**Сценарий Видео: Консольная Игра "Крестики-Нолики" на Java (Tic-Tac-Toe)**

---

**ЧАСТЬ 1: ВВЕДЕНИЕ (3 минуты)** 🕹️

- "Привет, друзья! Сегодня мы погрузимся в мир классической логики и программирования, создав с нуля консольную версию культовой игры **'Крестики-нолики'** на Java!"
- "Этот проект — идеальный способ закрепить знания по работе с **двумерными массивами**, отточить логическое мышление при **проверке условий победы** (строки, столбцы, диагонали) и научиться эффективно обрабатывать **ввод пользователя**."
- Мы разработаем не просто игру 3x3, а продвинутую версию, которая включает:
    - Возможность игры на поле **любого размера** (хотя по умолчанию 3x3).
    - Специальную **пасхалку** ('JAVA'), которая активирует режим **5x5** для хардкорной игры (победа по-прежнему за 3-в-ряд).
    - Надежную логику **проверки победы** для поля N x N (3-в-ряд).
    - Аккуратный вывод поля в консоль.
    - Обработку некорректного ввода.
- "Всё, что нужно, мы напишем в одном классе Main, шаг за шагом разбирая каждую функцию!"

---

**ЧАСТЬ 2: ТЕХНОЛОГИИ И ИНСТРУМЕНТЫ (3 минуты)** 🛠️

- "Для этого проекта нам не понадобятся сторонние библиотеки. Мы будем использовать только **стандартные и нативные** возможности Java."
- Что применим:
    - **Двумерные массивы (char[][]):** Основа игры, хранящая состояние поля ('X', 'O', ' ').
    - **Scanner:** Для взаимодействия с пользователем и получения номера ячейки.
    - **Arrays.fill:** Для быстрой инициализации поля пробелами.
    - **NumberFormatException:** Для корректной обработки ввода, если пользователь вводит не число.
    - **Циклы while / for и if/else:** Для главного игрового цикла, отрисовки поля и логики проверки победы/ничьей.

- **Ключевая логика:** Мы реализуем алгоритм преобразования **номера ячейки (от 1 до $N^2$)**, который вводит пользователь, в **координаты массива (строка/столбец)**, используя целочисленное деление и операцию по модулю.

---

**ЧАСТЬ 3: ЧЕМУ ВЫ НАУЧИТЕСЬ (3 минуты)** 🧠

- "После просмотра этого видео вы не только получите готовую игру, но и освоите важнейшие концепции для любого Java-разработчика."
- Вы поймёте:
    - Как эффективно работать с **глобальным состоянием игры** через статические переменные (board, currentPlayer, boardSize).
    - Как использовать **циклы со сдвигом** (for (int i = 0; i <= boardSize - 3; i++)) для проверки паттернов (3-в-ряд) на массивах **переменного размера**.
    - Как создать логику **пасхалки/секретного режима**, изменяющего поведение программы в процессе работы.
    - Принципы **чистого рекурсивного ввода** с обработкой ошибок (return getPlayerMove(); внутри catch).
    - Как реализовать **базовую игровую логику** от хода до проверки победы и ничьей.

---

**ЧАСТЬ 4: ФУНКЦИОНАЛ ИГРЫ (3 минуты)** 🌟

- "Давайте посмотрим, каким будет финальный функционал нашего консольного шедевра:"
- Что будет у нашей программы:
    - **Размер поля:** Изначально **3x3**.
    - **Интерактивный ввод:** Игроки вводят номер ячейки (например, от 1 до 9).
    - **Логика N-в-ряд:** Победа засчитывается, если найдено **три символа подряд** по горизонтали, вертикали или диагонали, независимо от текущего размера поля.
    - **Пасхалка (Secret Mode):** Если ввести код **'JAVA'**, поле **расширяется до 5x5**, и игра начинается заново с игрока 'X'.
    - **Проверка валидности:** Программа проверяет, находится ли номер ячейки в допустимом диапазоне и не занята ли ячейка.

---

**ЧАСТЬ 5: НАПИСАНИЕ КОДА (15 минут)** 👨‍💻

- "Приступаем к написанию кода! Всё будет сопровождаться подробными комментариями и объяснениями."
- **1. Блок 1: Основная структура и Инициализация.**
    - Создаём класс Main и статические переменные для состояния игры (board, boardSize, currentPlayer, gameIsOver, secretModeActivated).
    - Реализуем метод initializeBoard(), который заполняет массив пробелами, используя Arrays.fill.
- **2. Блок 2: Отрисовка и Ввод.**
    - Пишем метод printBoard() для форматированного вывода поля в консоль с разделителями.
    - Реализуем ключевой метод getPlayerMove(), который обрабатывает строковый ввод, проверяет на код **'JAVA'**, конвертирует номер в координаты (строка/столбец) и использует try-catch для обработки ошибок ввода.
- **3. Блок 3: Логика Хода и Управление Состоянием.**
    - Пишем простой isValidMove() для проверки границ и пустоты ячейки.
    - Реализуем switchPlayer() для смены хода.
    - Добавляем activateSecretMode(), чтобы сменить boardSize на 5 и переинициализировать поле.
- **4. Блок 4: Проверка Победы (The Core Logic).**
    - Пишем checkRows(), checkColumns() и checkDiagonals(). **Основной фокус:** объяснение цикла for (int i = 0; i <= boardSize - 3; i++), который позволяет проверять 3-в-ряд в окне, движущемся по полю любого размера.
    - Завершаем логикой isBoardFull() и объединяем все проверки в checkWin().

---

**ЧАСТЬ 6: ЗАВЕРШЕНИЕ (3 минуты)** ✅

- "Подводим итоги: Мы успешно создали полноценную, расширяемую консольную игру 'Крестики-нолики'!"
- **Резюме навыков:** Мы научились гибко работать с двумерными массивами, реализовали сложную логику проверки победы на N x N поле и освоили надежную обработку пользовательского ввода.

- "Если тебе понравился этот проект, ставь лайк и подписывайся! Впереди будет ещё больше интересных и практических проектов на Java, которые помогут тебе стать настоящим разработчиком!"