# Characteristics of Java

**Java Programming language** has static typing. All variables must be declared first before

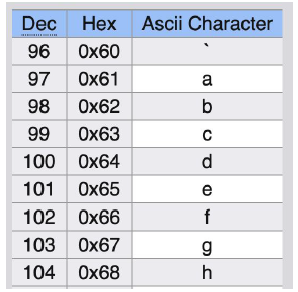
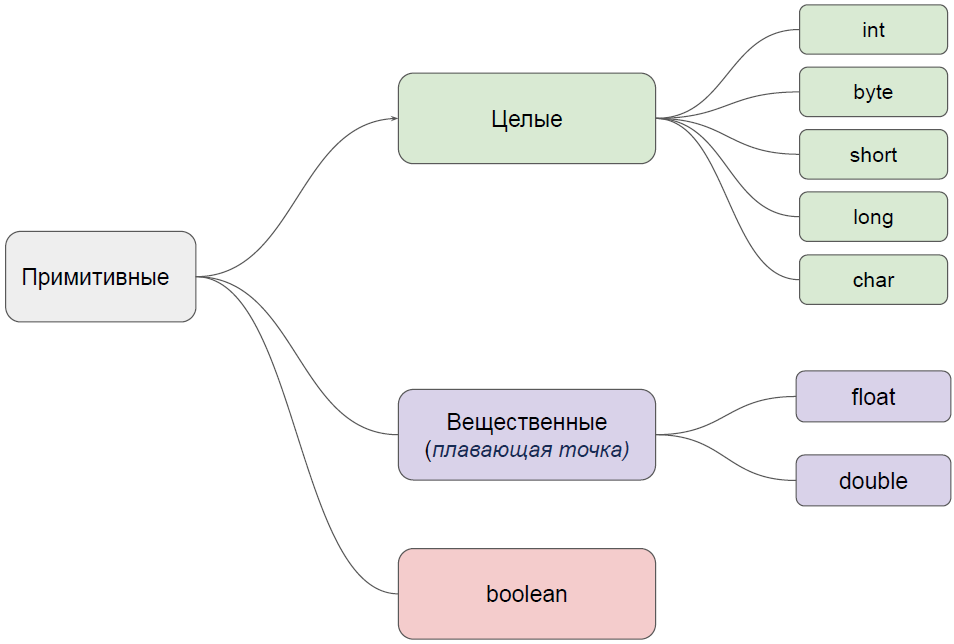
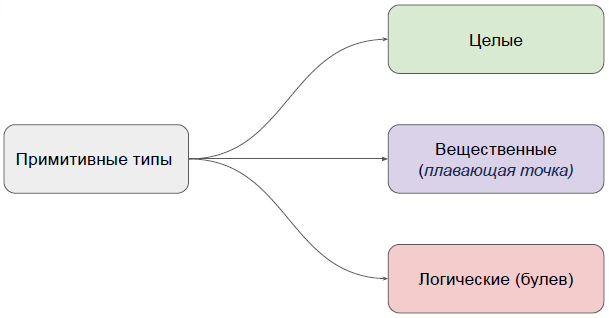
they can be used.

Java has:

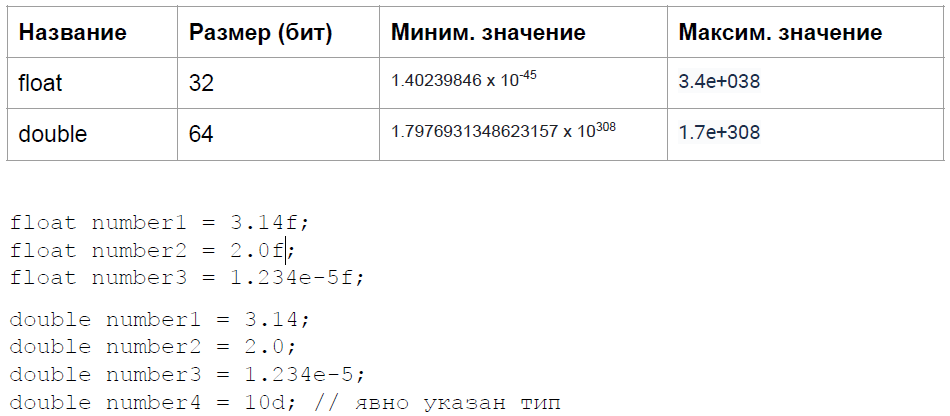
* lowercase style.
* camelCase.
* UPPERCASE.

Variables are usually named in lowerCamelCase style.

Java Programming language has **primitive** and **reference** types.



# Целые типы и вещественные типы

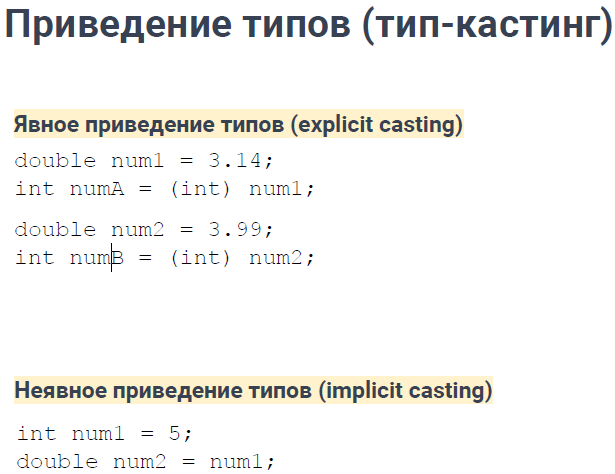


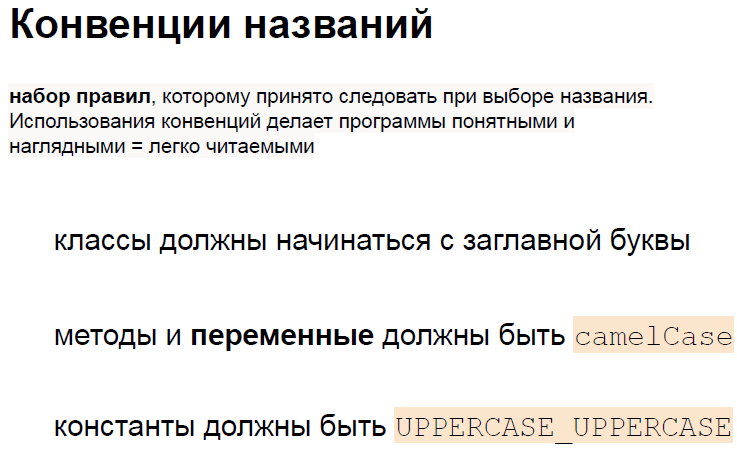
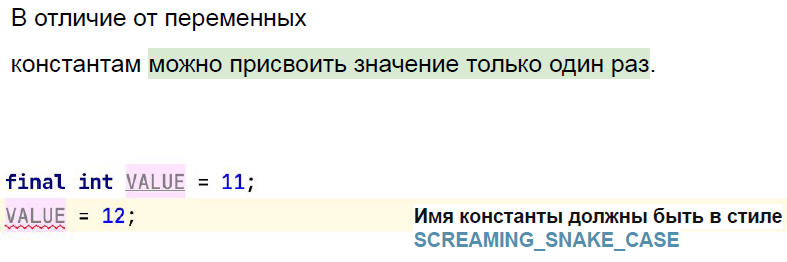
# Main types of autotests

1. **Unit** (быстрые, максимально изолированы, самое большое количество ,тестируют одну функцию)- эти тесты предназначены для того, чтобы убедиться, что ваш код работает так, как вы ожидаете. Создают в основном разработчики.
2. **Integration** (частично изолированы, покрывают процессы взаимодействия) - с помощью интеграционных тестов вы гарантируете, что несколько частей вашего программного обеспечения взаимодействуют друг с другом должным образом. Создаются как правило тестировщиками, иногда разработчиками.
3. **UI** (End to End, Acceptance, иметируют поведение пользователя, требуют настройки окружения, должны быть обязательно, потому что они дают уверенность в том, что сценарии пользователя работают так как ожидается) - рассматривают все приложение целиком. Разрабатываются тестировщиками-автоматизаторами.
4. **Performance** **тесты** особенно важны для продуктов, где стабильность и скорость являются ключевыми требованиями.

# Testing Pyramid Concept and Testing Pyramid

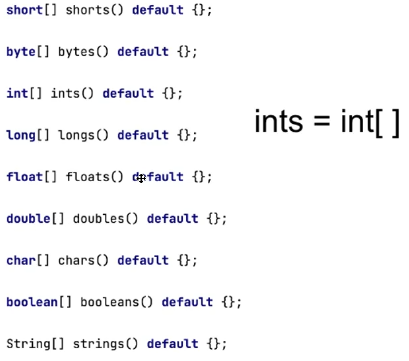
Конццепция, котрая описывает разный уровень тестов.





<https://www.oracle.com/java/technologies/javase/codeconventions-comments.html>

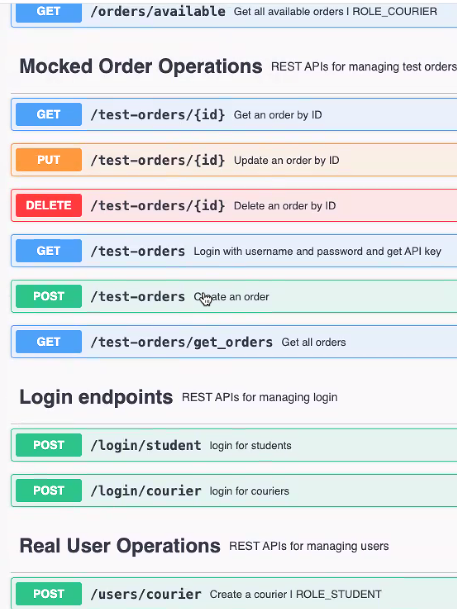
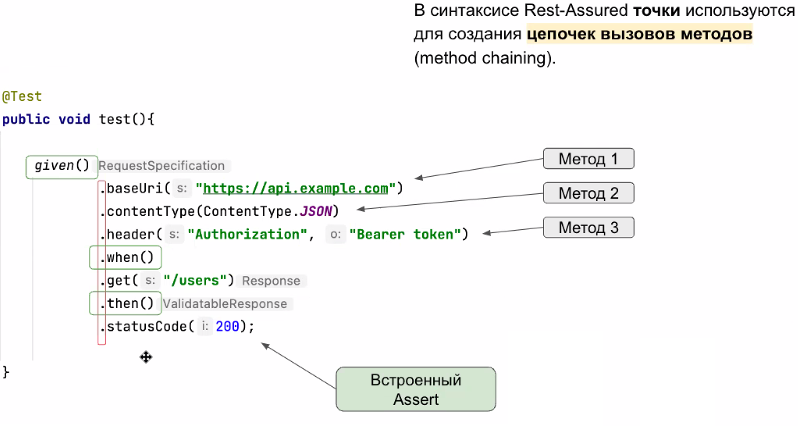
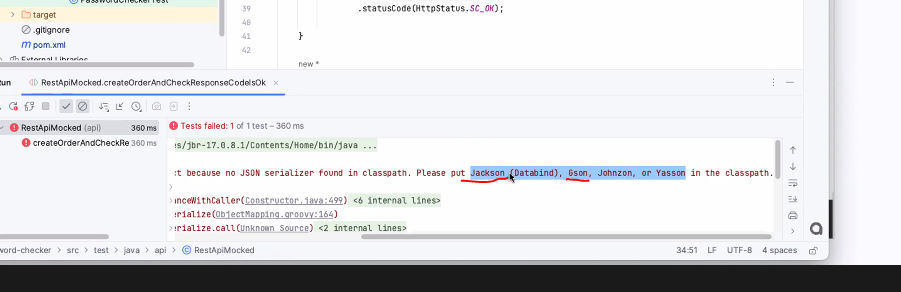
Параметризованный тест – типы данных



# Vocabulary

1. **Автоматизация** - это процесс создания, использования и поддержки инструментов, которые упрощают работу тестировщика и повышают качество тестирования.
2. **Автотесты** - это инструмент.
3. **Непрерывная интеграция (Continuous Integration) -** это практика разработки программного обеспечения, при которой разработчики регулярно объединяют свой код в общее хранилище, после чего автоматически запускаются сборка и тестирование этого кода. Это позволяет обнаруживать и исправлять интеграционные проблемы быстрее.
4. **Непрерывная доставка (Continuous Delivery) -** это дополнение к непрерывной интеграции, при котором каждое успешное изменение в коде проходит через автоматизированный процесс сборки, тестирования и развертывания в целевой среде. Цель состоит в том, чтобы код всегда был готов к развертыванию в производственной среде.
5. **Клиент** отвечает за взаимодействие с пользователем ↔ **сервер** отвечает за логические операции, вычисления ↔ **База** **данных** отвечает за хранение и обработку данных.
6. **Константы** - переменные, значения которых нельзя изменять
7. **Сhar** - сохраняет один символ/букву или значения ASCII.
8. **Boolean** true (истина) или false (ложь) (1 bit size)
9. **Int** - целые числа, можно выполнять различные операции: сложение, вычитание, умножение, деление и другие.
10. **Сhecklist** - in testing is a list of steps or criteria that helps systematize the process of checking software.
11. **JUnit** - это фреймворк для тестирования Java-приложений. Он облегчает написание и запуск тестовых сценариев, автоматизируя проверку корректности работы кода, что позволяет разработчикам быстрее обнаруживать и исправлять ошибки.  
    Основные преимущества: простота использования, автоматизация тестирования, интеграция с средами разработки, поддержка аннотаций, отчеты о выполнении тестов.

**Аннотации:**@Test: Обозначает метод как тестовый.  
@Before: Метод, выполняющийся перед каждым тестом.  
@After: Метод, выполняющийся после каждого теста.  
@BeforeClass (JUnit 4): Статический метод, выполняющийся перед всеми тестами в классе.  
@AfterClass (JUnit 4): Статический метод, выполняющийся после всех тестов в классе.  
  
Порядок теста не зависит друг от друга. Нельзя делать между ними связку для тестирования, т.к они все индивидуальные.

1. **Maven** - это инструмент для управления проектами и сборки программного обеспечения в Java. Он позволяет автоматизировать процесс сборки, тестирования, документирования и управления зависимостями в проекте. Maven использует файлы pom.xml для описания проекта и его зависимостей, что делает процесс сборки более простым и структурированным. Maven облегчает управление проектом и его зависимостями, а также предоставляет средства для автоматизации рутинных задач разработки.
2. **XML (Extensible Markup Language) -** это язык разметки для представления структурированных данных в текстовом формате. Он используется для хранения и передачи информации между компьютерными системами. XML состоит из пользовательских тегов, которые определяют структуру и содержание данных. Этот формат данных позволяет легко читать и обрабатывать информацию как людям, так и компьютерам.
3. **YML** – это удобочитаемый формат сериализации данных. Он используется для представления информации в виде структурированных текстовых файлов с использованием отступов для обозначения иерархии данных. YAML часто применяется для конфигурационных файлов, метаданных и других целей, где важны удобочитаемость человеком и простота использования.
4. **GitHub** - это веб-сервис для хостинга репозиториев Git, который обеспечивает совместную работу над проектами с использованием системы контроля версий Git. Он позволяет разработчикам хранить свой код, отслеживать изменения, совершать коммиты, создавать ветви для разработки новых функций или исправлений, а также сливать изменения обратно в основную ветвь проекта. GitHub также предоставляет инструменты для обсуждения кода, управления задачами и интеграции с другими сервисами разработки программного обеспечения. Кратко говоря, GitHub - это платформа для совместной разработки и хранения кода.Bamboo
5. **Bamboo** - это инструмент для непрерывной интеграции и доставки (CI/CD), разработанный компанией Atlassian. Он предназначен для автоматизации процессов сборки, тестирования и развертывания программного обеспечения. Bamboo позволяет разработчикам создавать, запускать и отслеживать сборки проектов, а также интегрировать их с различными системами управления версиями кода, такими как Git, Mercurial или SVN. В контексте тестирования, Bamboo может использоваться для автоматизации запуска тестов, отслеживания их результатов, а также уведомления команды о результатах тестирования. Он также может интегрироваться с другими инструментами тестирования и системами управления проектами для обеспечения непрерывной интеграции и доставки качественного программного обеспечения.
6. **Pipeline** - представляет собой автоматизированный процесс выполнения определенных шагов или этапов тестирования программного обеспечения, начиная с сборки приложения и заканчивая его развертыванием или выпуском.   
     
   Пайплайн обычно включает в себя следующие основные этапы:   
   - Сборка (**Build**): Создание исполняемой версии приложения из исходного кода.  
   - Тестирование (**Testing**): Запуск автоматизированных тестов для проверки работоспособности и качества приложения. Этот этап может включать модульные тесты, интеграционные тесты, системные тесты и т. д.  
   - Анализ (**Analysis**): Анализ результатов тестирования и отображение отчетов о покрытии кода, статическом анализе, выявлении ошибок и т. д.  
   - Развертывание (**Deployment**): Автоматизированное развертывание приложения на тестовых, предпродакшн или продакшн серверах.  
   - Уведомления (**Notifications**): Уведомление членов команды о результатах выполнения пайплайна (например, успешное или неудачное выполнение тестов). Пайплайны помогают создать структурированный и повторяемый процесс тестирования и развертывания программного обеспечения, что позволяет быстро выявлять и исправлять ошибки, ускоряя цикл разработки и повышая качество продукта.
7. **Regex** **(регулярные выражения**) - это последовательности символов, которые используются для поиска и сопоставления шаблонов в строках текста. Они предоставляют мощный и гибкий способ выполнения поиска и замены текста на основе определенных шаблонов.  
   Регулярные выражения могут использоваться для:  
   - Поиска определенных строк или паттернов в тексте.  
   - Валидации ввода данных пользователем (например, проверка формата электронной почты или номера телефона).  
   - Разделения текста на подстроки согласно определенным правилам.  
   - Замены определенных паттернов на другие строки.  
     
   Например, регулярное выражение \d{3}-\d{2}-\d{4} соответствует строке, содержащей номер социального страхования в формате XXX-XX-XXXX, где X - это цифры.  
     
   Regex являются мощным инструментом для обработки и анализа текстовых данных и широко используются в программировании, анализе данных, обработке текстов и других областях информатики.
8. **REST API (Representational State Transfer Application Programming Interface)** - это структура для разработки веб-сервисов, которая следует принципам **REST**, таким как использование **HTTP**-методов (**GET**, **POST**, **PUT**, **DELETE**) для доступа к ресурсам и передачи данных. REST API позволяет клиентам взаимодействовать с сервером посредством стандартных **HTTP**-запросов, обычно в формате **JSON** или **XML**. Он предоставляет унифицированный интерфейс для доступа к данным и функциям удаленных систем. **REST API** широко используется в веб-разработке для создания веб-сервисов, мобильных приложений и интеграции различных приложений и систем.  
     
   **REST** **запросы** относятся к запросам, отправляемым к веб-сервису, который следует архитектурному стилю **REST** (Representational State Transfer). Этот стиль определяет правила для создания веб-сервисов с использованием стандартных протоколов **HTTP**. **REST** запросы могут быть выполнены с использованием различных методов HTTP, таких как **GET**, **POST**, **PUT**, **DELETE** и другие. Они предназначены для выполнения операций над ресурсами, которые идентифицируются уникальными **URL**-адресами.   
     
   **API запросы (Application Programming Interface)** - это запросы, отправляемые к программному интерфейсу приложения. **API** запросы могут быть разного типа, в зависимости от того, какие действия должны быть выполнены. Например, они могут запрашивать данные из базы данных, отправлять данные на сервер, обновлять информацию и т. д. API запросы могут быть отправлены к любому доступному **API**, включая **REST** **API**, **SOAP** **API**, **GraphQL** **API** и другие. Они обычно отправляются с использованием стандартных протоколов, таких как **HTTP** или **HTTPS**.
9. **SOAP (Simple Object Access Protocol)** - это протокол обмена структурированными данными в распределенной архитектуре приложений. **SOAP** запросы представляют собой сообщения, которые отправляются между клиентом и сервером, используя **XML** для описания структуры данных внутри сообщения. **SOAP** запросы часто используются для вызова удаленных процедур (**RPC**) и передачи сообщений между веб-сервисами. **SOAP** запросы обычно отправляются с использованием протокола **HTTP** или **HTTPS**, но **SOAP** также может работать поверх других протоколов, таких как SMTP (для электронной почты) или **JMS (Java Message Service).** Ключевая особенность SOAP - это его формализованный подход к определению сообщений и их структуры с помощью спецификации WSDL (Web Services Description Language), что делает его более строгим и менее гибким по сравнению с другими протоколами, такими как REST.
10. **JSON (JavaScript Object Notation)** - это легковесный формат обмена данными, основанный на синтаксисе объектов **JavaScript**. **JSON** используется для передачи структурированных данных между клиентом и сервером по сети. Он представляет собой набор пар "ключ-значение" и поддерживает различные типы данных, такие как строки, числа, логические значения, массивы и объекты. **JSON** легко читаем и понятен как человеком, так и компьютером, что делает его популярным выбором для обмена данными в веб-разработке и **API**.
11. **CURL** - это утилита командной строки, используемая для отправки и получения данных по **URL**-адресам с использованием различных протоколов, таких как HTTP, **HTTPS**, **FTP** и многих других. Она поддерживает множество опций и параметров для настройки запросов, включая методы запросов, заголовки, данные формы и аутентификацию. CURL широко используется в скриптах, автоматизированных задачах, тестировании API и взаимодействии с веб-сервисами из командной строки.
12. **Postman -** это платформа для разработки, тестирования и документирования **API**. Он предоставляет интуитивно понятный интерфейс для создания и отправки **HTTP**-запросов различных типов (**GET**, **POST**, **PUT**, **DELETE**, **HEAD**, **PATCH**, **OPTIONS**) ,а также для проверки ответов от сервера. Postman также обладает возможностями автоматизации тестирования API и создания коллекций запросов для повторного использования или передачи другим членам команды.
13. **REST-assured** – это фреймворк для автоматизированного тестирования **REST API** на языке Java. Он облегчает написание автотестов для проверки различных аспектов работы **API**, таких как проверка ответов на запросы, верификация статус кодов, работы с заголовками и телами запросов и ответов. **REST-assured** позволяет создавать читаемые и поддерживаемые тесты, что делает процесс тестирования **API** более эффективным и надежным. (**GIVEN**, **WHHEN**, **THEN**)  
    
14. Популярные библиотеки  
    
15. **Gson** – нужен для сериализации и десериализации. "Сериализация" и "десериализация" - это процессы преобразования данных в формат, который может быть легко сохранен или передан через сеть, и обратно в их исходное состояние. Например, вы можете сериализовать данные в формат JSON для сохранения на диске или передачи по сети, а затем десериализовать их обратно в программе для восстановления исходной структуры данных. Эти процессы особенно полезны при работе с сетевым взаимодействием или сохранении состояния приложений.
16. **Парсинг** - это процесс анализа строки или потока данных с целью извлечения нужной информации или преобразования её в другой формат данных. В отличие от сериализации и десериализации, которые обычно связаны с преобразованием данных в формат, пригодный для сохранения или передачи, парсинг фокусируется на извлечении конкретной информации из исходных данных.  
    Например, парсинг HTML-документа может включать извлечение текста, ссылок или других элементов разметки. Парсинг JSON может включать извлечение конкретных значений из JSON-объекта. Таким образом, парсинг часто используется для обработки данных в различных форматах с целью работы с ними в программе или системе.
17. **DTO** - **Data Transfer Object** это обычно простой объект, который используется для передачи данных между подсистемами или слоями программного обеспечения. Он может содержать поля для хранения данных и методы для доступа к этим данным. Обычно DTO используется для передачи данных между клиентом и сервером в распределенных приложениях, где данные должны быть упакованы в объекты для передачи по сети. Использование DTO может помочь упростить обмен данными и сделать код более чистым и модульным.
18. **В Java, массив (array)** - это структура данных, которая хранит фиксированное количество элементов одного типа данных. Он представляет собой упорядоченную коллекцию элементов, где каждый элемент имеет свой индекс. Массивы в Java могут быть одномерными и многомерными.  
      
    **Пример одномерного массива в Java:**  
    int[] numbers = new int[5]; // Объявление массива целых чисел длиной 5  
      
    В этом примере **numbers** - это массив целых чисел, который может содержать 5 элементов. Индексы элементов массива начинаются с 0, так что первый элемент массива будет **numbers[0]**, а последний - **numbers[4]**.

**Пример многомерного массива в Java:**  
int[][] matrix = new int[3][3]; // Объявление двумерного массива целых чисел размером 3x3

В этом примере **matrix** - это двумерный массив целых чисел, который представляет собой матрицу 3x3. Для доступа к элементу в многомерном массиве используются индексы для каждого измерения, например, **matrix[0][0]** - это первый элемент матрицы, а **matrix[2][2]** - последний элемент.  
  
Массивы в Java являются объектами, поэтому они наследуют методы от класса **Object**, такие как **toString()**, **equals()**, и другие. Однако, их размер фиксирован при создании, и его нельзя изменить после этого без создания нового массива.