# Characteristics of Java

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

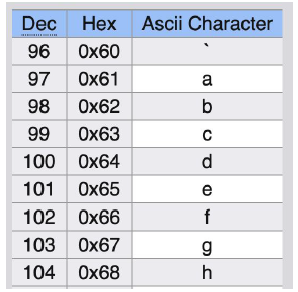
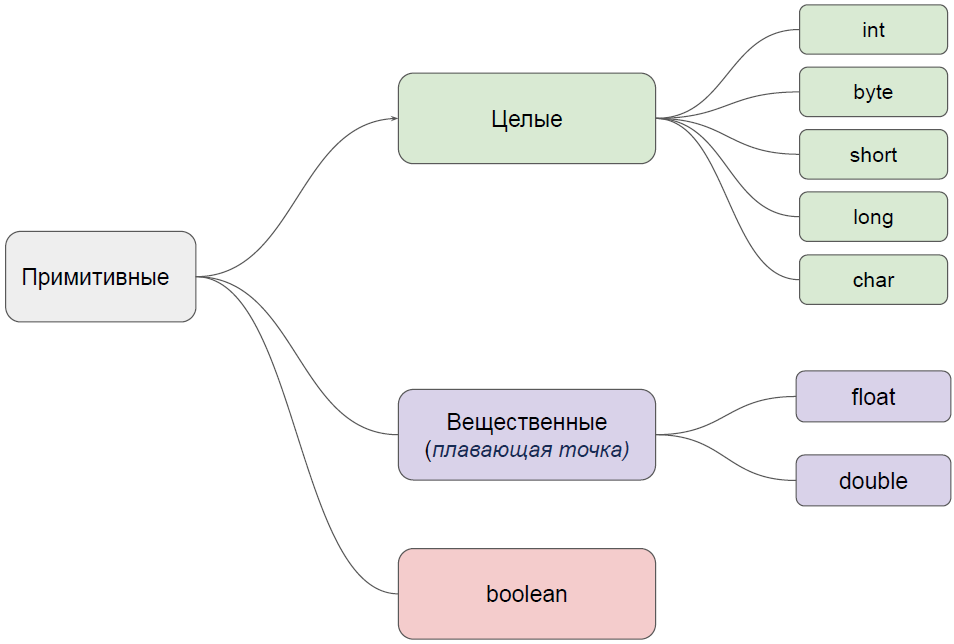
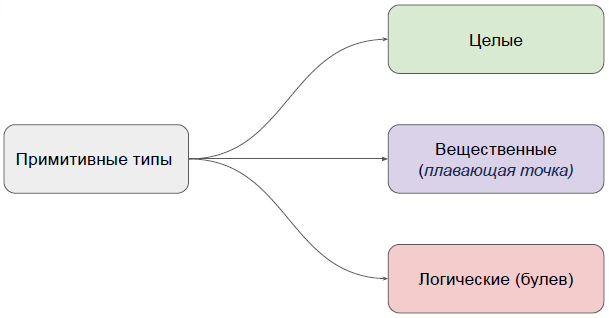
they can be used.

Java has:

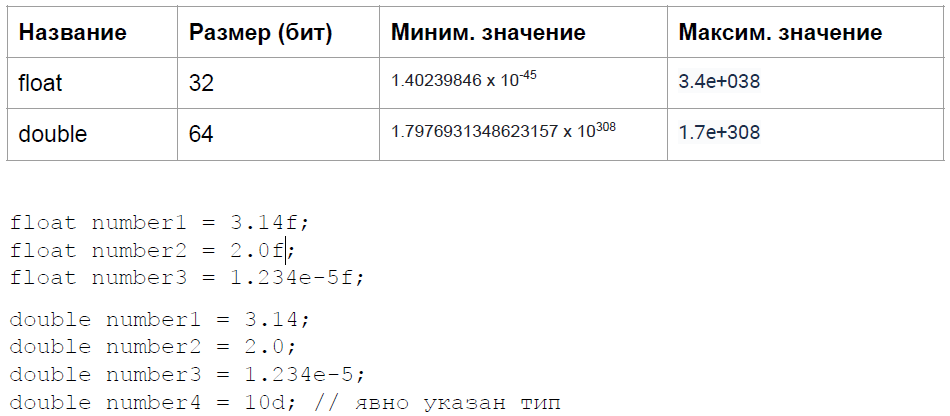
* lowercase style.
* camelCase.
* UPPERCASE.

Variables are usually named in lowerCamelCase style.

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



# Целые типы и вещественные типы

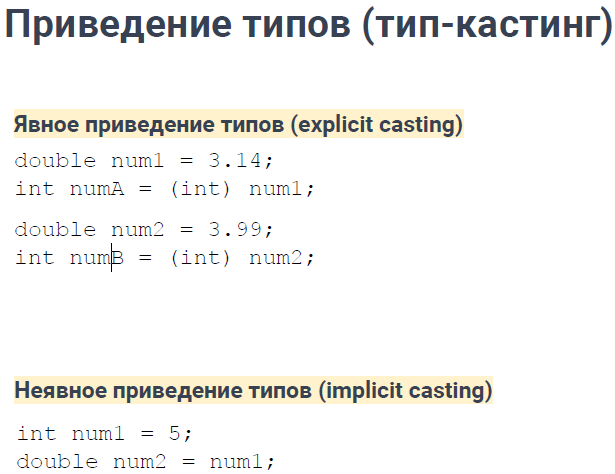


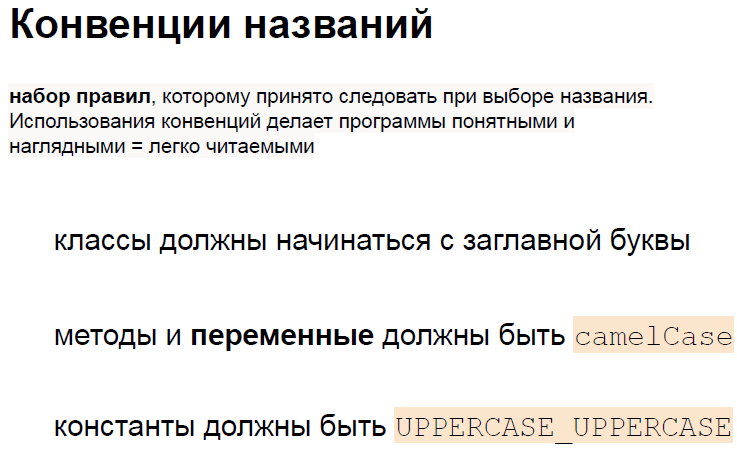
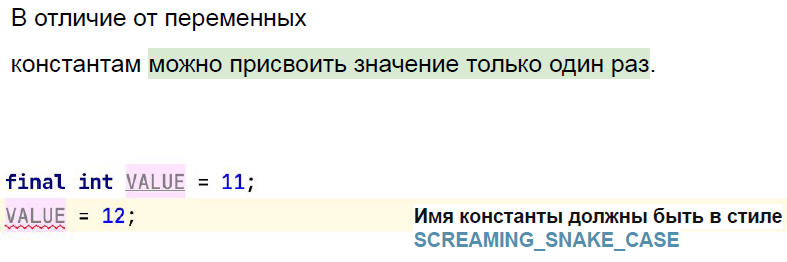
# Main types of autotests

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>

# 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.
12. **XML** –
13. **YML** –

**Аннотации:**@Test: Обозначает метод как тестовый.  
@Before: Метод, выполняющийся перед каждым тестом.  
@After: Метод, выполняющийся после каждого теста.  
@BeforeClass: Статический метод, выполняющийся перед всеми тестами в классе.  
@AfterClass: 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. **Yml**. – используется для описания конфигурации и pipeline.
3. **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
4. **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.
5. **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.
6. **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.
7. f