Изображение выглядит как текст

Автоматически созданное описание

СОДЕРЖАНИЕ

[Ведение 4](#_Toc91533914)

[1 Язык программирования Java 6](#_Toc91533915)

[1.1 Причины выбрать Java для изучения 6](#_Toc91533916)

[1.2 Направления, где используется Java 8](#_Toc91533917)

[2 Сравнение между IntelliJ Idea и Eclipse 10](#_Toc91533918)

[2.1 IntelliJ Idea vs Eclipse 10](#_Toc91533919)

[2.2 Что такое IntelliJ Idea? 10](#_Toc91533920)

[2.3 Что такое Eclipse IDE? 11](#_Toc91533921)

[2.4 Основные различия между IntelliJ Idea и Eclipse IDE 12](#_Toc91533922)

[3 Темы для видео-уроков 15](#_Toc91533923)

[3.1 Программа Hello World 15](#_Toc91533924)

[3.2 Объявление переменных 17](#_Toc91533925)

[3.2.1 Назначение переменных 19](#_Toc91533926)

[3.2.2 Печать переменных 20](#_Toc91533928)

[3.2.3 Арифметические операторы 21](#_Toc91533929)

[3.2.4 Числа с плавающей точкой 23](#_Toc91533930)

[3.2.5 Оператор String 26](#_Toc91533931)

[3.3 Методы 27](#_Toc91533932)

[3.3.1 Определение новых методов 27](#_Toc91533933)

[3.3.2 Параметры и аргументы 29](#_Toc91533934)

[3.3.3 Множественные параметры 32](#_Toc91533935)

[3.3.4 Возвращаемые значения 34](#_Toc91533936)

[3.4 Цепочки и вложенность 36](#_Toc91533937)

[3.4.1 Реляционные операторы 37](#_Toc91533938)

[3.4.2 Выражение if-else 37](#_Toc91533939)

[3.4.3 Цепочку и вложенность 41](#_Toc91533940)

[3.4.4 Заявление-переключатель 42](#_Toc91533941)

[3.5 Массивы и ссылки 44](#_Toc91533942)

[3.5.1 Создание массивов 45](#_Toc91533943)

[3.5.2 Доступ к элементам 47](#_Toc91533944)

[3.5.3 Отображение массивов 49](#_Toc91533945)

[4 Среда создания видео VEGAS Pro 52](#_Toc91533946)

[Заключение 55](#_Toc91533947)

[Список используемой литературы 57](#_Toc91533948)

# **ВЕДЕНИЕ**

Основной закон об образовании гласит: "Под дистанционными образовательными технологиями понимаются образовательные технологии, реализуемые преимущественно с использованием информационных и

телекоммуникационных сетей при опосредованном (на расстоянии)

взаимодействии обучающихся и преподавателей".

Дистанционное обучение – это взаимодействие обучающихся и преподавателей на расстоянии, показывающие компоненты, присущие процессу обучения (цели, содержание, методы, алгоритмы, средства обучения) и реализуемое с помощью специальных интернет-технологий или других средств, обеспечивающих интерактивность.

Актуальность дистанционного обучения очевидна: русский педагог Сеидов объясняет растущую популярность этой формы обучения тем, что в России только 51% студентов университетов моложе 24 лет и лишь четверть в возрасте 18–22 лет. Остальные – взрослые люди и заняты семейными и рабочими делами. Очная форма обучения в университете не всегда удобна для них. Дистанционное образование также является экономически эффективным, если принять во внимание транспортные и организационные расходы очной системы. По прогнозам ученых США ученики средней и старшей школы, студенты университетов будут проводить на очных занятиях лишь 20–30% своего времени, 70% будет приходиться на дистанционное обучение, а остальное – на самообразование. Отсюда понятен растущий интерес к дистанционному обучению – в университете, в школе, в системе дополнительного образования, т. е. на всех уровнях обучения в течение всей жизни.

Можно говорить, что дистанционное обучение открывает новые возможности, значительно расширяя как информационное пространство, так и информационное поле образования.

Средством, которое помогает эффективности дистанционного обучения, могут стать видео-уроки по программированию на языке Java.

1. **Язык программирования Java**

Java – один из самых распространенных и надежных языков для разработки приложений и программного обеспечения. Он имеет широкий спектр различных инструментов, которые могут значительно облегчить этот процесс [1].

Java – это объектно-ориентированный язык программирования. Его можно использовать для написания программного обеспечения на стороне сервера, программ для персональных компьютеров и мобильных приложений для различных целей и задач. Давайте рассмотрим причины, по которым вы должны выбрать Java в качестве языка программирования.

**1.1 Причины выбрать Java для изучения**

Рассмотрим причины выбрать Java для изучения:

1) Сильное Java-сообщество – по данным Java, в мире насчитывается 5 миллионов студентов и 12 миллионов разработчиков языка – от авторов библиотек до целых компаний. Здесь даже новичок может легко найти готовые куски кода для решения своих задач или ответы на любые вопросы.

Сообщество организовано и активно участвует в жизни языка, развивает его, разрабатывает множество библиотек и инструментов, в том числе для разработки серверной части. Благодаря сообществу экосистема Java продолжает развиваться и легко адаптируется к современным требованиям;

2) Разнообразие инструментов – Java имеет большое количество библиотек для решения общих задач, API, сред разработки и фреймворков для создания программ и приложений различной степени сложности. Возможности этого языка постоянно обновляются;

Множество инструментов Java для разработки на стороне сервера означает, что программы могут быть написаны быстрее, с меньшим количеством потенциальных ошибок и проблем, и быстрее выходят на рынок. И эта особенность языка делает его привлекательным для корпоративных приложений.

3) Виртуальная машина Java – это не только язык программирования, но и среда выполнения. Она позволяет писать достаточно высокопроизводительное программное обеспечение. Многие языки программирования используют Java runtime в качестве среды выполнения. Например, Groovy, Clojure, Scala, Kotlin, а также есть версии Python, JRuby, работающие на JVM. Создатели этих языков выбрали виртуальную машину Java из-за ее производительности, кроссплатформенности и возможности использовать множество библиотек и инструментов Java;

4) Кроссплатформенность Java – является кроссплатформенным языком. Недаром один из главных лозунгов этого языка до сих пор – Write once, run anywhere. Он позволяет разработчику создавать переносимый код, то есть написать программу один раз и без изменений выполнять ее в нескольких операционных системах (Windows, Linux и MacOS). JVM позволяет языку достичь такой гибкости;

5) Обратная совместимость – стремление Java к обратной совместимости, с одной стороны, тормозит его рост и развитие, но с другой – является неоспоримым преимуществом, особенно для корпоративных приложений. Эта характеристика дает ощущение стабильности и уверенности в том, что с каждой новой версией языка приложение будет работать;

6) Многочисленные способы изучения Java – сегодня существует множество инструментов для изучения языка. На рынке представлено огромное количество соответствующего материала: книги, статьи, семинары, конференции, курсы;

Но часто бывает так, что специалисту просто не хватает времени или понимания, чтобы сделать первый шаг в той или иной области. Получить базу по определенной теме или вектор движения можно на специализированных курсах и тренингах. Это будет полезно не только для профессионалов, которые хотят изучить возможности новой версии, API или библиотеки. Для начинающих разработчиков, которые только входят в профессию и хотят быть востребованными на рынке и построить успешную карьеру, такой способ получения знаний будет более предпочтительным.

7) Возможность быть востребованным на рынке – роль автоматизации в бизнесе растет. В то же время на рынке труда ощущается нехватка рабочей силы. Согласно отчету CompTIA, в декабре 2020 года наем ИТ-специалистов в США увеличился на 391 000 человек, при этом наибольшая доля новых предложений пришлась на разработчиков программного обеспечения и приложений – 62 900 человек;

Согласно индексу сообщества программистов TIOBE, в июле 2021 года Java займет второе место, уступая только языку C. Напомню, что этот индекс широко используется для проверки актуальности ваших навыков программирования или для помощи в принятии стратегического решения.

## **Направление, где используется Java**

Java – это язык, который используется для серверных приложений в крупных корпорациях. Особенно часто Java используется в банках, страховых компаниях, розничных сетях т. д. Например, такие банки, как Deutsche Bank, Citigroup, Barclays, Goldman Sachs и многие-многие другие используют Java для написания бэк-энд и фронт-энд офисных электронных систем и т. д. Еще пример: серверная часть Google+ написана на Java. Примеры типов приложений , написанных на языке Java:

1) вэб-приложения – широкое применение Java нашла в электронной коммерции и в вэб-приложениях. Например, если говорить о рынке Европы, США, многие вэб-приложения госучреждений, страховых, образовательных, оборонительных учреждений написаны на Java;

2) трейдинговые приложения – например, LMAX, Murex;

3) Android приложения – если у Вас телефон на Android, то все приложения в нём написаны на Java с использованием Google и Android API;

4) Десктопные приложения, программные средства и средства разработки – например, Eclipse, Netbeans IDE, jEdit (Programmer`s Text Editor), jDownloader (open-source download management tool), Azureus/Vuse;

5) Встраиваемые системы – Java используется на смарт-картах и сенсорах. Например, в банковских карточках используется Java;

Java все еще остается прогрессивным и востребованным языком программирования. Он универсален, его несложно освоить и, самое главное, для этого созданы все условия [1].

**2 Сравнение между IntelliJ Idea и Eclipse**

В течение последних многих лет Java является одним из ведущих языков программирования, используемых для различных целей. Большинство программистов для разработки надежных и безопасных настольных приложений используют Java. Большие предприятия и организации используют Java в качестве основного языка.

Для того чтобы писать коды на Java, необходима интегрированная среда разработки (IDE). Это программное обеспечение, в котором установлены различные пакеты и библиотеки, используемые для написания кода на различных языках. IntelliJ Idea и Eclipse – два таких программных обеспечения, которые активно используются для создания приложений, чтобы объединить инструменты разработчика с единым графическим интерфейсом пользователя.

## **2.1 IntelliJ Idea vs Eclipse**

Разница между IntelliJ Idea и Eclipse заключается в том, что Eclipse способен обрабатывать большие проекты быстрее, чем IntelliJ. С другой стороны, IntelliJ работает быстрее, чем Eclipse, в случае существующих проектов.

**2.2 Что такое IntelliJ Idea?**

IntelliJ IDEA для Java-разработчиков, выпущенная в 2001 году, является самой мощной, распространенной и полнофункциональной IDE. Она разработана, управляется и поддерживается компанией Jet Brains. Лицензия IntelliJ IDEA – Apache 2.0.

IntelliJ можно установить и использовать на Windows, Linux и macOS со следующими аппаратными требованиями:

1) минимум 2 ГБ ОЗУ, рекомендуется 4 ГБ ОЗУ;

2) минимум 1,5 ГБ дискового пространства + минимум 1 МБ для кэша;

3) разрешение экрана: 1024X768;

IntelliJ IDEA поставляется в двух редакциях:

1) Community Edition – эта версия доступна бесплатно и, в основном, используется разработчиками Java и Android. Поддерживается большинство языков, таких как Java, Clojure, Scala, Kotlin, Groovy и другие. Она содержит такие функции, как глубокий статический анализ, завершение кода, бегунок тестов, интеллектуальный рефакторинг, отладчик и т. д;

2) Ultimate Edition – эта версия включает в себя самую сложную коллекцию функций для разработки веб-приложений и настольных приложений. Она поощряет интеграцию фреймворка Spring и таких фреймворков веб-разработки, как Node.js, React и Angular;

**2.3 Что такое Eclipse IDE?**

Eclipse – это IDE с открытым исходным кодом для разработки приложений с использованием Java, Ruby, Python, C, C++ и др. Она была опубликована под лицензией Eclipse Public License (EPL) компанией IBM в 2001 году. Вскоре она стала известной для создания бесплатных и коммерческих проектов. На сегодняшний день данная программа самая распространенная Java IDE. Она предоставляет базовое рабочее пространство и расширяемую структуру плагинов для настройки среды.

Для запуска Eclipse IDE можно использовать любую платформу, такую как Windows, Linux, macOS и Solaris. Система должна поддерживать JVM со следующим оборудованием:

1) минимум 0,5 ГБ оперативной памяти, рекомендуется минимум 1 ГБ;

2) минимум 300 МБ места на жестком диске, рекомендуется минимум 1 ГБ;

3) минимальная скорость процессора: 800 МГц, рекомендуется минимум 1,5 ГГц;

**2.4 Основные различия между IntelliJ Idea и Eclipse IDE**

Иногда пользователь хочет оценить только выражение, чтобы увидеть его значение, что помогает отладить код. В IntelliJ Idea не нужно выделять все выражение, достаточно поместить курсор внутрь выражения и нажать Alt+F8, а в Eclipse нужно выделить все выражение, чтобы оно было оценено.

IntelliJ понимает, какой метод вам нужен, и дает возможность выбрать его, в то время, как в Eclipse – автозавершение работает не так гладко. Иногда он выбрасывает весь список элементов, которые начинаются с введенного символа, что, в большинстве случаев, бесполезно.

IntelliJ Idea предлагает только около 750 плагинов, в то время, как Eclipse предоставляет 1250+ плагинов с большим количеством плагинов для новых технологий.

Если говорить о производительности, то Eclipse намного быстрее работает с большими проектами по сравнению с IntelliJ, благодаря своей технике индексирования всего проекта на начальной стадии. Однако, IntelliJ Idea намного быстрее, когда пользователь работает над существующим проектом.

IntelliJ имеет более быструю кривую обучения и гораздо более простой в использовании интерфейс по сравнению с Eclipse. Eclipse предлагает завершение кода, быстрый просмотр, выпадающие окна и многие другие возможности, но IntelliJ является явным победителем по удобству использования.

В [таблице](#Таблице) 1 показаны удобно все недостатки и плюсы каждой IDE.

Таблица 1 – Сравнительная таблица между IntelliJ Idea и Eclipse

|  |  |  |
| --- | --- | --- |
| Параметры для сравнения | IntelliJ **Idea** | **Eclipse** |
| Отладка | Не нужно выделять все выражение с помощью IntelliJ Idea. Достаточно установить курсор внутри выражения в любой позиции и нажать Alt+F8 | В случае отладки в Eclipse для оценки некоторого выражения, необходимо выделить все выражение целиком |
| Автозаполнение | IntelliJ мгновенно может предложить единственные допустимые варианты для метода или функции и, следовательно, обладает потрясающей функцией автозавершения | Eclipse не может быстро понять действующий метод или контекст. Он показывает все, что начинается с введенного символа, что, в большинстве случаев, бесполезно. |
| Плагины | IntelliJ Idea предлагает около 750 плагинов, что меньше, чем Eclipse | 1250 плагинов предоставляются Eclipse Marketplace, и в нем есть много плагинов для новых технологий |
| Производительность | IntelliJ работает гораздо быстрее, когда речь идет о существующем проекте, по сравнению с Eclipse | Eclipse способен обрабатывать большие и тяжелые проекты быстрее, чем IntelliJ Idea, благодаря индексации, которую он выполняет на начальной стадии проекта |
| Удобный для пользователя | IntelliJ имеет простой, быстрый вид и удобен в использовании. Он имеет гораздо более приятную среду разработки, чем Eclipse. | Eclipse также предоставляет множество функций, делающих интерфейс удобным, но их недостаточно, чтобы превзойти IntelliJ. |

На эту тему существует множество аргументов, из-за которых трудно определить, какой из двух продуктов является наиболее актуальным. В результате большинство разработчиков утверждают, что обе IDE сопоставимы по своим достоинствам, и выбор любой из них – дело вкуса.

Но очень важно провести сравнение различных параметров между ними, и поэтому, если вы хотите получить бесплатный сервис с достойным набором функций и инструментов, вам следует выбрать Eclipse. Однако, если вы можете себе это позволить, в IntelliJ Idea есть много удивительных вещей, которых не хватает Eclipse.

# **3 Темы для видео-уроков**

## 

## **3.1 Программа Hello World**

Традиционно первая программа, которую пишут при изучении нового языка программирования, называется программой Hello World. Все, что она делает, это выводит на экран слова Hello, World! На языке Java выглядит это следующим образом на [рисунке 1](#Рисунок1):

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 1

Когда эта программа выполняется, она выводит следующее:

Hello, World!

Обратить внимание, что вывод не включает кавычки.

Программы Java состоят из определений классов и методов, а методы состоят из утверждений. Оператор – это строка кода, которая выполняет основное действие. В программе Hello World эта строка является оператором print, который выводит сообщение пользователю:

System.out.println("Hello, World!");

System.out.println выводит результаты на экран; название println означает print line. Сбивает с толку то, что print может означать как вывод на экран, так и отправку на принтер. В этой курсовой работе нужно принять display-вывод на экран. Как и большинство операторов, оператор print заканчивается точкой с запятой (;).

Java чувствительна к регистру, что означает, что прописные и строчные буквы не совпадают. В программе Hello World слово System должно начинаться с заглавной буквы; system и SYSTEM не подойдут.

Метод – это именованная последовательность утверждений. В этой программе определен один метод с именем main на [рисунке 2](#Рисунок2):

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 2

Имя и формат main особенные: когда программа выполняется, она начинается с первого утверждения в main и заканчивается, когда завершается последнее утверждение. Позже будут упомянуты программы, которые определяют более одного метода.

Эта программа определяет класс с именем Hello. Пока что класс – это коллекция методов; ниже описание об этом подробнее. Можно дать классу любое имя, но обычно оно начинается с заглавной буквы. Имя класса должно совпадать с именем файла, в котором он находится, поэтому данный класс должен находиться в файле с именем Hello.java.

В Java используются фигурные скобки ({ и }) для группировки элементов. В Hello.java крайние скобки содержат определение класса, а внутренние скобки – определение метода.

Строка, начинающаяся с двух косых черт (//), является комментарием, то есть небольшим английским текстом, поясняющим код. Когда Java видит //, она игнорирует все, начиная с этого места и до конца строки. Комментарии не влияют на выполнение программы, но они облегчают другим программистам (и вашему будущему "я") понимание того, что подразумевали сделать.

## **3.2 Объявление переменных**

Одной из самых мощных возможностей языка программирования является возможность определения переменных и работы с ними. Переменная – это именованная область памяти, в которой хранится значение. Значениями могут быть числа, текст, изображения, звуки и другие типы данных. Чтобы хранить значение, сначала нужно объявить переменную как на [рисунке 3](#Рисунок3):



Рисунок 3

Этот оператор называется объявлением, потому что он декларирует, что переменная message имеет тип String. Каждая переменная имеет тип, который определяет, какие значения она может хранить. Например, тип int может хранить целые числа, такие как 1 и -5, а тип char может хранить символы, такие как 'A' и 'z'.

Некоторые типы начинаются с заглавной буквы, а некоторые со строчной. Значение этого различия будет описано ниже. Основное , что нужно понять не существует таких типов, как Int или String.

Чтобы объявить целочисленную переменную с именем x, достаточно набрать следующее: int x;

Обратите внимание, что x – это произвольное имя переменной. В общем случае следует использовать имена, которые указывают, что означают переменные, а именно на [рисунке 4](#рисунке4) показано как нужно:

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 4

В этом примере объявлены две переменные с типом String и две с типом int. Последняя строка показывает, как объявить несколько переменных одного типа: час и минута являются целыми числами. Следует обратить внимание, что каждое объявление заканчивается точкой с запятой (;).

Имена переменных обычно начинаются со строчной буквы, в отличие от имен классов (например, Hello), которые начинаются с заглавной буквы. Когда имя переменной содержит более одного слова (например, firstName), принято писать первую букву каждого последующего слова с заглавной буквы. Имена переменных чувствительны к регистру, поэтому firstName – это не то же самое, что first name или FirstName.

Можно использовать любое имя для переменной. Но есть около 50 зарезервированных слов, называемых ключевыми словами, которые запрещено использовать в качестве имен переменных. Эти слова включают :public, class, static, void и int, которые используются компилятором для анализа структуры программы.

Можно ознакомиться с полным списком ключевых слов, но не нужно их запоминать. Большинство редакторов программирования обеспечивают подсветку синтаксиса, благодаря которой различные части программы отображаются разными цветами. И компилятор будет жаловаться, даже если одно из ключевых слов проскочит мимо вас и вашего редактора.

### 3.2.1 Назначение переменных

Теперь, когда объявлены некоторые переменные, можно использовать их для хранения значений. Делать это можно с помощью оператора присваивания, пример на [рисунке 5](#рисунке5):

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 5

В этом примере показаны три присваивания, а комментарии иллюстрируют различные способы, которыми люди иногда говорят об операторах присваивания. Лексика здесь может быть запутанной, но идея проста: объявление переменной создает именованное место хранения. Когда происходит присваивание переменной – обновляется ее значение.

Как правило, переменная должна иметь тот же тип, что и значение, которое ей присваивается. Например, нельзя хранить строку в minute или целое число в message. Ниже несколько примеров, которые, кажется, нарушают это правило, но вернемся к этому позже.

Частым источником путаницы является то, что некоторые строки выглядят как целые числа, но таковыми не являются. Например, на [рисунке 6](#рисунке6) сообщение может содержать строку "123", которая состоит из символов '1', '2' и '3'. Но это не то же самое, что целое число 123:

Изображение выглядит как текст, устройство, счетчик, датчик

Автоматически созданное описание

Рисунок 6

Переменные должны быть инициализированы (присвоены в первый раз), прежде чем их можно будет использовать. Можно объявить переменную, а затем присвоить ей значение, как в предыдущем примере. Также можно объявить и инициализировать переменную в одной строке, пример на [рисунке 7](#рисунке7):

### Изображение выглядит как текст Автоматически созданное описание

Рисунок 7

### 3.2.2 Печать переменных

Можно вывести на экран текущее значение переменной с помощью print или println. В приведенных на [рисунке 8](#рисунке8) операторах объявляется переменная firstLine, ей присваивается значение "Hello, again!" и выводится это значение:

Изображение выглядит как текст, оранжевый

Автоматически созданное описание

Рисунок 8

Когда говорится об отображении переменной, обычно имеется в виду ее значение. Чтобы отобразить имя переменной, необходимо заключить его в кавычки, как на [рисунке 9](#рисунке9):

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 9

Для этого примера вывод будет следующим:

Значение firstLine равно Hello, again!

Удобно, что код для отображения переменной одинаков независимо от ее типа. Например, на [рисунке 10](#рисунке10):

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 10

Вывод этой программы показан здесь:

Текущее время: 11:59.

Для вывода нескольких значений на одной строке обычно используется несколько состояний print, за которыми в конце следует println. Но не следует забывать про println! На многих компьютерах вывод print сохраняется без отображения до выполнения println; тогда вся строка выводится сразу. Если опустить println, программа может вывести сохраненный вывод в неожиданное время или даже завершиться, ничего не показав.

### 3.2.3 Арифметические операторы

Операторы – это символы, которые представляют простые вычисления. Например, оператор сложения – это +, вычитания – это -, умножения – это \*, а деления – это /.

Следующая программа на [рисунке 11](#рисунке11) преобразует время суток в минуты:

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 11

Вывод выглядит следующим образом:

Количество минут с полуночи: 719

В этой программе час \* 60 + минута – это выражение, которое представляет собой одно вычисляемое значение (719). Когда программа выполняется, каждая переменная заменяется своим текущим значением, а затем применяются операторы. Значения, с которыми работают операторы, называются операндами.

Выражения обычно представляют собой комбинацию чисел, переменных и операторов. При компиляции и выполнении они превращаются в одно значение. Например, выражение 1 + 1 имеет значение 2. В выражении час – 1 Java заменяет переменную на ее значение, в результате чего получается 11-1, которое имеет значение 10.

В выражении час \* 60 + минута обе переменные заменяются, получается 11 \* 60 + 59. Сначала происходит умножение, в результате чего получается 660 + 59. Затем сложение дает 719.

Сложение, вычитание и умножение выполняют то, что ожидается, но деление может удивить. Например, следующий фрагмент на [рисунке 12](#рисунке12) пытается вычислить дробь прошедшего часа:

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 12

Вывод выглядит следующим образом:

Дробь прошедшего часа: 0

Этот результат часто сбивает людей с толку. Значение минуты равно 59, и 59, деленное на 60, должно быть 0,98333, а не 0. Проблема в том, что Java выполняет целочисленное деление, когда операндами являются целые числа. По своей конструкции целочисленное деление всегда округляется в сторону нуля, даже в таких случаях, как этот, когда следующее целое число близко.

В качестве альтернативы можно вычислить процент, а не дробь, как показано на [рисунке 13](#рисунке13):

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 13

Новый вывод будет выглядеть следующим образом:

Процент прошедшего часа: 98

Результат снова округлен в меньшую сторону, но, по крайней мере, теперь он приблизительно правильный.

### 3.2.4 Числа с плавающей точкой

Более общим решением является использование чисел с плавающей точкой, которые представляют значения с десятичными знаками. В Java тип с плавающей точкой по умолчанию называется double, что является сокращением от double-precision. Можно создавать переменные double, как на [рисунке 14](#рисунке14) и присваивать им значения так же, как делалось это для других типов:

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 14

Java выполняет деление с плавающей точкой, когда один или несколько операндов являются двойными значениями. Таким образом, можно решить проблему из предыдущего примера на [рисунке 15](#рисунке15):

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 15

Вывод показан здесь:

Доля прошедшего часа: 0.9833333333333333

Хотя числа с плавающей точкой полезны, они могут быть источником путаницы. Например, Java различает целочисленное значение 1 и значение с плавающей точкой 1.0, даже если кажется, что это одно и то же число. Они принадлежат к разным типам данных, и, строго говоря, не разрешается делать присваивания между типами.

Следующее действие на [рисунке 16](#рисунке16) является незаконным, потому что переменная слева – int, а значение справа – double:



Рисунок 16

Это правило легко забыть, потому что во многих случаях Java автоматически преобразует из одного типа в другой, на [рисунке 17](#рисунке17) показан пример:



Рисунок 17

Предыдущий пример должен быть незаконным, но Java разрешает его, автоматически преобразуя значение int 1 в значение double 1.0. Такая снисходительность удобна, но часто вызывает проблемы у новичков. Например, на [рисунке 18](#рисунке18):



Рисунок 18

Можно ожидать, что переменная y получит значение 0.333333, что является законным значением с плавающей запятой. Но вместо этого она получает значение 0,0. Выражение справа делит два целых числа, поэтому Java выполняет целочисленное деление, которое дает значение int 0. При преобразовании в double значение, присвоенное y, равно 0.0.

Один из способов решения этой проблемы на [рисунке 19](#рисунке19) (как только вы обнаружите ошибку) сделать правую часть выражением с плавающей точкой. Следующее значение y равно 0,333333, как и ожидалось:



Рисунок 19

В соответствии со стилем , всегда нужно присваивать значения с плавающей точкой переменным с плавающей точкой. Компилятор не заставляет делать это, но никогда не знаешь, когда простая ошибка вернется и будет нарушать код.

### 3.2.5 Оператор String

В общем случае нельзя выполнять математические операции над строками, даже если строки выглядят как числа. Следующие выражения являются незаконными: "Hello" - 1 "World" / 123 "Hello" \* "World".

Оператор + работает со строками, но он может делать не то, что ожидается. Для строк оператор + выполняет конкатенацию, что означает объединение из конца в конец. Таким образом, "Hello, " + "World!" дает строку "Hello, World!".

Аналогично, если есть переменная name, имеющая тип String, выражение "Hello, " + name добавляет значение name к строке hello, что создает персонализированное приветствие.

Поскольку сложение определено как для чисел, так и для строк, Java выполняет автоматические конверсии, которых можно не ожидать, например, на [рисунке 20](#рисунке20):



Рисунок 20

Java выполняет эти операции слева направо. В первой строке 1 + 2 равно 3, а 3 + "Hello" равно "3Hello". Но во второй строке "Hello" + 1 – это "Hello1", а "Hello1" + 2 – это "Hello12".

Когда в выражении встречается более одного оператора, они оцениваются в соответствии с порядком выполнения операций. Вообще говоря, Java оценивает операторы слева направо (как видно в предыдущем разделе). Но для числовых операторов Java следует математическим соглашениям:

– Умножение и деление имеют приоритет перед сложением и вычитанием, то есть они выполняются первыми. Поэтому 1 + 2 × 3 дает 7, а не 9, а 2 + 4 / 2 дает 4, а не 3.

– Если операторы имеют одинаковый приоритет, они оцениваются слева направо. Так, в выражении минута × 100 / 60 умножение происходит первым, а если операторы значение минуты равно 59, получаем 5900 / 60, что дает 98. Если бы эти же операции выполнялись справа налево, результат был бы 59 × 1, что неверно.

– Если изменить порядок операций , то можно использовать круглые скобки. Выражения в круглых скобках оцениваются первыми, поэтому (1 + 2) × 3 равно 9. Также можно использовать круглые скобки, чтобы сделать выражение более удобным для чтения, как, например, (минута × 100) / 60, хотя это не меняет результат.

Полная таблица старшинства операторов приведена в официальных руководствах по Java. Если порядок операций не очевиден при рассмотрении выражения, то всегда можно добавить круглые скобки, чтобы сделать его более понятным. Нужно усвоить эти детали языка Java.

## **3.3 Методы**

До сих пор писали программы, в которых есть только один метод, названный main. В этой главе покажу вам, как организовать программы с несколькими методами. Мы также рассмотрим класс Math, который предоставляет методы для общих математических операций. Наконец, обсудим стратегии постепенной разработки и тестирования вашего кода.

### 3.3.1 Определение новых методов

Некоторые методы выполняют вычисления и возвращают результат. Например, nextDouble считывает ввод с клавиатуры и возвращает его в виде двойки. Другие методы, например println, выполняют последовательность действий без возврата результата. Для определения таких методов в Java используется ключевое слово void, пример, на [рисунке 21](#рисунке21):

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 21

В этом примере методы newLine и main являются публичными, что означает, что они могут быть вызваны из других классов. И оба они void, что означает, что они не возвращают результат (в отличие от nextDouble). Вывод программы показан здесь:

Первая строка.

Вторая строка.

Нужно обратить внимание на дополнительное пространство между строками. Если нужно было бы больше пространства между ними, можно вызвать один и тот же метод несколько раз. Или написать еще один метод (с именем threeLine), который выводит три пустые строки, пример, на [рисунке 22](#рисунке22):

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 22

В этом примере имя класса NewLine. По традиции имена классов начинаются с заглавной буквы. NewLine содержит три метода, newLine, threeLine и main. Помните, что Java чувствительна к регистру, поэтому NewLine и newLine – это не одно и то же.

По соглашению имена методов начинаются со строчной буквы и используют верблюжий регистр, что является милым названием для jammingWordsTogetherLikeThis. Можно использовать любое имя для методов, кроме main или любого из ключевых слов Java.

### 3.3.2 Параметры и аргументы

Некоторые из использованных методов требуют аргументов – значений, которые указываются в круглых скобках при вызове метода.

Например, метод println принимает аргумент String. Чтобы вывести сообщение, нужно указать его: System.out.println("Hello"). Аналогично, метод printf может принимать несколько аргументов. Оператор System.out.printf("%d in = %f cmn", inch, cm) имеет три аргумента: строку формата, значение дюйма и значение см.

Когда вызывается метод предоставляются аргументы. Когда определяют метод, то называют параметры, которые являются переменными, указывающими, какие аргументы необходимы. На следующем [рисунке 23](#рисунке23) показан классе пример:

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 23

Метод printTwice имеет параметр s с типом String. Когда вызывается printTwice, должно предоставить аргумент с типом String.

Перед выполнением метода аргумент присваивается параметру. В примере printTwice аргумент "Не заставляйте меня повторять это дважды!" присваивается параметру s.

Этот процесс называется передачей параметров, поскольку значение передается извне метода внутрь. Аргументом может быть любое выражение, поэтому, если есть переменная String, можно использовать ее значение в качестве аргумента, пример, на [рисунке 24](#рисунке24):

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 24

Значение, которое предоставляется в качестве аргумента, должно иметь тот же (или совместимый) тип, что и параметр. Например, если делать следующее, пример, на [рисунке 25](#рисунке25):

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 25

Cинтаксическая ошибка , получаете сообщение об ошибке следующего вида: File: Test.java [line: 10] Error: method printTwice in class Test cannot be applied to given types; required: java.lang.String found: int reason: actual argument int cannot be converted to java.lang.String by method invocation conversion

Это сообщение об ошибке говорит, что найден аргумент int, но требуемый параметр должен быть String. В случае с printTwice Java не будет автоматически преобразовывать целое число 17 в строку "17".

Иногда Java может автоматически преобразовать аргумент из одного типа в другой. Например, для Math.sqrt требуется double, но, если вызывается Math.sqrt(25), целое значение 25 автоматически преобразуется в значение с плавающей точкой 25.0.

Параметры и другие переменные существуют только внутри своих методов. В примере printTwice в методе main нет такой вещи, как s. Если попытаться использовать его там, то получится ошибка компилятора.

Аналогично, внутри printTwice нет такой вещи, как message. Эта переменная принадлежит main. Поскольку переменные существуют только внутри методов, в которых они определены, их часто называют локальными переменными.

### 3.3.3 Множественные параметры

Вот пример на [рисунке 26](#рисунке26) метода, который принимает два параметра:

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 26

Чтобы вызвать этот метод, должно предоставить два целых числа в качестве аргументов, например, как на [рисунке 27](#рисунке27):

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 27

Новички иногда совершают ошибку, объявляя аргументы, например, как на [рисунке 28](#рисунке28):

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 28

Это синтаксическая ошибка, потому что компилятор воспринимает int hour и int minute как объявления переменных, а не как выражения, представляющие значения. Объявлять типы аргументов нельзя в вызове метода с целыми числами, например, на [рисунке 29](#рисунке29):



Рисунок 29

Собрав вместе фрагменты кода, можно получить полную программу, как на [рисунке 30](#рисунке30):

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 30

У printTime есть два параметра, названные час и минута. А у main есть две переменные, также названные hour и minute. Хотя у них одинаковые имена, эти переменные не одинаковы. Час в printTime и час в main ссылаются на разные области памяти и могут иметь разные значения.

Например, на [рисунке 31](#рисунке31) можем вызвать printTime следующим образом:

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 31

Перед вызовом метода Java оценивает аргументы; в данном примере это 12 и 0. Затем она присваивает эти значения параметрам. Внутри printTime значение часа равно 12, а не 11, а значение минуты равно 0, а не 59. Более того, если printTime изменяет один из своих параметров, это изменение не влияет на переменные в main.

### 3.3.4 Возвращаемые значения

Когда вызываем метод void, вызов обычно находится в отдельной строке. Например, на [рисунке 32](#рисунке32):



Рисунок 32

С другой стороны, когда вызывается метод, возвращающий значение, с возвращаемым значением нужно что-то сделать. Обычно присваивается его переменной или используем как часть выражения, например, как на [рисунке 33](#рисунке33):

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 33

По сравнению с методами void, методы с возвратом значения отличаются двумя способами:

–Они объявляют тип возвращаемого значения (тип возврата).

– Они используют по крайней мере один оператор return для предоставления возвращаемого значения.

Вот пример из программы с именем Circle.java. Метод calculateArea принимает double в качестве параметра и возвращает площадь круга с данным радиусом (т. е. πr2), пример на [рисунке 34](#рисунке34):

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 34

Как обычно, этот метод является публичным и статическим. Но в том месте, где в прошлых примерах был void, теперь double, что означает, что возвращаемое значение из этого метода – double.

Последняя строка – это новая форма оператора return, которая означает: "Вернитесь немедленно из этого метода и используйте следующее выражение в качестве возвращаемого значения". Выражение, которое предоставляется, может быть произвольно сложным, поэтому написать этот метод можно более лаконично, как на [рисунке 35](#рисунке35):

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 35

С другой стороны, временные переменные типа result часто облегчают отладку, особенно когда просматривается код с помощью интерактивного отладчика.

На [рисунке 36](#рисунке36) показано, как значения данных проходят через программу. Когда метод main вызывает calculateArea, параметру radius присваивается значение 5.0. Затем calculateArea возвращает значение 78.54, которое присваивается переменной area.

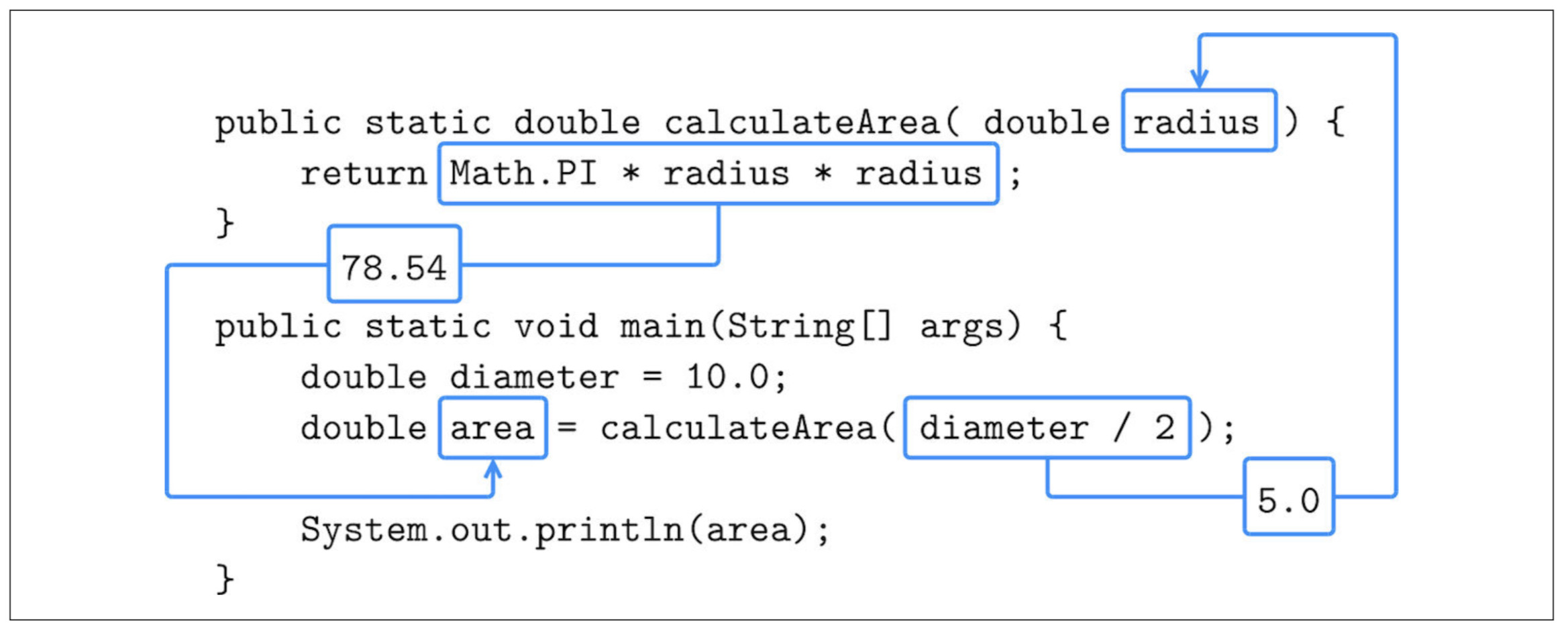


Рисунок 36

Тип выражения в операторе возврата должен соответствовать типу возврата самого метода. Когда объявляется, что возвращаемый тип – double, этот метод ,в конечном итоге, выдаст двойное значение. Если пытаться вернуть значение без выражения или выражение с неправильным типом, то компилятор выдаст ошибку.

## **3.4 Цепочки и вложенность**

Программы, описанные в предыдущих главах, каждый раз выполняют практически одно и то же действие, независимо от входных данных. Для более сложных вычислений программы обычно реагируют на входные данные, проверяют определенные условия и генерируют соответствующие результаты. В этом видео представлены возможности языка Java для выражения логики и принятия решений.

### 3.4.1 Реляционные операторы

В Java есть шесть реляционных операторов, которые проверяют связь между двумя значениями (например, равны ли они или одно больше другого). Следующие выражения показывают, как они используются:

1. x==y; // x равен y;
2. x!=y; // x не равно y;
3. x>y; // x больше y;
4. x<y; // x меньше y;
5. x>=y; // x больше или равно y ;
6. x<=y // x меньше или равно y;

Результатом реляционного оператора является одно из двух специальных значений: true или false. Эти значения относятся к типу данных boolean, названному в честь математика Джорджа Буля. Он разработал алгебраический способ представления логики.

Распространенной ошибкой является использование одинарного = вместо двойного == при сравнении значений. Оператор = – это оператор присваивания, а == – реляционный оператор. Кроме того, операторы =< и => не существуют.

Две стороны реляционного оператора должны быть совместимы. Например, выражение 5 < "6" недействительно, потому что 5 – это int, а "6" – String. При сравнении значений различных числовых типов Java применяет те же правила преобразования, что и в предыдущем случае ранее с оператором присваивания. Например, при вычислении выражения 5 < 6.0 Java автоматически преобразует 5 в 5.0.

### 3.4.2 Выражение if-else

Чтобы писать полезные программы, почти всегда нужно проверять условия и реагировать соответствующим образом. Условные операторы дают такую возможность. Простейшим условным оператором в Java является оператор if, пример на [рисунке 37](#рисунке37):

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 37

Выражение в круглых скобках называется условием. Если оно истинно, выполняются операторы в скобках. Если условие ложно, то выполнение пропустит этот блок кода. Условие в круглых скобках может быть любым булевым выражением.

Вторая форма условного оператора имеет две возможности, обозначаемые символами if и else. Эти возможности называются ветвями, а условие определяет, какая ветвь будет выполнена, пример на [рисунке 38](#рисунке38):

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 38

Если остаток при делении x на 2 равен 0, знаем, что x четный, и программа выводит соответствующее сообщение. Если условие ложно, то вместо него выполняется второй оператор печати. Поскольку условие должно быть истинным или ложным, будет выполнена ровно одна из ветвей.

Скобки необязательны для ветвей, содержащих только один оператор. Поэтому предыдущий пример можно было бы записать так, ка на [рисунке 39](#рисунке39):

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 39

Однако, лучше использовать скобки, даже если они необязательны, как на [рисунке 40](#рисунке40), чтобы избежать ошибки добавления утверждений в однострочный блок if или else. Этот код вводит в заблуждение, потому что отступы в нем сделаны неправильно:

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 40

Поскольку здесь нет скобок, только первый println является частью оператора if. Вот что на самом деле видит компилятор, как на [рисунке 41](#рисунке41):

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 41

В результате второй println выполняется независимо от того, что происходит. Даже опытные программисты допускают эту ошибку.

Во всех предыдущих примерах в конце строк if и else нет точки с запятой. Вместо этого следует определить новый блок с помощью фигурных скобок. Другая распространенная ошибка – ставить точку с запятой после условия, как в этом примере на [рисунке 42](#рисунке42):

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 42

Этот код будет скомпилирован, но программа выведет "x is even" независимо от значения x. Вот тот же некорректный код с лучшим форматированием на [рисунке 43](#рисунке43):

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 43

Из-за точки с запятой оператор if компилируется так, как будто в нем нет скобок, и последующий блок выполняется независимо. Как правило, каждая строка кода Java должна заканчиваться точкой с запятой или скобкой, но не тем и другим.

Компилятор не будет жаловаться, если опустить необязательные скобки или если написаны пустые утверждения. Это разрешено языком Java, но это часто приводит к ошибкам, которые трудно найти. Инструменты разработки, такие как Checkstyle, могут предупредить вас об этих и других ошибках программирования.

### 3.4.3 Цепочку и вложенность

Иногда нужно проверить связанные условия и выбрать одно из нескольких действий. Одним из способов сделать это является цепочка блоков if и else, как на [рисунке 44](#рисунке44):

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 44

Эти цепочки могут быть сколь угодно длинными, хотя их трудно читать, если они выходят из-под контроля. Один из способов облегчить их чтение – использовать стандартные отступы, как показано в этих примерах. Если все утверждения и скобки выстроены в ряд, вероятность синтаксических ошибок снижается.

Последняя ветвь – это просто else, а не else if (x == 0). На данном этапе цепочки x не является положительным и x не является отрицательным. Нет необходимости проверять, равен ли x 0, потому что другой возможности нет.

Помимо цепочки, можно принимать сложные решения путем вложения одного условного оператора в другой. Предыдущий пример можно написать как на [рисунке 45](#рисунке45):

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 45

Внешнее условие имеет две ветви. Первая ветвь содержит оператор печати, а вторая ветвь содержит другой условный оператор, который имеет две собственные ветви. Эти две ветви также являются операторами печати, но они могли бы быть и условными операторами.

Такие вложенные структуры встречаются часто, но их очень быстро становится трудно читать. Для того чтобы структура (или предполагаемая структура) стала очевидной для читателя, необходимы хорошие отступы.

### 3.4.4 Заявление-переключатель

Если нужно принять серию решений, цепочка блоков else if может стать длинной и избыточной. Например, на [рисунке 46](#рисунке46) программа, которая преобразует целые числа 1, 2 и 3 в слова "один", "два" и "три":

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 46

Эта цепочка может продолжаться до бесконечности, особенно для банковских программ, которые записывают числа в длинной форме (например, "сто двадцать три и 45/100 долларов"). Альтернативный способ оценить множество возможных значений выражения – использовать оператор switch, как на [рисунке 47](#рисунке47):

Изображение выглядит как текст, табло, снимок экрана

Автоматически созданное описание

Рисунок 47

Тело оператора switch состоит из одного или нескольких блоков case. Каждый случай заканчивается оператором break, который завершает тело switch. Блок по умолчанию является необязательным и выполняется только в том случае, если ни один из случаев не применим.

Хотя операторы switch([рисунок 48](#рисунке48)) выглядят длиннее, чем цепочки блоков else if, они особенно полезны, когда можно сгруппировать несколько случаев:

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 48

## **3.5 Массивы и ссылки**

До этого момента использовались переменные только для отдельных значений, таких как числа или строки. В этом видео узнаем, как хранить несколько значений одного типа с помощью одной переменной. Эта особенность языка позволит писать программы, манипулирующие большими объемами данных.

Например, на [рисунке 49](#рисунке49) предлагалось проверить, встречается ли каждая буква в строке ровно два раза. Один из них проходит по строке 26 раз, по одному разу для каждой строчной буквы:

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 49

Такой подход с вложенными циклами неэффективен, особенно когда строка длинная. Например, в книге "Война и мир" более 3 миллионов символов; чтобы обработать всю книгу, вложенный цикл выполнится около 80 миллионов раз.

Другой алгоритм мог бы инициализировать 26 переменных нулями, просмотреть строку один раз и использовать гигантский оператор if для обновления переменной для каждой буквы. Но кто захочет объявлять 26 переменных?

Вот тут-то и приходят на помощь массивы. Можно использовать одну переменную для хранения 26 целых чисел. Вместо того, чтобы использовать оператор if для обновления каждого значения, можно использовать арифметику для обновления n-го значения напрямую.

### 3.5.1 Создание массивов

Массив – это последовательность значений; значения в массиве называются элементами. Можно создать массив из int’s, double’s, String’s или любого другого типа, но все значения в массиве должны иметь одинаковый тип.

Чтобы создать массив, нужно объявить переменную с типом массива, а затем создать сам массив. Типы массивов выглядят как другие типы Java, за исключением того, что за ними следуют квадратные скобки ([]). Например, на [рисунке 50](#рисунке50) объявлено, что counts – это целочисленный массив, а values – double массив:

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 50

Чтобы создать сам массив, необходимо использовать оператор new. Оператор new выделяет память для массива и автоматически инициализирует все его элементы нулями, пример на [рисунке 51](#рисунке51):

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 51

Первое присваивание делает counts ссылкой на массив из четырех целых чисел. Второе заставляет значения ссылаться на массив двойных чисел, но количество элементов зависит от значения size (в момент создания массива).

Конечно, также можно объявить переменную и создать массив одной строкой кода, как на [рисунке 52](#рисунке52):

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 52

Можно использовать любое целочисленное выражение для размера массива, если значение не является отрицательным. Если попытаться создать массив, например, с -4 элементами, вы получите NegativeArraySizeException. Массив с нулевыми элементами допускается, и для таких массивов есть специальные применения.

Можно инициализировать массив последовательностью элементов, разделенных запятыми и заключенных в фигурные скобки, как на [рисунке 53](#рисунке53):



Рисунок 53

Этот оператор создает переменную массива, a, и заставляет ее ссылаться на массив с четырьмя элементами.

### 3.5.2 Доступ к элементам

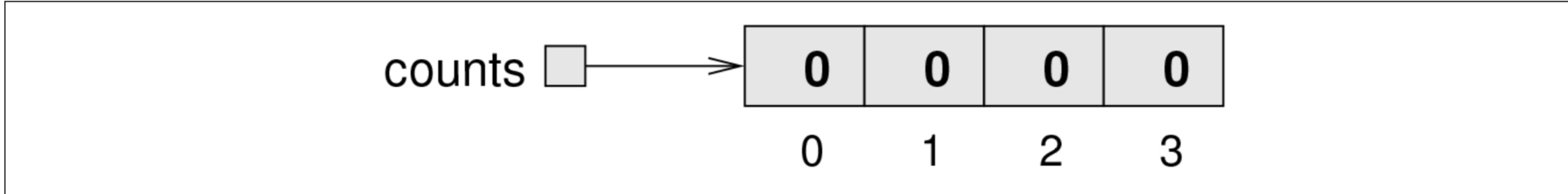
Когда создается массив с помощью оператора new, элементы инициализируются нулем. На [рисунке 54](#рисунке54) показана диаграмма памяти массива counts.

Рисунок 54

Стрелка показывает, что значение counts является ссылкой на массив. Надо знать о массиве и переменной, которая ссылается на него, как о двух разных вещах. Можно назначить другую переменную для ссылки на тот же массив, а также изменить значение counts для ссылки на другой массив.

Жирные цифры внутри ячеек – это элементы массива. Более светлые числа вне рамок – это индексы, используемые для идентификации каждого места в массиве. Как и в случае со строками, индекс первого элемента равен 0, а не 1. По этой причине называется первый элемент нулевым.

Оператор [] выбирает элементы из массива, как на [рисунке 55](#рисунке55):



Рисунок 55

Оператор [] можно использовать в любом месте выражения, как на [рисунке 56](#рисунке56):

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 56

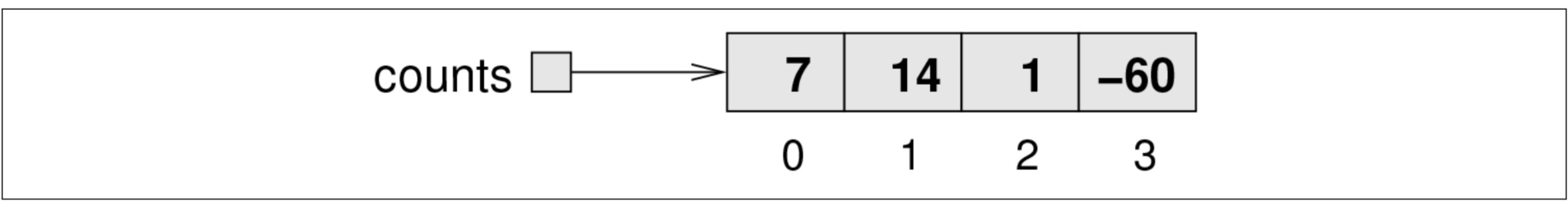
На [рисунке 57](#рисунке57) показан результат выполнения этих выражений.

Рисунок 57

В качестве индекса используется любое выражение, если оно имеет тип int. Один из самых распространенных способов индексирования массива – переменная цикла. Например, на [рисунке58](#рисунке58):

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 58

Этот цикл while считает от 0 до 4. Когда i равно 4, условие не выполняется и цикл завершается. Таким образом, тело цикла выполняется только тогда, когда i равно 0, 1, 2 или 3. В данном контексте имя переменной i является сокращением от index.

При каждом прохождении цикла используется i как индекс массива, отображая на экране i-й элемент. Такой тип обработки массива обычно записывается в виде цикла for, как на [рисунке 59](#рисунке59):

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 59

Для массива counts единственными допустимыми индексами являются 0, 1, 2 и 3. Если индекс отрицательный или больше 3, результатом будет исключение ArrayIndexOutOfBoundsException.

### 3.5.3 Отображение массивов

Можно использовать println для вывода массива на экран, но это, скорее всего, не даст того результата, который можно было получить. Например, на [рисунке 60](#рисунке60) если выводить массив следующим образом:

Изображение выглядит как текст, оранжевый, темный

Автоматически созданное описание

Рисунок 60

Вывод будет выглядеть примерно так: [I@bf3f7e0

Скобка указывает на то, что значение является массивом, I означает целое число, а остальное представляет собой адрес массива в памяти.

Если вывести элементы массива на экран, можно сделать это самим, например, как на [рисунке 61](#рисунке61):

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 61

Учитывая предыдущий массив, вывод printArray будет следующим: {1, 2, 3, 4}

В библиотеке Java есть класс java.util.Arrays, который предоставляет методы для работы с массивами. Один из них, toString, возвращает строковое представление массива. Импортировав Arrays, можно вызвать toString следующим образом, как на [рисунке 62](#рисунке62):

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 62

Выходные данные показаны здесь: [1, 2, 3, 4].

Arrays.toString использует квадратные скобки вместо фигурных скобок. Но это лучше, чем написать свой собственный метод printArray.

# **4 Среда создания видео VEGAS Pro**

Видеоредакторы, то есть программы для редактирования видео, занимают отдельную нишу на рынке программного обеспечения [2]. Существует множество программ для видеомонтажа, но наиболее известными и популярными являются всего несколько:

– Pinnacle Video Studio. Эта программа проста в использовании, но очень ограничена в своих возможностях. Это скорее программа для тех, кто хочет быстро создать фильм, но не хочет углубляться в детали монтажа;

– Corel Video Studio. Об этой программе можно сказать то же самое, что и о вышеупомянутой;

– Windows Movie Maker. Программа встроена в операционные системы Windows (для Windows 7 ее можно скачать бесплатно в составе пакета Windows Live). Единственный "плюс" этой программы в том, что поскольку она уже является частью операционной системы, вам не придется за нее платить. Возможности этого видеоредактора очень и очень скромные;

– Adobe Premiere Pro. Очень мощная и гибкая программа профессионального уровня. Но на ее освоение требуется много времени, поэтому не каждый новичок сможет справиться с ней без специальной подготовки;

– Sony Vegas Pro. Эта программа сочетает в себе возможности профессионального уровня и простоту интерфейса. Здесь нет ничего лишнего, но есть все инструменты, необходимые для качественного видеомонтажа.

Программа Sony Vegas Pro отличается высокой скоростью работы и низкими системными требованиями. Sony Vegas Pro – это программа, которую выбирают как новички, так и профессионалы, потому что она действительно универсальна. Именно об этом и пойдет речь в этой главе.

Кроме описанных выше программ для видеомонтажа, существует множество других. Остановимся на программе Sony Vegas Pro. Как видно из названия, в создании программы участвует компания Sony, более известная как производитель качественной электроники. Кстати, они также выпускают видеокамеры, в том числе и профессионального уровня.

Sony Vegas Pro долгое время был в тени других видеоредакторов, но в последнее время он становится все более популярным. И это вполне объяснимо. Программа, как уже говорилось, нетребовательна к вычислительным ресурсам системы, имеет удобный и простой интерфейс, и в то же время содержит все необходимые для работы инструменты.

Теперь о возможностях Sony Vegas Pro версии 10.

Программа предназначена для редактирования видео. Видеомонтаж предполагает, как минимум, обрезку исходного видеоматериала и объединение его в последовательность в соответствии со сценарием. Но на этом возможности Vegas не заканчиваются. С помощью этой программы можно:

– Переносить DV и HDV видео с видеокамеры на компьютер (поддерживаются и другие форматы, в том числе профессиональные, такие как XDCAM);

– Импортировать аудио с дорожек AudioCD;

– Смонтировать фильм, который может включать видеоклипы, статичные изображения (графические файлы, в том числе с поддержкой альфа-канала), звуковые файлы различных форматов;

– Добавлять титры к фильму. Для создания статичных и бегущих титров в Vegas есть три встроенных инструмента. Интересен относительно новый инструмент ProType Titler, который позволяет создавать различные титры с анимированными параметрами, например, титры, плавно изменяющие расстояние между персонажами;

– Дополнять свой фильм различными видеопереходами. Vegas имеет огромную библиотеку видеопереходов, параметры которых можно настраивать;

– Добавлять в фильм спецэффекты. С помощью спецэффектов можно как отрегулировать качество изображения, так и придать ему особый стиль, например, размыть или добавить эффект свечения;

– Выводить готовый фильм на видеокамеру, в видеофайлы различных форматов, а также на CD, DVD и Blue-ray диски;

Перечислены только основные возможности программы, которые, скорее, являются этапами создания фильма;

Интерфейс Sony Vegas Pro не сильно меняется от версии к версии. Однако, в каждой последующей версии появляется все больше новых возможностей. В десятой версии программы появилась уникальная возможность создавать собственные фильмы с трехмерным изображением (3D фильмы). Причем поддерживаются многие форматы 3D-видео, включая самый доступный: с использованием очков с красным и синим фильтрами. То есть можно создать собственный стереоскопический фильм для просмотра на мониторе или телевизоре.

Также в новой версии появилась возможность стабилизации видеоизображения. С помощью этого инструмента можно частично или полностью устранить неприятный эффект дрожания видеоизображения, снятого "с рук". Для видеоматериала, снятого с использованием штатива, стабилизация чаще всего не нужна.

В версии 10 реализована поддержка аппаратного ускорения при декодировании видео формата H.264. Звуковые эффекты теперь можно применять к каждому аудиоклипу отдельно, а не ко всей аудиодорожке, как в предыдущих версиях.

Реализована поддержка работы с субтитрами. Также добавлено множество внутренних нововведений, которые расширяют технические возможности программы, но не видны явно [2].

# **ЗАКЛЮЧЕНИЕ**

В заключении курсовой работы можно подвести итоги: почему стоит выбрать Java для работы и развития своих знаний в этой области, что узнали за весь курс изучения Java, что изучать после начального введения в программирования на языке Java.

1) Java – это многопрофильный язык программирования, поскольку он используется для написания логики на стороне сервера для распределенных систем, а также настольных, мобильных и веб-приложений. Поэтому ориентация на Java дает широкий выбор направления в мире разработки [9]. Легко можно начать заниматься другим видом разработки, не меняя языка программирования [3].

В то же время Java – очень удобная платформа с точки зрения наличия готовых решений и компонентов [3]. Наличие огромного сообщества делает процесс обучения и вхождения в профессию более легким и быстрым [9]. В финансовом плане спрос на Java-разработчиков растет с каждым днем, поэтому Java достаточно стабильная среда, где можно быть уверенным в завтрашнем дне [3].

2) За курс обучения изучены основы языка программирования Java: вывод текста, переменные, операторы, массивы, методы, цепочки и вложенности.

3) Данный курс ограничен только основами Java, после основ идёт ООП(объект-ориентированное программирование). В разделе ООП изучаются коллекции, конструкторы, абстракция, интерфейсы, инкапсуляция, наследование, полиморфизм и т. д.

Если вдруг язык Java не понравился написанием и чтением кода, можно перейти на изучение других языков программирования, так как основные темы, которые изучены в данном курсе, не сильно отличаются, разница в других языках программирования – только синтаксическое написание кода.

По моему мнению, можно научиться программировать с помощью видеоуроков на языке Java, в удобное время и в удобной форме.

**СПИСОК ИСПОЛЬЗУЕМОЙ ЛИТЕРАТУРЫ**

1. Патащинский В. С. Видеомонтаж в Sony Vegas Pro 10 / В. С. Патащинский. – М.: Питер, 2013. 272 с.
2. 7 причин изучать Java 23 августа 2021 // – URL: <https://vc.ru/dev/284968-7-prichin-vybrat-java-dlya-izucheniya> (дата обращения 20.12.2021)
3. Язык программирования Java: особенности, популярность, ситуация на рынке труда 15 марта 2021 // – URL: <https://ru.hexlet.io/blog/posts/yazyk-programmirovaniya-java-osobennosti-populyarnost-situatsiya-na-rynke-truda> (дата обращения 20.12.2021)