**Задача 5-3.**

**Алгоритмы сортировки массивов.**

**Цели задания:**

1. Изучить популярные алгоритмы сортировок
2. Продолжить тренировки в использовании пакетов, классов, методов
3. Повторить использование класса **java.util.Random**
4. Использовать класс java.util.Date
5. Попробовать запись в файл

**Описание:**

Когда вы закончите обучение (а, может, и раньше), вы обязательно пойдете на свои первые собеседования. А на собеседованиях этих, почти так же обязательно, вас спросят про то, какие алгоритмы сортировки вы знаете.

Могут и не спросить, конечно, но если вы – Junior Developer, то собеседующему нужно определить, знакомы вы с базовыми алгоритмами или нет. А самый базовый из всех алгоритмов – сортировка. Так что я на 60-70% могу гарантировать, что спросят.

Вопрос этот, про сортировки, очень коварен. Потому что, как только вы ответите – «Да», вас тут же заставят писать на листике сортировку пузырьком ☺. И дальше развитие вашего собеседования пойдет по одному из трех сценариев:

1. На вопрос – «знакомы ли вы с алгоритмами сортировки?» вы отвечаете – «Нет». После этого собеседование пойдет заметно хуже, так как у вашего собеседника стоит пунктик на алгоритмы, а вы ими не владеете. Тратить на вас бесценное время становится немного жаль.
2. Вы отвечаете да на вопрос, но не можете написать пузырек. Это – самый худший вариант, потому что вы не только алгоритмами не владеете, но еще и на первый вопрос соврали. «Незнающего можно обучить, лжеца не исправить никогда» – после такой, или похожей псевдо-восточной мудрости с вами могут распрощаться ☺
3. Вы отвечаете «Да» на вопрос, и пишете пузырек! Но…

Но, как ни странно, третий вариант тоже не самый хороший. Дело в том, что если вы ограничитесь только пузырьком, то у собеседующего возникнет дилемма:

*«С одной стороны, я хотел от него алгоритмов, и он мне их дал… А с другой стороны… Блин, ну кто сегодня не напишет пузырек?! Да любой его напишет! Что же мне с ним делать? С одной стороны, он может быть хорошим джуном, а с другой стороны – пузырек может оказаться единственным, что он знал, и ему просто повезло… Что же делать, что же делать? Вывод очевиден – вздрючу его QuickSort-ом!»*

В общем, глупости все это. Не будет у вас таких наркоманских собеседующих. Просто выбросьте из головы предыдущий лист. Задание начинается здесь ☺.

Но, если к вам все-таки применят эти садистские приемы и заставят писать сортировку на листе, вы должны мстить! Немедленно и безжалостно! Спускать такое в век IDE – недопустимо! Поэтому, как только он протянет вам листик, говорите – «Конечно я напишу пузырек. Только дайте мне сразу пять листов. Я вам еще пару аналогов набросаю». А про себя думайте:

*«Пузырек тебе, падла? Будет тебе пузырек! Сейчас тебе столько этих if (a[i] < a[j]) прилетит, что ты потом еще пол ночи будешь свои носки по цветам раскладывать, чтобы отпустило!..»*

Да блин! Вы тоже так отвечать не будете, поэтому – начинается задание!

Так или иначе, алгоритмы перед собеседованием лучше знать, чем не знать, поэтому суть задачи в следующем:

Зайдите в википедии на страницу, где перечисляется большой список сортировок - <https://bit.ly/2LYQ5eR>

Выберите из этого списка следующие пять алгоритмов:

|  |  |  |
| --- | --- | --- |
| 1 | Сортировка пузырьком | Bubble sort |
| 2 | Сортировка вставками | Insertion sort |
| 3 | Сортировка слиянием | Merge sort |
| 4 | Сортировка Шелла | Shell sort |
| 5 | Быстрая сортировка | Quicksort |

И реализуйте их. Вот и все задание ☺

А для того, чтобы была возможность проверить вашу работу – создайте отдельный класс с методом **main()**. В этом классе\методе:

1. Объявите константу для базовой директории сохранения результатов. Например:

**private static final String RESULTS\_BASE\_DIR = “D:/results”**

2. Попросите пользователя ввести размер массива и дождитесь от него корректного ввода.

3. Создайте массив в соответствии с размером из предыдущего пункта и заполните этот массив случайными целыми числами.

4. Отсортируйте исходный массив всеми алгоритмами, приведенными выше, и для каждого алгоритма замерьте время выполнения (в нано секундах).

**Вопрос:** Как измерить время выполнения какого-либо алгоритма?

**Ответ:** У класса System есть метод, который возвращает количество нано секунд, прошедших с 1 января 1970-го года ☺:

**long nanoTime = System.nanoTime();**

Дальше я хотел вам полное решение написать, но потом подумал – «Ребятам ведб не будет интересно тогда ☺». Поэтому, что вам нужно делать дальше с этим вопросом – подумайте самостоятельно.

**Подсказка:** Когда я начал писать вам этот ответ, прошло какое-то количество нано секунд с 1970-го года. Когда я закончил, прошло еще какое-то количество нано секунд. Теперь пропойте в голове песню Аллы Пугачевой – *«Жизнь невозможно повернуть назааааад, и время не на миг не остановишь…»*. После этого в голове должен родиться алгоритм.

Нормальная подсказка: Ну или просто почитайте комментарии от Java разработчиков, которые они оставили над методом **System.nanoTime()**.

5. Результаты каждой сортировки, а также время выполнения запишите в отдельный файл.

**Вопрос:** Как записать данные в файл?

**Ответ:** У нас еще не было такой темы, но, конечно, она будет. Пока же вы можете использовать вот такой алгоритм:



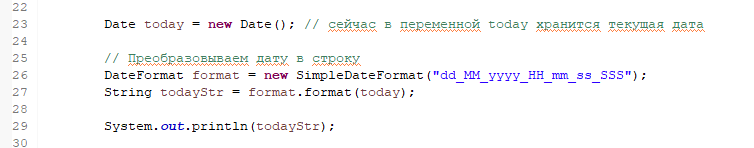
5. Каждый новый запуск программы должен писать результаты в новый файл, не удаляя при этом предыдущие результаты.

**Вопрос:** Какое имя лучше всего подобрать для такого случая?

**Ответ:** В этом задании лучше всего использовать текущую дату с точностью до миллисекунды.

**Вопрос:** как получить текущую дату текущую дату с точностью до миллисекунды?

**Ответ:** Создайте объект **java.util.Date** – вы получите текущую дату. После этого получите из даты строку, в заданном вами формате, при помощи класса **java.text.DateFormat**. Вот так:



Замечание: Естественно вам придется почитать про классы **java.util.Date** и **java.text.DateFormat**.

6. Для сортировки Quicksort используйте алгоритм **Dual Pivot Quick Sort**:

<https://bit.ly/2Yvxc64>

<https://bit.ly/2uvfcLc>

Этот пункт обусловлен тем, что именно Dual Pivot Quick Sort используется в Java в методе **Arrays.sort()**. Вопрос – «Какой именно алгоритм сортировки используется в Java» – может прозвучать на собеседовании. Как лучше всего запомнить ответ на этот вопрос? Естественно, реализовать алгоритм ☺

7. Ну и, на всякий случай. Хотя я уверен, что вы понимаете, что:

1. Каждый алгоритм вы должны реализовать в отдельном методе.
2. Все методы сортировок должны находиться в отдельном классе и отдельном пакете.

Вот теперь точно все. Заключение можно не читать.

**Заключение:** Друзья, это сложная работа. Но ее нужно сделать. Во-первых, потому что эта работа повторится у нас еще два раза. Во-вторых, потому что здесь реально много новых вещей, которые можно попробовать. И, в-третьих, то, с чего начинали – риск собеседования по алгоритмам очень велик.

И как уже говорил, сортировки – это самое простое, что у вас могут спросить. А есть еще очереди и стэки, деревья и хэш-таблицы, алгоритмы поиска и алгоритмы обхода узлов графа, и еще туча других ругательных слов.

Если вы попадете на такое собеседование – это будет ваш маленький персональный ад – тут без вариантов ☺. Но вы можете попытаться подкрутить термостат в своем котле. Для этого до своего первого собеседования очень неплохо прочитать (а главное, максимально понять) любую книгу по алгоритмам.

Одной из лучших книг этой области на Java считается вот эта – **«Структуры данных и алгоритмы в Java»** – <https://bit.ly/2HQvZBl>

**Вопрос:** а ты сам-то эту книгу читал?

**Ответ:** Ай все! Слишком длинное задание получилось ☺

Успехов вам, мои юные гении алгоритмов!