Московский Авиационный Институт  
(Национальный Исследовательский Университет)

Факультет прикладной математики и информатики  
Кафедра вычислительной математики и программирования

**Курсовой проект  
По дисциплине  
«Вычислительные системы»  
1 семестр  
Реферат:**

**«Язык программирования Java»**

|  |  |
| --- | --- |
| **Студент:** | Армишев К. К. |
| **Группа:** | М8О-108Б-21 |
| **Преподаватель:** | Трубченко Н. М. |
| **Подпись:** |  |
| **Оценка:** |  |
| **Дата:** | 22.12.2021 |

Москва

2021

**Содержание**

**Введение** 3

1. История создания языка Java 4

2. Основные особенности Java 8

3. Основные библиотеки и фреймворки Java 12

4. Сравнение Java, Python и С 18

5. Сфера применения языка Java 20

**Заключение** 21

**Список литературы** 22

# Введение

В современном мире создаваемые приложения должны быть высокопроизводительны, безопасны и быть нейтральными к архитектуре. Все эти факторы привели к необходимости нового взгляда на сам процесс создания и распределения приложений на множестве машин различной архитектуры. Требования к переносимости заставили отказаться от традиционного способа создания и доставки бинарных файлов, содержащих машинные коды и, следовательно, привязанных к определенной платформе. Системат разработки Java удовлетворяет всем этим требованиям. Java – объектно-ориентированный язык, удобный и надёжный в эксплуатации благодаря таким своим достоинствам, как многозадачность, поддержка протоколов Internet и многоплатформенность. Java – это интерпретируемый язык, и каждая Java-программа компилируется для гипотетической машины, называемой Виртуальная Машина Java. Результатом такой компиляции является байт-код Java, который в свою очередь может выполняться на любой операционной системе при условии наличия там системы времени выполнения Java, которая интерпретирует байт-код в реальный машинный код конкретной системы.

Язык Java является объектно-ориентированным и поставляется с достаточно объемной библиотекой классов. Библиотеки классов Java значительно упрощают разработку приложений, предоставляя в распоряжение программиста мощные средства решения распространенных задач. Поэтому программист может больше внимания уделить решению прикладных задач, а не таких, как, например, организация динамических массивов, взаимодействие с операционной системой или реализация элементов пользовательского интерфейса.

Целью данной работы является история языка программирования Java, рассмотрение преимуществ данного языка над другими.

1. **История создания языка Java**

## **Команда для разработки нового языка**

Платформа и язык Java начинались как внутренний проект в Sun Microsystems в декабре 1990 года, предоставляя альтернативу языкам программирования C и C++. Инженер Patrick Naughton (Патрик Нотон, американский разработчик программного обеспечения) организовал команду, в которую входили Джеймс Гослинг (James Gosling), Майк Шеридан (Mike Sheridaп), Крис Уарт (Chris Warth) и Эд Франк (Ed Frank).

Команда начала работу над Green Project предусматривающим разработку операционной системы для электронных бытовых устройств. В разработке и развитии нового языка приняли участие немало других специалистов: Билл Джой (Bill Joy), Артур ван Хофф (Arthur van Hoff), Джонатан Пэйн (Jonathan Payne), Франк Йеллин (Frank Yellin) и Тим Линдхольм (Тlm Lindholm) которые внесли основной вклад в развитие исходного прототипа Java.

## **Поиск базы для нового языка**

Первоначально команда рассматривала возможность использования модифицированного языка C++, но отклонила его по причинам требовательности к объемам памяти и высокой сложности. Отсутствие в С++ сборки мусора означало, что программистам приходилось вручную управлять системной памятью, сложной и подверженной ошибкам задачей. Тем более что они разрабатывали встроенную систему с ограниченными ресурсами. Наконец, они хотели получить платформу, которую будет легко портировать на любые типы устройств. Но все же синтаксис Java во многом заимствован из C и C ++, а объектно-ориентированные функции смоделированы на основе языков С++, Smalltalk and Objective-C.

## **Язык программирования Oak**

Разработкой нового языка вплотную занимался Джеймс Гослинг и изначально назвал его Оak (в переводе с английского Дуб). Рядом с офисом разработчика действительно рос дуб. Наверное Джеймс, работая в офисе, поглядывал в окно и видя крепкий дуб, излучающий здоровье и долголетие, решил назвать новый язык в честь зеленого дерева. К лету 1992 года команда смогла продемонстрировать части новой платформы, включая операционную систему Green OS, язык Oak, библиотеки и аппаратное обеспечение.

Их первая попытка, продемонстрированная 3 сентября 1992 года, была сосредоточена на создании устройства PDA (Personal Digital Assistant, карманный компьютер) под названием Star7, которое имело графический интерфейс и смарт-агента под названием «Duke» для помощи пользователю. Карманный компьютер Star7 был принципиально новым устройством, которое намного опередило своё время, но из-за высокой стоимости не смогло произвести переворот в мире технологии и постепенно было забыто. Возможно, в настоящее время именно оно вернулось к нам в виде умных андроид устройств.

## **Происхождение названия Java**

1994 году язык Oak был переименован в виду того, что название «Oak» уже использовалось как торговая марка компании Oak Technology (американская организация, поставщик электронных компонентов). Oak был переименован в Java, в честь марки кофе которое получило название одноименного острова Ява. Поэтому на официальной эмблеме изображена чашка с горячим кофе. Существует и другая версия происхождения названия языка, связанная с шутливым намеком на кофе-машину как пример бытового устройства, для программирования которых изначально язык и создавался. Наконец Java 1.0 была окончательно выпущена в 1996 году.

## **Предпосылки Java**

До появления языка Java программистам, приходилось выбирать между языками, которые обеспечивали только узкий спектр характеристик. Первые языки программирования BASIC, COBOL и FORTRAN были разработаны без учета принципов структурирования. Структурированный язык Pascal не был предназначен для написания разнообразных программ. Язык С++ частично решал эти проблемы. Но трудность применения С и С++ состоит в том, что написанные на них программы должны компилироваться для конкретной платформы.

Пытаясь найти решение этих проблем, Джеймс Гослинг и другие разработчики начали работу над переносимым, не зависящим от конкретной платформы языком. Исходный код, созданный на таком языке, должен был выполняться на разнотипных процессорах в различных операционных системах. И в итоге их усилия привели к созданию первого в мире кроссплатформенного языка под названием Java.

## **Концепции языка Java**

Язык Jаvа был задуман как простой в изучении и эффективный в употреблении профессиональными программистами. Овладеть языком Java тем, у кого имеется некоторый опыт программирования, не составит особого труда. Группа разработчиков обобщила основные понятия Java и составила следующий перечень его особенностей:

* простота
* безопасность
* переносимость
* объектная ориентированность
* надежность
* многопоточность
* архитектурная нейтральность
* интерпретируемость
* высокая производительность
* распределённость
* динамичность

## **Java и С++**

Многие особенности Jаvа унаследованы от языков С и С++. Вместе с тем языку Java присущи значительные практические и концептуальные отличия. Хотя С++ оказал влияние на характеристики Jаvа, но всё же язык программирования Java не является усовершенствованной версией С++. Разумеется, у него немало сходств с языком С++, и в исходном коде программы на Java программист С++ будет чувствовать себя почти как дома.

Но есть и коренные отличия двух языков: программа на C++ компилируется только для конкретной среды выполнения, что делает C++ зависимым от платформы языком. В Java исходный код сначала преобразуется в байт-код во время компиляции. Этот байт-код затем интерпретируется интерпретатором и средой выполнения для создания выходных данных, которые делают Java независимым от платформы языком. Язык Jаvа предназначен не для замены С++, а для решения других задач, для которых язык С++ не годится.

## **Java и C#**

Платформа .NET представляет собой примерно то же самое, что и виртуальная машина Java (JVM) и библиотеки Java, а язык C# имеет явное сходство с языком Java. Вне всяких сомнений это лучшее, что создала компания Microsoft в области языков и сред программирования. Впервые с момента своего появления y Java появился реальный соперник.

Языки C# и Java появились в разное время. Язык Java был создан задолго до появления C#. Таким образом, если Java создавался, опираясь в большей степени на опыт языков Objective-C и C, С++, то для C# такой опорой уже являлись C++ и сам Java. И, несмотря на своё название, C# оказался ближе к Java, чем к C++.

С точки зрения разработчика языки Java и C# очень похожи. Оба языка являются строго типизированными, объектными. Оба языка опираются на сборку мусора. Оба языка сопровождаются богатыми коллекциями библиотек. Но есть в языках также свои особенности и различия. C# учёл многие недостатки Java, и исправил их в своей реализации. Но и Java не стоит на месте, развиваясь параллельно с C#.

## **Современный Java**

И сегодня Java остается в лидерах языков программирования. Индекс TIOBE Programming Community - показатель популярности языков программирования, обновляемый ежемесячно. Рейтинги основаны на количестве квалифицированных инженеров по всему миру, курсов и сторонних поставщиков. Популярные поисковые системы также используются для расчета рейтинга. Если взять статистику с 2000 по 2020 годов видно, что язык программирования Java уверенно держится в лидерах, не опускаясь ниже 3-его места. У Java программистов рынок труда один из самых больших. Кроме того, высокая популярность языка означает, что в интернете по нему легко можно найти практически любую информацию.

В мире не может быть более распространенного компьютерного кода, чем на Java. Java везде - практически на каждом мобильном устройстве, сервере, ИТ-системе и сети. Язык Java используется для создания мобильных приложений под операционную систему Android. При этом программы компилируются в нестандартный байт-код. Для такой компиляции используется Android SDK, разработанный компанией Google.

В настоящее время язык и платформа Java активно развивается. 3 миллиарда устройств работают под управлением Java—дома, в автомобиле и офисе. 12 миллионов разработчиков Java по всему миру. Недавно было объявлено, что для Java будет использоваться шестимесячный цикл выпуска. Java сегодня успешна, потому что даже код, написанный 20 лет назад, совершенно точно работает.

Общий вывод получается таким: насчёт "самого популярного" языка можно спорить, но что Java "один из самых популярных" — безусловно.

1. **Особенности Java**

Основным фактором, обусловив­шим изобретение Java, стала потребность в обеспечении переносимости и без­опасности, но свою роль в формировании окончательной версии языка сыграли и другие факторы. Группа разработчиков обобщила основные понятия Java и со­ставила следующий перечень его особенностей:

* простота;
* безопасность;
* переносимость;
* объектная ориентированность;
* надежность;
* многопоточность;
* архитектурная нейтральность;
* интерпретируемость;
* высокая производительность;
* распределенность;
* динамичность.

## **Простота**

Язык Java был задуман как простой в изучении и эффективный в употреблении профессиональными программистами. Овладеть языком Java тем, у кого имеется некоторый опыт программирования, не составит особого труда. Если же вы уже знакомы с основными принципами объектно-ориентированного программирова­ния, то изучить Java вам будет еще проще. А от тех, кто имеет опыт программиро­вания на C++, переход к java вообще потребует минимум усилий. Язык Java наследу­ет синтаксис C/C++ и многие объектно-ориентированные свойства C++, поэтому для большинства программистов изучение Java не составит больших трудностей.

## **Объектная ориентированность**

Хотя предшественники языка Java и оказали влияние на его архитектуру и син­таксис, при его проектировании не ставилась задача совместимости по исходному коду с каким-нибудь другим языком. Это позволило группе разработчиков созда­вать Java, по существу, с чистого листа. Одним из следствий этого явился четкий, практичный, прагматичный подход к объектам. Помимо того, что язык Java поза­имствовал свойства многих удачных объектно-программных сред, разработанных на протяжении нескольких последних десятилетий, в нем удалось достичь золотой середины между строгим соблюдением принципа “все элементы программы являются объектами” и более прагматичного принципа “прочь с дороги”. Объектная модель Java проста и легко расширяема. В то же время такие элементарные типы данных, как целочисленные, сохраняются в виде высокопроизводительных ком­понентов, не являющихся объектами.

## **Надежность**

Многоплатформенная среда веб предъявляет к программам повышенные тре­бования, поскольку они должны надежно выполняться в разнотипных системах. Поэтому способность создавать надежные программы была одним из главных прио­ритетов при pa3pa6oTKeJava. Для обеспечения надежности Bjava накладывается ряд ограничений в нескольких наиболее важных областях, что вынуждает программи­стов выявлять ошибки на ранних этапах разработки программы. В то же время Java избавляет от необходимости беспокоиться по поводу многих наиболее часто встре­чающихся ошибок программирования. А поскольку Java — строго типизированный язык, то проверка кода выполняется во время компиляции. Но проверка кода де­лается и во время выполнения. В результате многие трудно обнаруживаемые про­граммные ошибки, которые часто приводят к возникновению с трудом воспроизво­димых ситуаций во время выполнения, попросту невозможны в программе Hajava. Предсказуемость кода в разных ситуациях — одна из основных особенностей Java.

Чтобы понять, каким образом достигается надежность программ на Java, рас­смотрим две основные причины программных сбоев: ошибки управления памя­тью и неправильная обработка исключений (т.е. ошибки при выполнении). В тра­диционных средах создания программ управление памятью — сложная и трудоем­кая задача. Например, в среде C/C++ программист должен вручную резервировать и освобождать всю динамически распределяемую память. Иногда это ведет к возникновению трудностей, поскольку программисты забывают освободить ранее зарезервированную память или, что еще хуже, пытаются освободить область памяти, все еще используемую другой частью кода. Java полностью исключает такие ситуации, автоматически управляя резервированием и освобождением памяти. (Освобождение оперативной памяти полностью выполняется автоматически, по­скольку Java предоставляет средства сборки неиспользуемых объектов в “мусор”.) В традиционных средах условия для исключений часто возникают в таких ситуа­циях, как деление на нуль или отсутствие искомого файла, а управление ими долж­но осуществляться с помощью громоздких и трудных для понимания конструкций. Java облегчает выполнение этой задачи, предлагая объектно-ориентированный механизм обработки исключений. В грамотно написанной программе Hajava все ошибки при выполнении могут (и должны) обрабатываться самой программой.

## **Многопоточность**

Язык Java был разработан в ответ на потребность создавать интерактивные сетевые программы. Для этой цели в Java поддерживается написание много­поточных программ, способных одновременно выполнять многие действия. Исполняющая система Java содержит изящное, но вместе с тем сложное решение задачи синхронизации многих процессов, которое позволяет строить устойчиво работающие интерактивные системы. Простой подход к организации многопо­точной обработки, реализованный в java, позволяет программистам сосредоточи­вать основное внимание на конкретном поведении программы, а не на создании многозадачной подсистемы.

## **Архитектурная нейтральность**

Основной задачей, которую ставили перед собой разработчики Java, было обе­спечение долговечности и переносимости кода. Одной из главных трудностей, стоявших перед разработчиками, когда они создавали Java, было отсутствие вся­ких гарантий, что код, написанный сегодня, будет успешно выполняться завтра — даже на одном и том же компьютере. Операционные системы и процессоры по­стоянно совершенствуются, и любые изменения в основных системных ресурсах могут стать причиной неработоспособности программ. Пытаясь каким-то образом изменить это положение, разработчики приняли ряд жестких решений в самом языке и виртуальной машине Java. Они поставили перед собой следующую цель:

“написано однажды, выполняется везде, в любое время и всегда”. И эта цель была в значительной степени достигнута.

## **Интерпретируемость и высокая производительность**

Как упоминалось ранее, выполняя компиляцию программ в промежуточное представление, называемое байт-кодом, Java позволяет создавать межплатфор­менные программы. Такой код может выполняться в любой системе, на которой реализована виртуальная машина JVM. С первых же попыток разработать межплатформенные решения удалось достичь поставленной цели, хотя и за счет сни­жения производительности. Как пояснялось ранее, [байт-код Java](file:////coding/3713-байт-код-java-чудеса-или-реальность) был тщательно разработан таким образом, чтобы его можно было с высокой эффективностью преобразовывать непосредственно в машинозависимый код на конкретной платформе с помощью динамического компилятора. Исполняющие системы Java, обе­спечивающие такую возможность, сохраняют все преимущества кода, не завися­щего от конкретной платформы.

## **Распределенность**

Язык Java предназначен для распределенной среды Интернета, поскольку он поддерживает семейство сетевых протоколов TCP/IP. По существу, обращение к ресурсу по унифицированному указателю информационного ресурса (URL) мало чем отличается от обращения к файлу. В Java поддерживается также удаленный вы­зов методов (RMI — Remote Method Invocation). Такая возможность позволяет вы­зывать методы из программ через сеть.

## **Динамичность**

Программы на Java содержат значительный объем данных динамического типа, используемых для проверки полномочий и разрешения доступа к объек­там во время выполнения. Это позволяет безопасно и рационально выполнять динамическое связывание кода. Данное обстоятельство исключительно важно для устойчивости среды Java, где небольшие фрагменты байт-кода могут динами­чески обновляться в действующей системе.

1. **Основные библиотеки и фреймворки Java**

Фреймворки Java — по сути, блоки предварительно написанного кода, в которые программист может добавлять свой код для решения конкретных проблем. Это каркас программной системы (или подсистемы). Он может включать:

* вспомогательные программы;
* библиотеки кода;
* язык сценариев;
* другое ПО, облегчающее разработку и объединение разных компонентов большого программного проекта.

В данном реферате приведу основные фреймворки и библиотеки Java:

### **1. Spring Framework**

Spring Framework — одна из самых мощных и широко используемых платформ Java. Благодаря своей концепции внедрения зависимостей и особенностям аспектно-ориентированного программирования Spring покорила мир разработки.

Это платформа с открытым исходным кодом, используемая для корпоративных приложений. Фреймворк обычно используется для Enterprise Java. Среда чрезвычайно универсальна и может использоваться для создания большинства типов приложений Java. С помощью Spring разработчики могут создавать слабосвязанные модули, в которых зависимости обрабатываются фреймворком, а не зависят от библиотек в коде.

Spring чаще всего используют в разработке веб-приложений. Его применяет ряд технологических гигантов, включая Netflix и Amazon. Главное преимущество фреймворка заключается в том, что он чрезвычайно лёгкий и не требует вызова веб-сервера. С точки зрения эффективности это одна из самых популярных платформ Java. Spring чрезвычайно прост и, как правило, может использоваться даже новичками. Он также обеспечивает обратную совместимость и возможность тестирования.

Фреймворк Spring является исчерпывающим и охватывает множество функций, включая безопасность и настройку подключения к БД, которые легко изучить. Кроме того, поскольку это самый популярный веб-фреймворк, вы можете найти множество документации и активное сообщество.

#### Основные концепции

Внедрение зависимостей (DI) (инверсия управления) — в этом принципе вместо того чтобы последовательно брать управление потоком приложение передаёт управление внешнему контроллеру, который управляет потоком. Внешний контроллер — это события. Когда происходит какое-то событие, поток приложения продолжается. Это придаёт гибкость приложению. В Spring IoC выполняется с помощью DI, которые бывают трёх типов — внедрение установщика, внедрение метода и внедрение конструктора.

В Spring объекты называются bean-компонентами. Существует BeanFactory, которая управляет и настраивает эти bean-компоненты. Вы можете думать о BeanFactory как о контейнере, который создаёт экземпляры.

Большинство приложений используют XML (beans.xml) для конфигурации. ApplicationContext, который является надмножеством BeanFactory, используется для более сложных приложений, которым требуется распространение событий, декларативные механизмы и интеграция с аспектно-ориентированными функциями Spring.

### **2. Hibernate**

Hibernate — это структура объектно-реляционного сопоставления. Основным преимуществом фреймворка Hibernate является то, что он очень эффективен, когда дело доходит до решения часто встречающейся проблемы при работе с базами данных на Java — несоответствия между обработкой данных объектно-ориентированными языками программирования и системами управления реляционными базами данных. Это достигается путём разработки постоянных классов и предоставления программисту возможности эффективно взаимодействовать с базой данных с минимальными изменениями кода.

Возможность выполнять операции с базой данных объектов в Java является проблемой, когда дело доходит до большинства других фреймворков. Но Hibernate очень эффективно решает эту проблему. Он используется рядом крупных компаний, включая IBM, Oracle и Accenture из-за его преимуществ в переносимости. Его можно использовать бесплатно, исходный код полностью открыт.

Хотя Hibernate не является фреймворком полного стека, он полностью изменил наш взгляд на базу данных. Как и SQL, запросы в Hibernate называются HQL (Hibernate Query Language).

Hibernate напрямую сопоставляет классы Java с соответствующими таблицами базы данных и наоборот. Основным файлом в спящем режиме является файл hibernate.cfg.xml, который содержит информацию о сопоставлении классов Java с конфигурацией базы данных.

Hibernate решает две основные проблемы с JDBC — JDBC не поддерживает отношения на уровне объектов, и если вы когда-нибудь решите перейти на другую базу данных, старые запросы могут не работать, что означает много изменений, то есть время и деньги.

Hibernate предоставляет уровень абстракции, так что код слабо связан с базой данных. Такие вещи, как установление соединения с базой данных, выполнение операций CRUD берёт на себя Hibernate. Поэтому разработчикам не нужно это реализовывать, что делает код независимым от используемой базы данных.

Преимущества Hibernate:

* полностью открыт;
* уменьшает избыточность через JDBC API;
* повышает производительность и удобство обслуживания;
* поддерживает API-интерфейсы Persistence;
* обеспечивает связь между приложением и любой базой данных.

### **3. Vaadin**

Фреймворк Vaadin специально ориентирован на доступность UX. Он позволяет программисту создавать настраиваемые веб-приложения, предоставляя ему оптимизированную и простую в использовании платформу для разработки. Vaadin также предоставляет доступ к DOM непосредственно с виртуальной машины Java. Важной особенностью фреймворка является тот факт, что взаимодействие, которое происходит между сервером и браузером, полностью автоматизировано.

Данные могут быть связаны с использованием как MVC, так и MVP через платформу Vaadin. Он поставляется со встроенной поддержкой инфраструктуры Spring, а также поддерживает языки JVM, такие как Scala и Kotlin. Он также имеет поддержку WebSocket, которая обеспечивает автоматическую связь между сервером и клиентом. Как Java, так и HTML можно использовать для построения представлений во фреймворке Vaadin. Он имеет интуитивно понятный пользовательский интерфейс с поддержкой взаимодействий перетаскивания, что значительно упрощает весь процесс веб-разработки для программистов любого уровня подготовки.

Vaadin — это флагманский продукт, а также платформа, на которой разработчики могут выполнять реализацию веб-интерфейса HTML5 с использованием Java. Это платформа с открытым исходным кодом, которая включает в себя веб-фреймворк Java и набор веб-компонентов, а также средства запуска приложений и инструменты. Эти веб-компоненты составляют ядро ​​Vaadin и могут быть настроены для получения высокопроизводительного и универсального кода для настольных и мобильных приложений.

Vaadin Flow — это фактическая часть инфраструктуры Java-платформы Vaadin, которая заботится о взаимодействии клиент-сервер, а также о маршрутизации. С помощью Vaadin Flow вы можете полностью написать веб-приложение на Java без суеты JS или CSS. Компоненты пользовательского интерфейса заботятся о действиях пользователя в браузере, используя автоматическую связь с браузером и сервером. Вы можете легко интегрировать компоненты Vaadin в любую IDE, которую используете, плюс это кроссплатформенный фреймворк, поэтому не нужно беспокоиться о переносе кода на другую платформу.

Особенности Vaadin:

* Java API для всех веб-компонентов;
* безопасная веб-разработка только для Java;
* две встроенные темы: Lumo и Material;
* переключение между тёмным или светлым режимами;
* альтернативы стилей из Java;
* поддержка кнопки «Назад» и ссылок на контент;
* навигация без перезагрузки всей страницы;
* поддержка JVM;
* по умолчанию получает встроенную поддержку Spring Framework;
* может автоматически устанавливать связь клиент-сервер с помощью веб-сокета;
* есть маршрутизатор с вложенным маршрутом, а также поддержка параметров;
* поддерживает привязку данных через MVC и MVP.

### **4. gRPC**

gRPC — это абстрактная спецификация. Она описывает абстрактную RPC (remote procedure call), то есть удалённый вызов процедуры, которая обладает определёнными свойствами.

Первое свойство — поддержка как одиночных вызовов, так и стриминга. То есть все сервисы, которые реализовывают эту спеку, поддерживают оба варианта. Следующий пункт — наличие метаданных, то есть чтобы вместе с полезной нагрузкой вы могли бы передать какие-то метаданные — условно, заголовки. И ещё поддерживается отмена запроса и таймаутов из коробки.

Также она предполагает, что описание сообщений и самих сервисов осуществляется через некий Interface Definition Language или IDL. Также спецификация описывает wire-протокол поверх HTTP/2, то есть gRPC предполагает работу только поверх HTTP/2.

5. Play

Play — это среда Java, предназначенная для веба и мобильных приложений. Основное применение — в приложениях, требующих большого масштабирования. Платформа также совместима с рядом языков помимо Java, такими как Python и Ruby. Play не основан на корпоративных стандартах Java. Обычно он используется для веб-приложений на основе создания контента, а также для приложений Scala.

Фреймворк Play в первую очередь известен тем, что он удобен для пользователей, даже для новичков. Он позволяет соглашаться с конфигурацией, а также может отображать сообщения об ошибках в самом браузере. Он особенно эффективен в высокопроизводительных приложениях благодаря поддержке неблокирующего ввода-вывода. Это, безусловно, одна из наиболее отказоустойчивых структур, которая также демонстрирует большую гибкость.

Play — это нетрадиционный и уникальный тип фреймворка, который следует подходу соглашения, а не конфигурации. Он основан на шаблоне MVC и представляет собой платформу веб-приложений с открытым исходным кодом.

Play похож на Django, Ruby on Rails или архитектуру ASP.NET и не обязательно соответствует веб-стандартам J2EE.

Особенности Play:

* высокая производительность за счёт асинхронной обработки;
* без контейнера, без состояний и построенный на реактивных принципах;
* использует статически типизированный язык, поэтому большинство ошибок обнаруживается во время компиляции, что позволяет избежать множества ошибок на ранних этапах жизненного цикла разработки;
* Scala исследует истинное ООП вместе с некоторыми концепциями функционального программирования. Его совместимость с Java делает систему отличной и мощной;
* вместе с Play2 была выпущена новая мощная система сборки sbt, которая упрощает интеграцию с проектами Maven, а также создание простых файлов JAR;
* встроены обширные библиотеки доступа к реляционным базам данных для общих функций.

Play написан с использованием языка программирования Scala. Предлагает разработку веб-приложений и мобильных приложений. Он следует архитектуре MVC. Play скомпилирован в Java bytecode, и это делает его одним из самых мощных фреймворков.

Некоторые преимущества использования Play Framework:

* мощные инструменты для консоли и сборки;
* встроенные инструменты тестирования;
* поддержка IDE для Eclipse и IntelliJ IDEA;
* гибкость и отказоустойчивость;
* поддерживает неблокирующие операции ввода-вывода;
* помогает в создании высокопроизводительных приложений;
* поддерживает «горячее кодирование» и «соглашение по конфигурации»;
* может выдавать сообщения об ошибках.

### **6. MapStruct**

Эта библиотека генерирует код, который будет автоматически переделывать один класс в другой. Это невероятно ускоряет разработку и тестирование кода.

### **7. Lombok**

Эта библиотека предназначена для генерации кода. Она способна генерировать код для достаточно популярных use-cases. Стоит лишь поставить необходимые аннотации.

Достаточно спорная библиотека. Кто-то её любит, кто-то ненавидит, но никого она не оставляет равнодушным.

### **8. Apache POI**

На основе этой библиотеки можно создать внутренний фреймворк для удобной и гибкой генерации отчётов. Библиотека поддерживает различные форматы документов, например Word и Excel.

### **9. Liquibase/ FlyWay**

Средства для миграции баз данных. В современном мире практически не существует проекта, который не будет использовать автоматическое исполнение скриптов для баз данных. Каждая из этих библиотек обладает достоинствами и недостатками. Прежде чем остановить свой выбор на одной из них, я крайне рекомендую ознакомиться с документацией на официальном сайте и принять взвешенное решение. Но даже если спустя время вы поймёте, что решение было ошибочным, эти библиотеки предоставляют достаточно безболезненную миграцию друг на друга.

Java-фреймворки и библиотеки значительно сокращают время программирования и предоставляют разработчикам такие важные функции, как безопасность, эффективность и производительность.

1. **Сравнение Java, Python и C**

Главные конкуренты Java по популярности — C и Python. Давайте сравним с ними Java по нескольким критериям: типу языка, скорости, гибкости и простоте изучения.

**Тип языка**

Мы уже разобрались, в чём разница между интерпретируемыми и компилируемыми языками, теперь можем сравнить.

Всего существуют четыре грейда:

| **Python** |  | **Java, C** |
| --- | --- | --- |
| Интерпретируемый |  | Несколько месяцев |

**Скорость**

Java быстрее Python, потому что код на Python процессор обрабатывает построчно: читает каждую строчку и только потом выдаёт результат. Но медленнее C, потому что C компилируется в язык процессора и не выполняет двойную работу, как Java, код которой сначала компилируется в Java-машину, а потом в язык процессора.

| **Python** | **Java** | **C** |
| --- | --- | --- |
| Медленный |  | Быстрый |

**Гибкость**

Java пишется один раз и сразу под все платформы. Python нужно оптимизировать, но все интерпретируемые языки — гибкие и их оптимизировать легче, чем компилируемые. А вот C нужно переписывать под каждую платформу, он не гибкий совсем.

| **Java** | **Python** | **C** |
| --- | --- | --- |
| Гибкий |  | Не гибкий |
|  |  |  |

**Порог входа**

C — самый сложный для входа в программирование. Он тяжёлый, потому что разработан давно и у него большая история. Python считается самым лёгким языком для входа, потому что самый молодой. Java где-то посередине.

| **Python** | **Java** | **C** |
| --- | --- | --- |
| Простой |  | Сложный |
|  |  |  |

Всё детское программирование построено на изучении Python. Java тоже подходит для начинающих, не сильно сложнее Python, но код длиннее. Там, где Python нужна одна строчка, Java хочет пять.

| **Python** | **Java** |
| --- | --- |
| print («Hello, World!») | class HelloWorld { public static void main (String[] args) { System.out.println («Hello World!»); } } |

1. **Сферы применения языка Java**

По данным компании Oracle, программы на Java запускаются на 3 млрд девайсов. Это маркетинговое сообщение сложно проверить. Тем не менее Java широко используется и входит в число самых востребованных языков, это не вызывает сомнения.

Например, подавляющее большинство крупных компаний так или иначе используют Java. Очень много серверных приложений для корпораций написаны на этом языке. Например, речь идёт о программах для финансовых организаций, которые обеспечивают проведение транзакций, фиксацию торговых операций.

На Java написано много веб-приложений. Популярные фреймворки, в том числе Spring, Stuts, JSP, используются для создания разных приложений в вебе: от ecommerce-проектов до крупных порталов, от образовательных платформ до правительственных ресурсов.

Популярная компьютерная игра Minecraft написана на Java.

Мобильная разработка — ещё одна область использования Java. На этом языке пишут приложения для устройств, работающих под управлением ОС Android.

На Java создают клиентские приложения. Простой и близкий разработчикам пример: IDE NetBeans написано на «джаве».

Также Java применяется для работы с Big Data, разработки программ для научных целей, например, обработки естественных языков, программирования приборов — от бытовых девайсов до промышленных установок.

То есть на Java можно писать разные типы приложений: веб, мобильный и десктопный софт, игры и так далее. Традиционно у этого языка сильные позиции в промышленном программировании, в сегменте крупных компаний (т.н. энтерпрайз).

Итог: Java — язык программирования общего назначения. Имеет си-подобный синтаксис. Используется для создания приложений в разных областях: от веба до разработки игр, от мобильного ПО до программ для корпораций и научных институтов.

**Заключение**

Java на данный момент остается достаточно популярным языком, и в ближайшем будущем будет также популярен. Каждые полгода выходят новые версии языка, крупные сообщества живут полной жизнью, а разделы помощи на официальном сайте регулярно обновляются в соответствии с популярными вопросами от пользователей.

Новые языки программирования — быстрые, комфортные в применении и просты в освоении. Более молодые языки, такие как Scala (2001 г) или Kotlin (2011 г) сегодня могут позволить себе конкурировать с Java в мобильной разработке — и кстати оба компилируются в JVM — и имеют немного больше различного «синтаксического сахара». Но в основной своей массе молодые языки ориентируются на решение узких задач.

В отличие от Java, который применяется почти во всех областях прикладной разработки. Java обладает колоссальным количеством фреймворков и библиотек, с помощью которых можно решить многие задачи: от простейшей игры «змейки» до разработки корпоративных приложений.

Знания и владение в Java открывает для специалиста множество перспектив в прикладном программировании. Зная Java, программист может с легкостью начать изучать другие языки программирования.

**Список литературы**

1. <https://www.interestprograms.ru/article-istoriya-sozdaniya-yazyka-programmirovaniya-java>
2. <https://oracle-patches.com/coding/отличительные-особенности-языка-java>
3. <https://practicum.yandex.ru/blog/chto-takoe-java-dlya-nachinayuschih/>
4. <https://ru.wikipedia.org/wiki/Java>
5. <https://ru.bmstu.wiki/Java>
6. <http://progopedia.ru/language/java/>
7. https://vertex-academy.com/tutorials/ru/biblioteka-java/