10.12

Поиск и сортировка

00:00–00:42
Введение

В этом видео кратко поговорим о том, какие бывают алгоритмы поиска и при чём тут сортировка.

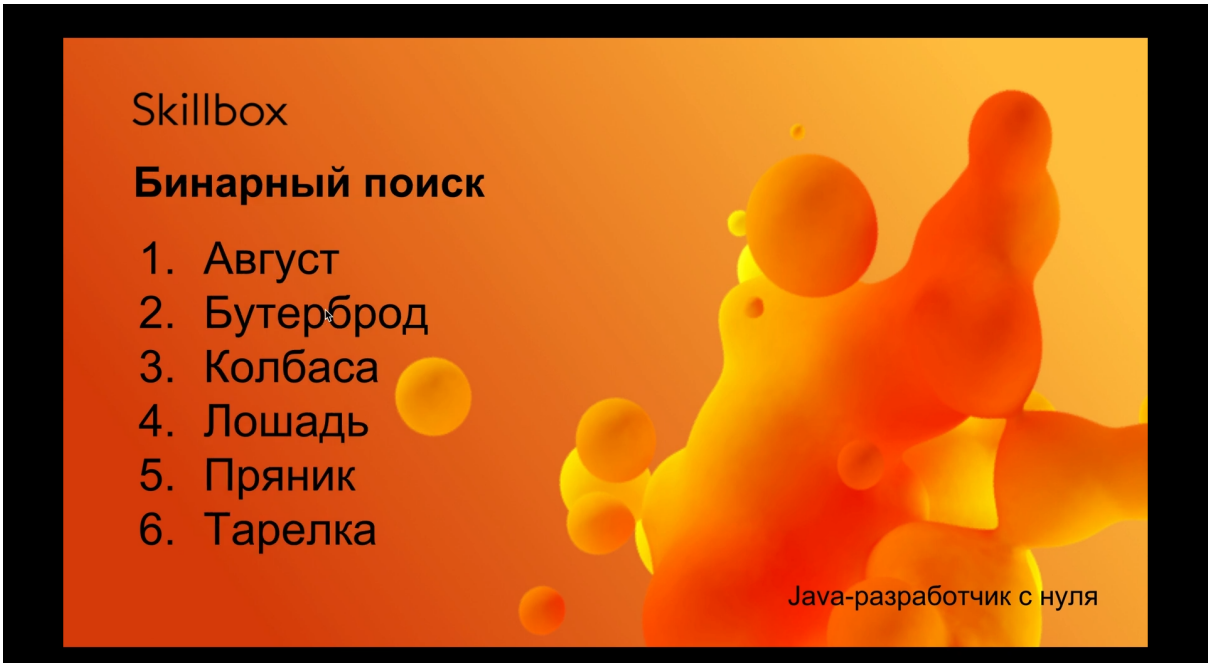
Это нужно изучить для того, чтобы вы понимали, какие структуры данных, какие коллекции, листы, сеты, хешмэпы или тримэпы использовать, если у вас много данных или вы хотите, чтобы ваша программа работала оптимально и максимально быстро.

00:42–01:26
Поиск в списке

Представим, что у вас есть список неких элементов. Сколько времени нужно, чтобы найти в этом списке конкретный элемент?
Мы не знаем, как они располагаются в списке и сколько их в нём — допустим, их два миллиона. Чтобы найти нужный элемент, вам нужно фактически обойти этот лист от начала до конца, сравнить каждый элемент с искомым и в конце выяснить, нашли вы его или нет. У ArrayList есть метод contains, который позволяет это сделать, но он делает это долго — он проверяет весь лист и сравнивает элемент с каждым элементом листа.

01:26–03:42
Бинарный поиск

Теперь представьте, что у вас есть словарь и что все элементы в нём упорядочены, допустим, по алфавиту.



Как найти необходимый нам элемент? Обычно если у нас в руках словарь, мы его открываем с середины и думаем, в какой половине находится нужное нам слово — в левой или в правой? Допустим, мы ищем слово «**лошадь**». Мы понимаем, что открыли где-то на «**колбасе**», и знаем, что нужное нам слово достаточно близко.

Как выглядит этот алгоритм? Он называется «**бинарный поиск**» и очень часто используется для упрощения поиска по отсортированным данным.
Мы разбиваем имеющееся множество пополам и смотрим, в какой половине находится нужный нам элемент. Видим, что он, допустим, находится в правой половине.
Правую половину также разбиваем пополам: если в правой половине оказалось нечётное количество значений, то середина находится не между элементами, а на каком-то конкретном элементе, и именно с ним мы сравниваем то, что ищем. Если элемент совпал — значит, элемент найден.
Если мы видим, что нужный нам элемент меньше центрового значения, то смотрим в левой части. Если больше — в правой.
В нашем случае слово «**лошадь**» меньше слова «**пряник**», значит, искать надо в левой части.

И когда мы берём левую часть, остаётся всего один элемент, мы его сравниваем с искомым и получаем то, что искали. В данном случае проводим всего три операции сравнения.

03:42–04:14
Поиск по хешу

Представьте, что вы знаете номер страницы, на которой находится слово «лошадь», допустим, оно на четвёртой странице. Вы открываете четвёртую страницу, видите нужное вам слово, перебрав 3-4 варианта, которые есть на этой странице.
Примерно так работает поиск по хешу, то есть HashSet и HashMap.

04:14
Класс Collections

Что касается сортировки бинарного поиска, то для коллекций существует специальный класс Collections, в котором есть метод sort — позволяет отсортировать ArrayList-ы. Этот метод ничего не возвращает, а сразу меняет лист. После того как вы отсортировали ваш лист, можете воспользоваться бинарным поиском, который будет возвращать целое число, то есть индекс элемента, и на ход он получает ту коллекцию, которую вы ищите, и искомую строку.
Если искомая строка не найдена, то бинарный поиск даст результат (−1).