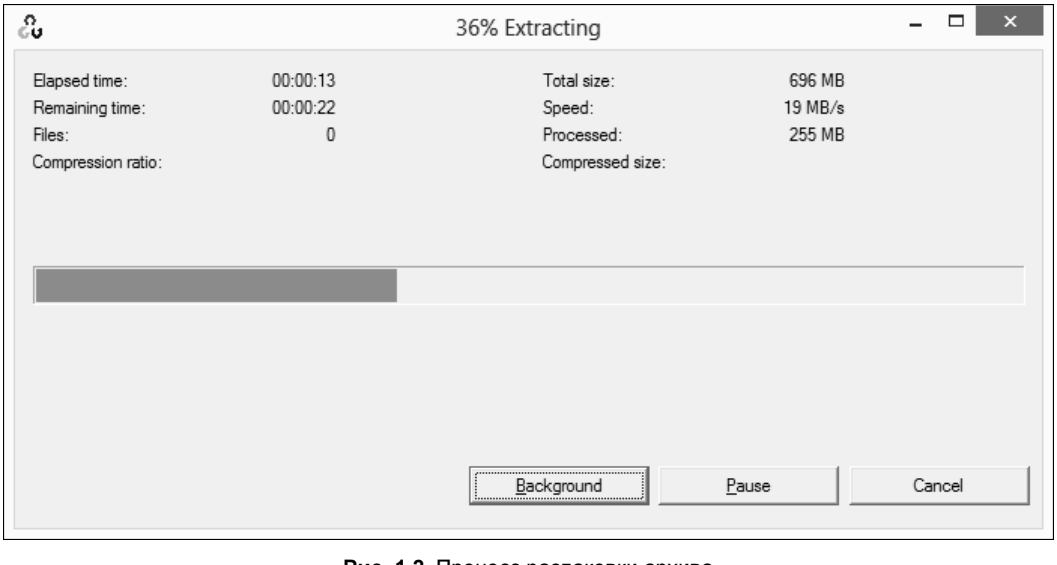


Рис. 1.2. Процесс распаковки архива

1.2. Настройка редактора Eclipse

Для компиляции и запуска примеров из книги мы будем пользоваться редактором Eclipse. Для загрузки редактора Eclipse переходим на страницу **http://www.eclipse.org/downloads/** и скачиваем архив с программой из раздела **Eclipse IDE for Java Developers**. Распаковываем архив в какую-либо папку (в моем случае программа установлена в папку C:\Android\Program) — в результате будет создана папка eclipse, внутри которой расположены файлы редактора. Редактор не нуждается в установке, поэтому просто запускаем файл eclipse.exe. При запуске редактор попросит указать папку с рабочим пространством. Указываем папку и нажимаем кнопку **OK**.

Для отображения результатов выполнения программ мы воспользуемся возможностями библиотеки JavaFX, поэтому дополнительно следует установить модуль e(fx)clipse. Информацию о его установке можно найти на странице **http://www.eclipse.org/efxclipse/install.html**. Прежде чем устанавливать модуль, убедитесь, что на компьютере установлена версия Java SE 8 (причем JDK, а не просто JRE). Если установлена более ранняя версия, то для отображения результатов нужно будет использовать библиотеку `Swing`.

Для создания *проекта* в меню **File** редактора Eclipse выбираем пункт **New | Java Project**. В открывшемся окне (рис. 1.3) в поле **Project name** вводим `OpenCV33`, выбираем версию JRE и нажимаем кнопку **Finish**.

Далее создадим *пакет*. Для этого в меню **File** выбираем пункт **New | Package**. В открывшемся окне (рис. 1.4) в поле **Name** вводим `application` и нажимаем кнопку **Finish**.

Теперь можно добавить в пакет *класс*. Для этого в меню **File** выбираем пункт **New | Class**. В открывшемся окне (рис. 1.5) в поле **Name** вводим `Main`, в поле **Package** — `application`, а затем нажимаем кнопку **Finish**.

Для удобного подключения библиотеки OpenCV к нескольким проектам в меню **Window** выбираем пункт **Preferences**. В открывшемся окне (рис. 1.6) переходим на вкладку **Java | Build Path | User Libraries** и нажимаем кнопку **New**. В открывшемся окне (рис. 1.7) в поле **User library name** вводим `opencv_330` и нажимаем кнопку **OK** — в список будет добавлен новый пункт. Выделяем его, нажимаем кнопку **Add External JARs**, выбираем файл C:\opencv_3_3\build\java\opencv-330.jar и нажимаем кнопку **Открыть** — в списке отобразится новый пункт с названием JAR-архива. Выделяем пункт **Native library location** и нажимаем кнопку **Edit**. В открывшемся окне (рис. 1.8) вводим путь к папке C:\opencv_3_3\build\java\x64 (содержит файл opencv_java330.dll) и нажимаем кнопку **OK**.

Если на вашем компьютере установлена 32-разрядная операционная система или JDK, то нужно будет указать путь к папке C:\opencv_3_3\build\java\x86. Точно таким же образом можно добавить поддержку и для версии 2.4 (JAR-архив opencv-2413.jar и путь C:\opencv_2_4\build\java\x64). Сохраняем изменения, нажимая кнопку **OK**.

Теперь необходимо подключить к проекту библиотеку. Для этого в окне **Package Explorer** щелкаем правой кнопкой мыши на названии проекта и из контекстного

меню выбираем пункт **Properties**. В открывшемся окне (рис. 1.9) переходим на вкладку **Java Build Path**, а затем справа выбираем вкладку **Libraries** и нажимаем кнопку **Add Library**. В открывшемся окне (рис. 1.10) выбираем пункт **User Library** и нажимаем кнопку **Next**. На следующем шаге (рис. 1.11) устанавливаем флажок **opencv_330** и нажимаем кнопку **Finish** — библиотека будет добавлена в список. Сохраняем изменения, нажимая кнопку **OK**. Точно таким же образом создайте проект с названием `OpenCV24` и подключите к нему библиотеку OpenCV версии 2.4.13.

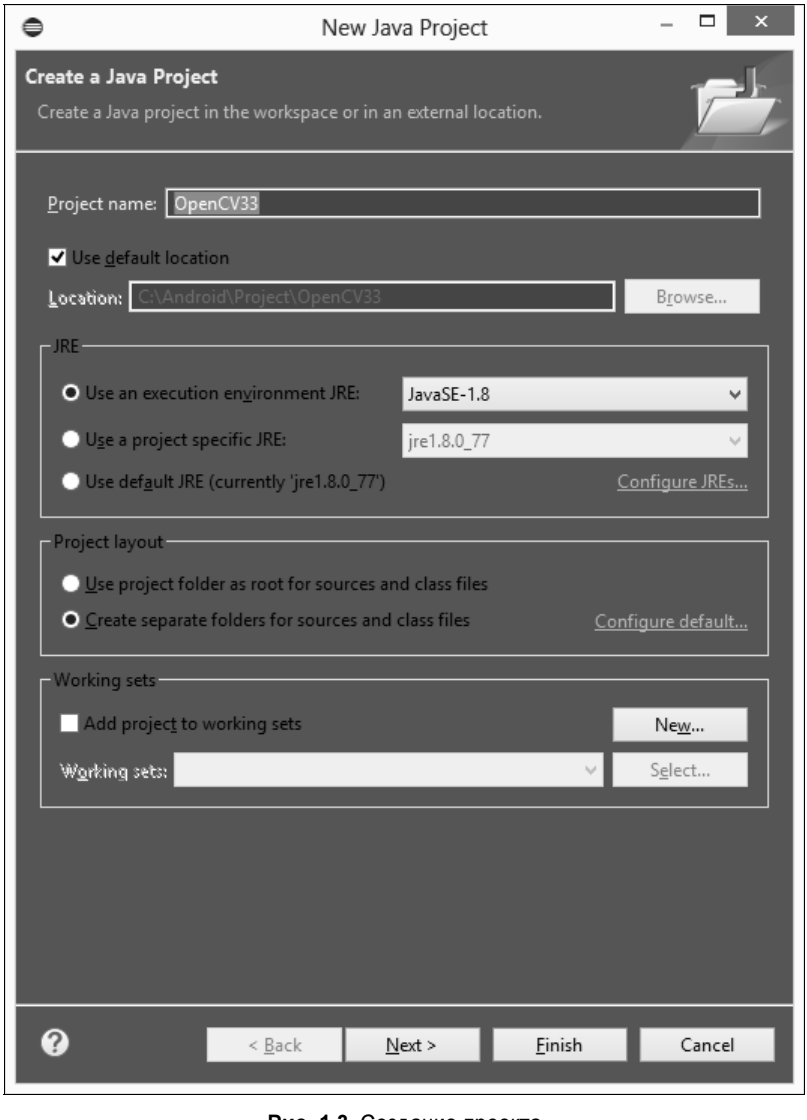


Рис. 1.3. Создание проекта

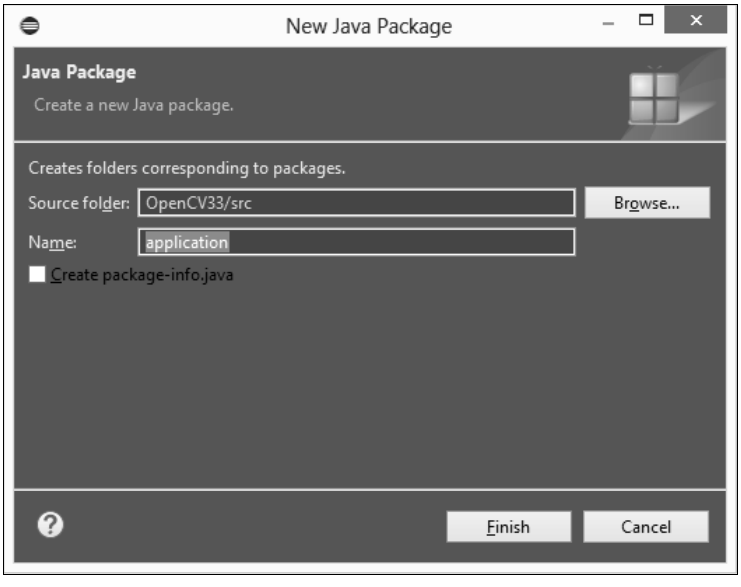


Рис. 1.4. Создание пакета

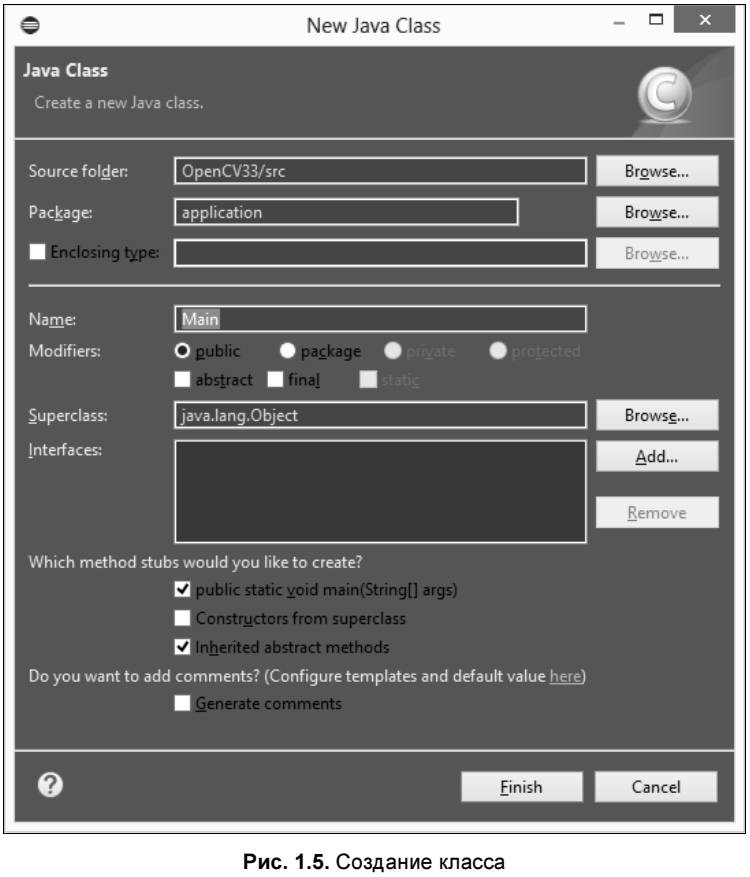


Рис. 1.5. Создание класса

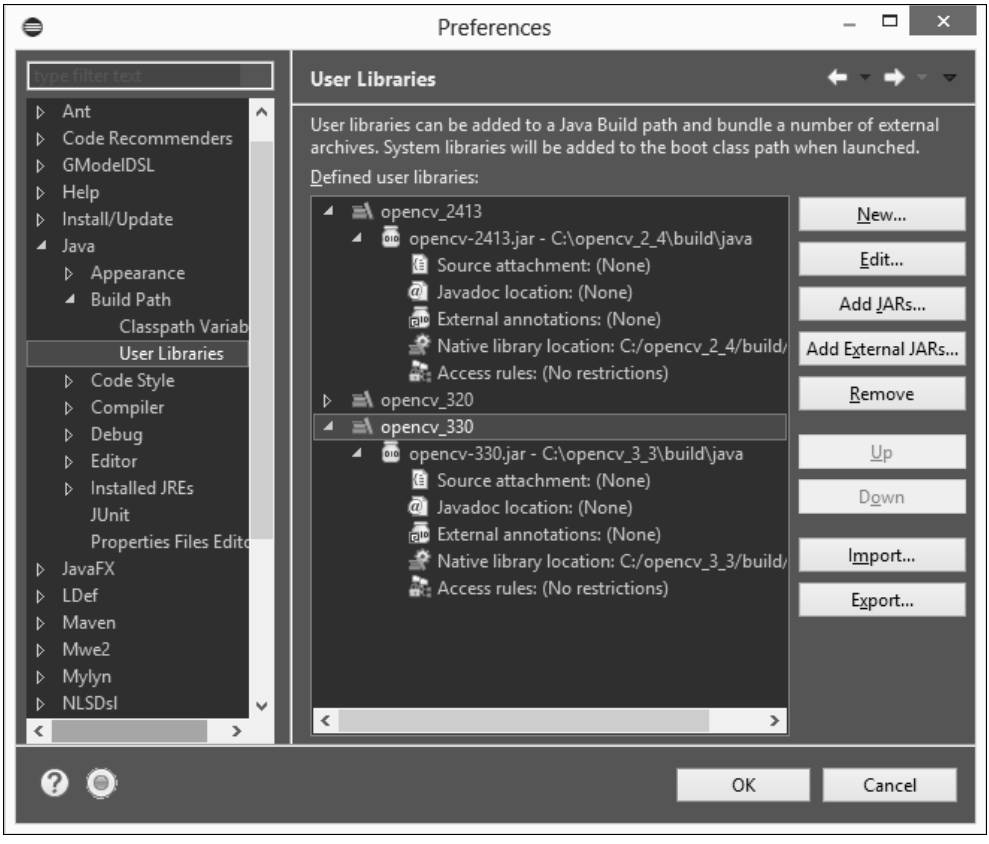


Рис. 1.6. Окно **Preferences**

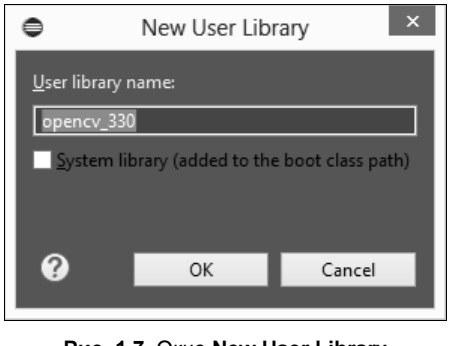


Рис. 1.7. Окно **New User Library**

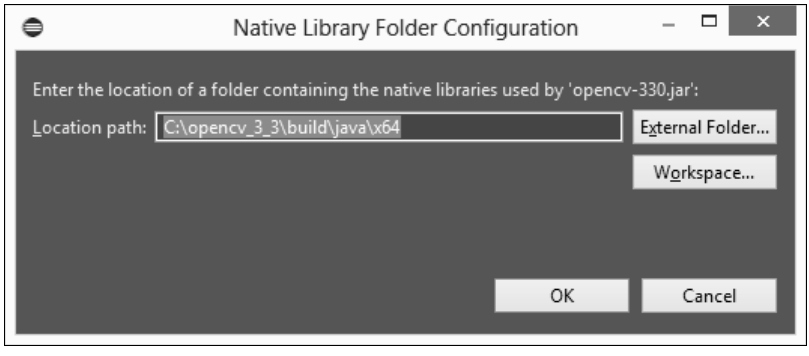


Рис. 1.8. Указание пути к DLL-файлу

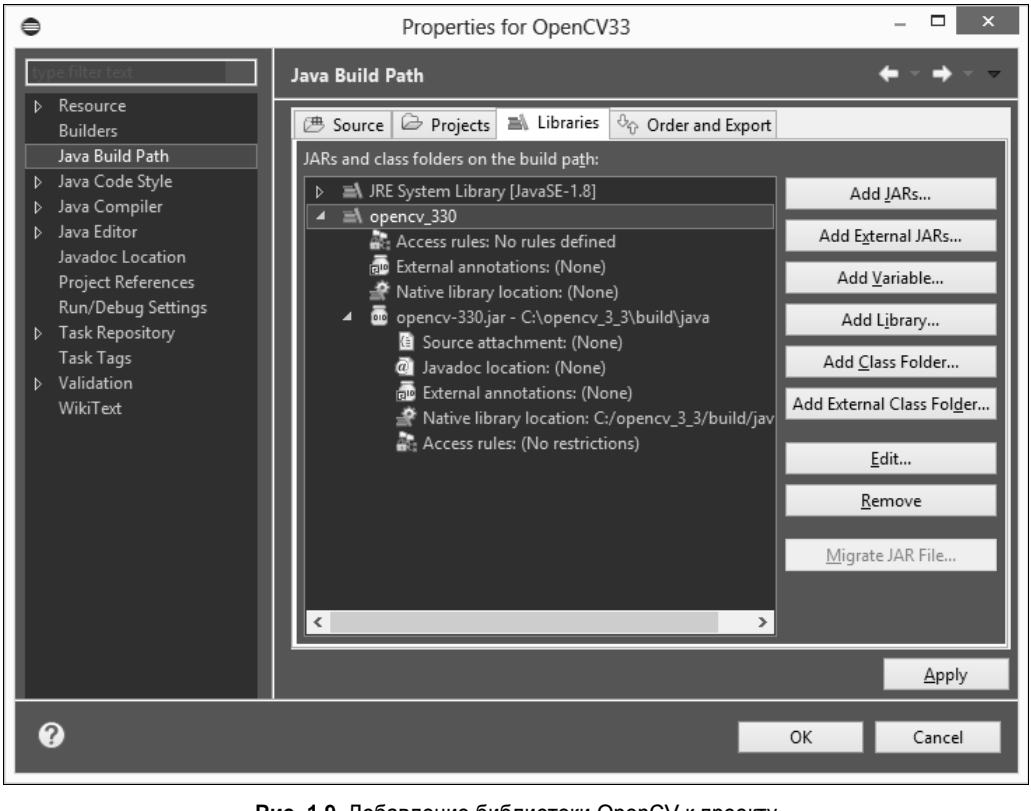


Рис. 1.9. Добавление библиотеки OpenCV к проекту

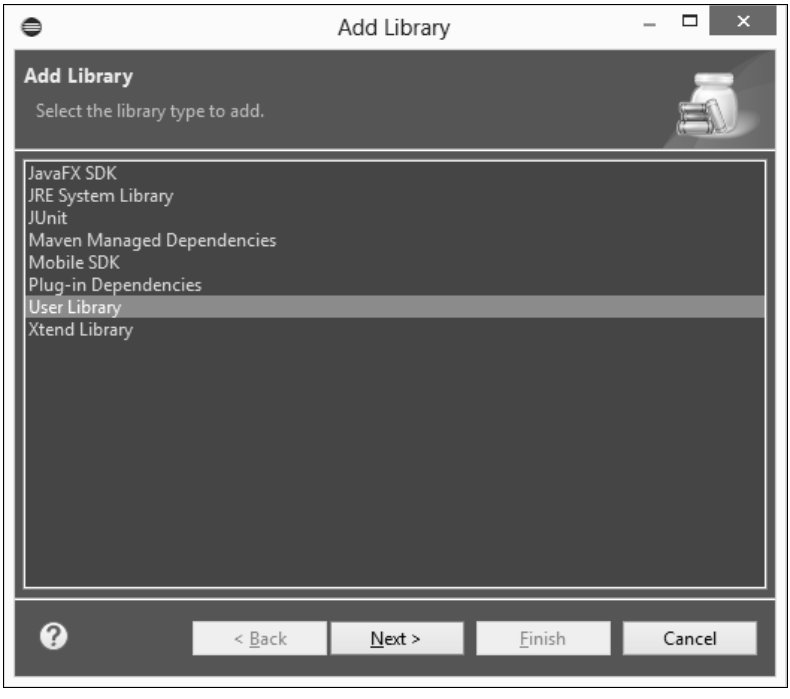


Рис. 1.10. Окно **Add Library**

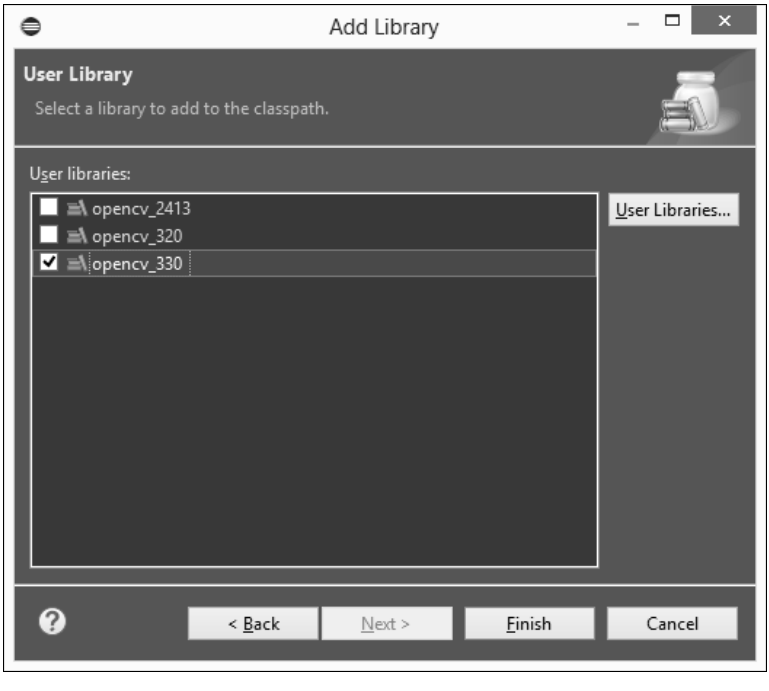


Рис. 1.11. Выбор нужной версии библиотеки OpenCV

Для проверки правильности установки создадим класс с названием `HelloCV` и внутри метода `main()` выведем версию библиотеки OpenCV, а также полную информацию о сборке (листинг 1.1).

Листинг 1.1. Вывод версии OpenCV

```
package application;

import org.opencv.core.Core;

public class HelloCV {
   static {
      System.loadLibrary(Core.NATIVE_LIBRARY_NAME);
   }

   public static void main(String[] args) {
      System.out.println(Core.VERSION);              // 3.3.0
      System.out.println(Core.VERSION_MAJOR);        // 3
      System.out.println(Core.VERSION_MINOR);        // 3
      System.out.println(Core.VERSION_REVISION);     // 0
      System.out.println(Core.NATIVE_LIBRARY_NAME); // opencv_java330
      System.out.println(Core.getBuildInformation());
   }
}
```

Для компиляции и запуска программы в меню **Run** выбираем пункт **Run**, или нажимаем комбинацию клавиш <Ctrl>+<F11>, или нажимаем кнопку с белым треугольником внутри зеленого круга на панели инструментов. Результат выполнения будет выведен в окне **Console**.

В OpenCV версии 2.4 получить информацию о версии можно так:

```
System.out.println(Core.VERSION);              // 2.4.13.0
System.out.println(Core.VERSION_EPOCH);        // 2
System.out.println(Core.VERSION_MAJOR);        // 4
System.out.println(Core.VERSION_MINOR);        // 13
System.out.println(Core.VERSION_REVISION);     // 0
```

Константа `VERSION_EPOCH` в OpenCV версии 3.3 отсутствует. Как видите, уже с первых строк кода видны различия в версиях. Причем версии несовместимы друг с другом, т. к. различий очень много.

> ***Примечание***
>
> Если при запуске кода из листинга 1.1 вы получили какую-либо ошибку, или в консоли не отобразилась версия OpenCV, то, скорее всего, на компьютере не хватает библиотек динамической компоновки. Сначала нужно посмотреть на название архива с OpenCV. В версии 3.3.0 архив называется *opencv-3.3.0-vc14.exe*. Фрагмент *vc14* говорит, что библиотеки были скомпилированы в Visual Studio 2015. Следовательно, нужно установить

либо Visual Studio (можно бесплатную версию Community), либо бесплатный пакет Visual C++ Redistributable for Visual Studio 2015, который содержит все необходимые библиотеки. После установки не забудьте перезагрузить компьютер.

1.3. Вспомогательные классы

Прежде чем мы приступим к изучению основных классов, необходимо рассмотреть несколько вспомогательных классов, с помощью которых указываются различные параметры, — например: координаты точки, размеры области, положение области в пространстве и т. д.

1.3.1. Класс *Point*: точка

Класс `Point` описывает координаты точки в двумерном пространстве. Инструкция импорта:

```
import org.opencv.core.Point;
```

Конструкторы класса:

```
Point()
Point(double x, double y)
Point(double[] vals)
```

Пример использования первого конструктора:

```
Point p = new Point();
System.out.println(p);               // {0.0, 0.0}
System.out.println(p.x + " " + p.y); // 0.0 0.0
p.x = 10.0;
p.y = 5.0;
System.out.println(p.x + " " + p.y); // 10.0 5.0
```

Пример использования второго конструктора:

```
Point p = new Point(10.0, 5.0);
System.out.println(p);               // {10.0, 5.0}
System.out.println(p.x + " " + p.y); // 10.0 5.0
```

Пример использования третьего конструктора:

```
Point p = new Point(new double[] {10.0, 5.0});
System.out.println(p);               // {10.0, 5.0}
System.out.println(p.x + " " + p.y); // 10.0 5.0
```

Перечислим основные методы класса `Point`:

- `set()` — задает новое значение. Формат метода:

  ```
  public void set(double[] vals)
  ```

  Пример:

  ```
  Point p = new Point(0.0, 0.0);
  System.out.println(p.x + " " + p.y); // 0.0 0.0
  ```

```
p.set(new double[] {10.0, 5.0});
System.out.println(p.x + " " + p.y); // 10.0 5.0
p.x = 20.0;
p.y = 3.0;
System.out.println(p.x + " " + p.y); // 20.0 3.0
```

- `equals()` — выполняет сравнение двух точек. Формат метода:

```
public boolean equals(Object obj)
```

Пример:

```
Point p = new Point(0.0, 0.0);
Point p2 = new Point(0.0, 0.0);
Point p3 = new Point(5.0, 3.0);
System.out.println(p.equals(p2)); // true
System.out.println(p.equals(p3)); // false
```

- `clone()` — создает копию объекта. Формат метода:

```
public Point clone()
```

Пример:

```
Point p = new Point(10.0, 5.0);
Point p2 = p;
p2.x = 30.0;
System.out.println(p.x + " " + p.y);   // 30.0 5.0
Point p3 = p.clone();
p3.x = 85.0;
System.out.println(p.x + " " + p.y);   // 30.0 5.0
System.out.println(p3.x + " " + p3.y); // 85.0 5.0
```

- `inside()` — возвращает `true`, если точка входит в указанную прямоугольную область. Формат метода:

```
import org.opencv.core.Rect;
public boolean inside(Rect r)
```

Пример:

```
Point p = new Point(9.0, 5.0);
Rect rect = new Rect(0, 0, 10, 10);
Rect rect2 = new Rect(10, 5, 10, 10);
System.out.println(p.inside(rect));  // true
System.out.println(p.inside(rect2)); // false
```

Прочие методы:

```
public double dot(Point p)
public String toString()
public int hashCode()
```

1.3.2. Класс *Point3*: 3D-точка

Класс `Point3` описывает координаты точки в трехмерном пространстве. Инструкция импорта:

```
import org.opencv.core.Point3;
```

Конструкторы класса:

```
Point3()
Point3(double x, double y, double z)
Point3(double[] vals)
Point3(Point p)
```

Пример использования первого конструктора:

```
Point3 p = new Point3();
System.out.println(p);                          // {0.0, 0.0, 0.0}
System.out.println(p.x + " " + p.y + " " + p.z); // 0.0 0.0 0.0
p.x = 20.0;
p.y = 3.0;
p.z = 12.0;
System.out.println(p.x + " " + p.y + " " + p.z); // 20.0 3.0 12.0
```

Пример использования второго конструктора:

```
Point3 p = new Point3(10.0, 20.0, 30.0);
System.out.println(p.x + " " + p.y + " " + p.z); // 10.0 20.0 30.0
```

Пример использования третьего конструктора:

```
Point3 p = new Point3(new double[] {10.0, 20.0, 30.0});
System.out.println(p.x + " " + p.y + " " + p.z); // 10.0 20.0 30.0
```

Пример использования четвертого конструктора:

```
Point3 p = new Point3(new Point(10.0, 20.0));
System.out.println(p.x + " " + p.y + " " + p.z); // 10.0 20.0 0.0
```

Перечислим основные методы класса `Point3`:

- `set()` — задает новое значение. Формат метода:

  ```
  public void set(double[] vals)
  ```

  Пример:

  ```
  Point3 p = new Point3();
  p.set(new double[] {10.0, 20.0, 30.0});
  System.out.println(p.x + " " + p.y + " " + p.z);
  // 10.0 20.0 30.0
  ```

- `equals()` — выполняет сравнение двух точек. Формат метода:

  ```
  public boolean equals(Object obj)
  ```

  Пример:

  ```
  Point3 p = new Point3(10.0, 20.0, 30.0);
  Point3 p2 = new Point3(10.0, 20.0, 30.0);
  ```

```
Point3 p3 = new Point3(10.0, 20.0, 0.0);
System.out.println(p.equals(p2)); // true
System.out.println(p.equals(p3)); // false
```

- `clone()` — создает копию. Формат метода:

```
public Point3 clone()
```

Пример:

```
Point3 p = new Point3(10.0, 20.0, 30.0);
Point3 p2 = p;
p2.x = 0.0;
System.out.println(p.x + " " + p.y + " " + p.z);
// 0.0 20.0 30.0
Point3 p3 = p.clone();
p3.x = 85.0;
System.out.println(p.x + " " + p.y + " " + p.z);
// 0.0 20.0 30.0
System.out.println(p3.x + " " + p3.y + " " + p3.z);
// 85.0 20.0 30.0
```

Прочие методы:

```
public Point3 cross(Point3 p)
public double dot(Point3 p)
public int hashCode()
public String toString()
```

1.3.3. Класс *Size*: размеры прямоугольной области

Класс `Size` описывает размеры прямоугольной области. Инструкция импорта:

```
import org.opencv.core.Size;
```

Конструкторы класса:

```
Size()
Size(double width, double height)
Size(double[] vals)
Size(Point p)
```

Пример использования первого конструктора:

```
Size s = new Size();
System.out.println(s); // 0x0
s.width = 100.0;
s.height = 50.0;
System.out.println(s.width + " " + s.height); // 100.0 50.0
```

Пример использования второго конструктора:

```
Size s = new Size(100.0, 50.0);
System.out.println(s.width + " " + s.height); // 100.0 50.0
```

Пример использования третьего конструктора:

```
Size s = new Size(new double[] {100.0, 50.0});
System.out.println(s.width + " " + s.height); // 100.0 50.0
```

Пример использования четвертого конструктора:

```
Size s = new Size(new Point(100.0, 50.0));
System.out.println(s.width + " " + s.height); // 100.0 50.0
```

Перечислим основные методы класса `Size`:

- `set()` — задает новое значение. Формат метода:

  ```
  public void set(double[] vals)
  ```

  Пример:

  ```
  Size s = new Size(100.0, 50.0);
  s.set(new double[] {10.0, 20.0});
  System.out.println(s.width + " " + s.height); // 10.0 20.0
  ```

- `area()` — возвращает площадь. Формат метода:

  ```
  public double area()
  ```

  Пример:

  ```
  Size s = new Size(100.0, 50.0);
  System.out.println(s.area());           // 5000.0
  System.out.println(s.width * s.height); // 5000.0
  ```

- `empty()` — возвращает значение `true`, если ширина или высота меньше или равна нулю. Формат метода:

  ```
  public boolean empty() // Только в версии 3.3
  ```

  Пример:

  ```
  Size s = new Size(100.0, 50.0);
  System.out.println(s.empty());           // false
  Size s2 = new Size(100.0, 0.0);
  System.out.println(s2.empty());          // true
  ```

Прочие методы:

```
public Size clone()
public boolean equals(Object obj)
public int hashCode()
public String toString()
```

1.3.4. Класс *Rect*: прямоугольная область

Класс `Rect` описывает координаты и размеры прямоугольной области. Инструкция импорта:

```
import org.opencv.core.Rect;
```

Конструкторы класса:

```
Rect()
Rect(int x, int y, int width, int height)
Rect(double[] vals)
Rect(Point p1, Point p2)
Rect(Point p, Size s)
```

Пример использования первого конструктора:

```
Rect rect = new Rect();
System.out.println(rect); // {0, 0, 0x0}
rect.x = 0;
rect.y = 10;
rect.width = 100;
rect.height = 50;
System.out.println(rect.x + " " + rect.y);          // 0 10
System.out.println(rect.width + " " + rect.height); // 100 50
```

Пример использования второго конструктора:

```
Rect rect = new Rect(0, 0, 100, 50);
System.out.println(rect.x + " " + rect.y);          // 0 0
System.out.println(rect.width + " " + rect.height); // 100 50
```

Пример использования третьего конструктора:

```
Rect rect = new Rect(new double[] {0.0, 0.0, 100.0, 50.0});
System.out.println(rect.x + " " + rect.y);          // 0 0
System.out.println(rect.width + " " + rect.height); // 100 50
```

Пример использования четвертого конструктора:

```
Rect rect = new Rect(new Point(0.0, 0.0), new Point(100.0, 50.0));
System.out.println(rect.x + " " + rect.y);          // 0 0
System.out.println(rect.width + " " + rect.height); // 100 50
```

Пример использования пятого конструктора:

```
Rect rect = new Rect(new Point(0.0, 0.0), new Size(100.0, 50.0));
System.out.println(rect.x + " " + rect.y);          // 0 0
System.out.println(rect.width + " " + rect.height); // 100 50
```

Перечислим основные методы класса `Rect`:

- `set()` — задает новое значение. Формат метода:

  ```
  public void set(double[] vals)
  ```

  Пример:

  ```
  Rect rect = new Rect(10, 5, 100, 50);
  rect.set(new double[] {0.0, 0.0, 100.0, 50.0});
  System.out.println(rect.x + " " + rect.y);          // 0 0
  System.out.println(rect.width + " " + rect.height); // 100 50
  ```

- `size()` — возвращает размеры области. Формат метода:

```
public Size size()
```

Пример:

```
Rect rect = new Rect(10, 5, 100, 50);
Size s = rect.size();
System.out.println(s);          // 100x50
```

- `tl()` — возвращает координаты левого верхнего угла прямоугольной области. Формат метода:

```
public Point tl()
```

- `br()` — возвращает координаты правого нижнего угла прямоугольной области. Формат метода:

```
public Point br()
```

Пример:

```
Rect rect = new Rect(10, 5, 100, 50);
Point topLeft = rect.tl();
Point bottomRight = rect.br();
System.out.println(topLeft);          // {10.0, 5.0}
System.out.println(bottomRight);      // {110.0, 55.0}
```

- `area()` — возвращает площадь прямоугольника. Формат метода:

```
public double area()
```

Пример:

```
Rect rect = new Rect(10, 5, 100, 50);
System.out.println(rect.area());          // 5000.0
```

- `contains()` — возвращает `true`, если указанная точка входит в прямоугольную область. Формат метода:

```
public boolean contains(Point p)
```

Пример:

```
Rect rect = new Rect(10, 5, 100, 50);
System.out.println(rect.contains(new Point(10.0, 5.0))); // true
System.out.println(rect.contains(new Point(9.0, 5.0)));  // false
```

- `empty()` — возвращает значение `true`, если ширина или высота меньше или равна нулю. Формат метода:

```
public boolean empty() // Только в версии 3.3
```

Пример:

```
Rect rect = new Rect(10, 5, 100, 50);
System.out.println(rect.empty());     // false
Rect rect2 = new Rect(10, 5, 100, 0);
System.out.println(rect2.empty());    // true
```

Прочие методы:

```
public Rect clone()
public boolean equals(Object obj)
public int hashCode()
public String toString()
```

В версии 3.3 доступен также класс `Rect2d`, который имеет точно такие же конструкторы и методы, как и класс `Rect`. Различие состоит в типе данных полей класса: в классе `Rect` поля `x`, `y`, `width` и `height` имеют тип `int`, а в классе `Rect2d` — тип `double`. Инструкция импорта:

```
import org.opencv.core.Rect2d;
```

Пример:

```
Rect2d rect = new Rect2d(10, 5, 100, 50);
System.out.println(rect);        // {10.0, 5.0, 100.0x50.0}
rect.set(new double[] {0.0, 0.0, 100.0, 50.0});
System.out.println(rect.size()); // 100x50
System.out.println(rect.tl());   // {0.0, 0.0}
```

1.3.5. Класс *RotatedRect*: повернутая прямоугольная область

Класс `RotatedRect` описывает прямоугольную область, которая повернута на некоторый угол. Инструкция импорта:

```
import org.opencv.core.RotatedRect;
```

Конструкторы класса:

```
RotatedRect()
RotatedRect(Point center, Size size, double angle)
RotatedRect(double[] vals)
```

Параметр `center` задает координаты центра прямоугольника, параметр `size` — размеры прямоугольника, а параметр `angle` — угол, на который повернут прямоугольник.

Пример использования первого конструктора:

```
RotatedRect rect = new RotatedRect();
System.out.println(rect); // { {0.0, 0.0} 0x0 * 0.0 }
rect.center = new Point(50.0, 50.0);
rect.size = new Size(100.0, 100.0);
rect.angle = 45.0;
System.out.println(rect); // { {50.0, 50.0} 100x100 * 45.0 }
```

Пример использования второго конструктора:

```
RotatedRect rect = new RotatedRect(new Point(50.0, 50.0),
                       new Size(100.0, 100.0), 45.0);
System.out.println(rect); // { {50.0, 50.0} 100x100 * 45.0 }
```

Пример использования третьего конструктора:

```
RotatedRect rect = new RotatedRect(new double[] {50.0, 50.0,
                                                 100.0, 100.0, 45.0});
System.out.println(rect);        // { {50.0, 50.0} 100x100 * 45.0 }
System.out.println(rect.center); // {50.0, 50.0}
System.out.println(rect.size);   // 100x100
System.out.println(rect.angle);  // 45.0
```

Перечислим основные методы класса `RotatedRect`:

- `set()` — задает новое значение. Формат метода:

  ```
  public void set(double[] vals)
  ```

  Пример:

  ```
  RotatedRect rect = new RotatedRect();
  rect.set(new double[] {50.0, 50.0, 100.0, 100.0, 45.0});
  System.out.println(rect);  // { {50.0, 50.0} 100x100 * 45.0 }
  ```

- `boundingRect()` — возвращает прямоугольную область, в которую вписан повернутый прямоугольник. Формат метода:

  ```
  public Rect boundingRect()
  ```

  Пример:

  ```
  RotatedRect rect = new RotatedRect(new Point(50.0, 50.0),
                         new Size(100.0, 100.0), 45.0);
  System.out.println(rect.boundingRect());
  // {-21, -21, 143x143}
  ```

- `points()` — записывает в массив координаты вершин прямоугольника. Формат метода:

  ```
  public void points(Point pt[])
  ```

  Пример:

  ```
  RotatedRect rect = new RotatedRect(new Point(50.0, 50.0),
                         new Size(100.0, 100.0), 45.0);
  Point[] pt = new Point[4];
  rect.points(pt);
  for (int i = 0; i < pt.length; i++) {
     System.out.println(pt[i]);
  }
  /*
  {-20.710678118654748, 50.00000000000001}
  {50.0, -20.710678118654748}
  {120.71067811865476, 49.99999999999999}
  {50.0, 120.71067811865476}*/
  ```

Прочие методы:

```
public RotatedRect clone()
public boolean equals(Object obj)
public int hashCode()
public String toString()
```

1.3.6. Класс *Scalar*: объект из четырех элементов

Класс `Scalar` описывает объект из четырех элементов. Используется в основном для указания скалярного значения, а также для описания значений компонентов цвета. Инструкция импорта:

```
import org.opencv.core.Scalar;
```

Конструкторы класса:

```
Scalar(double v0)
Scalar(double v0, double v1)
Scalar(double v0, double v1, double v2)
Scalar(double v0, double v1, double v2, double v3)
Scalar(double[] vals)
```

Пример:

```
Scalar s = new Scalar(1.0);
System.out.println(s); // [1.0, 0.0, 0.0, 0.0]
s = new Scalar(1.0, 2.0);
System.out.println(s); // [1.0, 2.0, 0.0, 0.0]
s = new Scalar(1.0, 2.0, 3.0);
System.out.println(s); // [1.0, 2.0, 3.0, 0.0]
s = new Scalar(1.0, 2.0, 3.0, 4.0);
System.out.println(s); // [1.0, 2.0, 3.0, 4.0]
s = new Scalar(new double[] {1.0, 2.0, 3.0, 4.0});
System.out.println(s); // [1.0, 2.0, 3.0, 4.0]
System.out.println(s.val[0]); // 1.0
System.out.println(s.val[1]); // 2.0
System.out.println(s.val[2]); // 3.0
System.out.println(s.val[3]); // 4.0
s.val[0] = 10.0;
s.val[1] = 20.0;
s.val[2] = 30.0;
s.val[3] = 40.0;
System.out.println(s); // [10.0, 20.0, 30.0, 40.0]
```

Для создания объекта с одинаковыми значениями всех компонентов можно воспользоваться статическим методом `all()`. Формат метода:

```
public static Scalar all(double v)
```

Пример:

```
Scalar s = Scalar.all(1.0);
System.out.println(s); // [1.0, 1.0, 1.0, 1.0]
```

Перечислим основные методы класса `Scalar`:

- `set()` — задает новое значение. Формат метода:

```
public void set(double[] vals)
```

Пример:

```
Scalar s = new Scalar(0.0);
s.set(new double[] {1.0, 2.0, 3.0, 4.0});
System.out.println(s); // [1.0, 2.0, 3.0, 4.0]
s.set(new double[] {10.0});
System.out.println(s); // [10.0, 0.0, 0.0, 0.0]
```

- `isReal()` — возвращает значение `true`, если второй, третий и четвертый компоненты равны `0.0`. Формат метода:

```
public boolean isReal()
```

Пример:

```
Scalar s = Scalar.all(1.0);
System.out.println(s.isReal()); // false
s = Scalar.all(0.0);
System.out.println(s.isReal()); // true
s = new Scalar(10, 0, 0, 0);
System.out.println(s.isReal()); // true
```

- `conj()` — возвращает значение `(v0, -v1, -v2, -v3)`. Формат метода:

```
public Scalar conj()
```

Пример:

```
Scalar s = new Scalar(1.0, 2.0, 3.0, 4.0);
System.out.println(s.conj()); // [1.0, -2.0, -3.0, -4.0]
```

- `mul()` — перемножает значения компонентов (которые расположены на одинаковых позициях) двух объектов и возвращает новый объект. Параметр `scale` задает дополнительный коэффициент масштаба (в первом формате параметр равен `1`). Форматы метода:

```
public Scalar mul(Scalar it)
public Scalar mul(Scalar it, double scale)
```

Пример:

```
Scalar s = new Scalar(1.0, 2.0, 3.0, 4.0);
System.out.println(s.mul(Scalar.all(2.0)));
// [2.0, 4.0, 6.0, 8.0]
s = new Scalar(1.0, 2.0, 3.0, 4.0);
System.out.println(s.mul(Scalar.all(2.0), 2));
// [4.0, 8.0, 12.0, 16.0]
```

Прочие методы:

```
public Scalar clone()
public boolean equals(Object obj)
```

```
public int hashCode()
public String toString()
```

1.3.7. Класс *Range*: диапазон

Класс `Range` описывает диапазон значений. Инструкция импорта:

```
import org.opencv.core.Range;
```

Конструкторы класса:

```
Range()
Range(int start, int end)
Range(double[] vals)
```

Параметр `start` задает начало диапазона (включая элемент с этим значением), а параметр `end` — конец диапазона (не включая элемент с этим значением).

Пример использования первого конструктора:

```
Range r = new Range();
r.start = 1;
r.end = 51;
System.out.println(r);       // [1, 51)
```

Пример использования второго конструктора:

```
Range r = new Range(1, 51);
System.out.println(r);       // [1, 51)
System.out.println(r.start); // 1
System.out.println(r.end);   // 51
```

Пример использования третьего конструктора:

```
Range r = new Range(new double[] {1, 51});
System.out.println(r);        // [1, 51)
```

С помощью статического метода `all()` можно создать диапазон всех возможных значений. Формат метода:

```
public static Range all()
```

Пример:

```
Range r = Range.all();
System.out.println(r);        // [-2147483648, 2147483647)
System.out.println(r.size()); // -1
```

Перечислим основные методы класса `Range`:

- `set()` — задает новые значения. Формат метода:

  ```
  public void set(double[] vals)
  ```

  Пример:

  ```
  Range r = new Range();
  r.set(new double[] {1, 51});
  System.out.println(r);        // [1, 51)
  ```

- `size()` — возвращает количество элементов, входящих в диапазон. Формат метода:

```
public int size()
```

Пример:

```
Range r = new Range(1, 51);
System.out.println(r.size()); // 50
```

- `empty()` — возвращает значение `true`, если диапазон не содержит элементов. Формат метода:

```
public boolean empty()
```

Пример:

```
Range r = new Range();
System.out.println(r);         // [0, 0)
System.out.println(r.size());  // 0
System.out.println(r.empty()); // true
```

- `shift()` — сдвигает границы диапазона на указанное значение. Формат метода:

```
public Range shift(int delta)
```

Пример:

```
Range r = new Range(1, 51);
System.out.println(r.shift(10));         // [11, 61)
```

- `intersection()` — возвращает пересечение двух диапазонов. Формат метода:

```
public Range intersection(Range r1)
```

Пример:

```
Range r = new Range(1, 51);
Range r2 = new Range(30, 81);
System.out.println(r.intersection(r2));  // [30, 51)
```

Прочие методы:

```
public Range clone()
public boolean equals(Object obj)
public int hashCode()
public String toString()
```