Санкт-Петербургский политехнический университет Петра Великого

Институт компьютерных наук и технологий

Высшая школа интеллектуальных систем и суперкомпьютерных технологий

**КУРСОВОЙ ПРОЕКТ**

**Дисциплина:** Алгоритмы и структуры данных

**Тема:** решатель домино

Выполнил

студент гр. 3530901/90003 \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Ремнева Е.Е.

(подпись)

Руководитель \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Ахин М.Х.

(подпись)

«\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2020 г.

Санкт-Петербург   
2019

**ОГЛАВЛЕНИЕ**

[**ТЕХНИЧЕСКОЕ ЗАДАНИЕ** 3](#_Toc58804651)

[**МЕТОД РЕШЕНИЯ** 4](#_Toc58804652)

[**СКРИНШОТЫ РАБОТЫ ПРОГРАММЫ** 6](#_Toc58804653)

[**ЛИТЕРАТУРА** 10](#_Toc58804654)

# **ТЕХНИЧЕСКОЕ ЗАДАНИЕ**

Решатель «Домино» подразумевает под собой нахождение самой длинной возможной цепочки из костяшек, выбранных пользователем. Естественно, цепочка должна быть построена по всем правилам игры в домино. А именно, каждая следующая костяшка, подставленная к предыдущей, обязана иметь в начале такое же количество точек, как и предшествующая в конце.

Пользователь вправе выбрать любое количество костяшек, всего в наборе 28. Выбор костяшки происходит путем щелчка левой кнопки мыши по ней, ее цвет становится на несколько тонов светлее, так пользователь может определить, какие костяшки были им выбраны. Чтобы вернуть костяшку в первоначальное состояние (невыбранное) достаточно еще раз щелкнуть по ней. После нажатия на кнопку построить пользователь увидит новое окно, на котором будет высвечена самая длинная цепочка, возможная для данного набора. Не каждая цепочка сможет поместить в размере окна, поэтому было предусмотрено то, что пользователь сможет ее листать вправо и влево с помощью нажатия кнопок клавиатуры «стрелка влево» и «стрелка вправо».

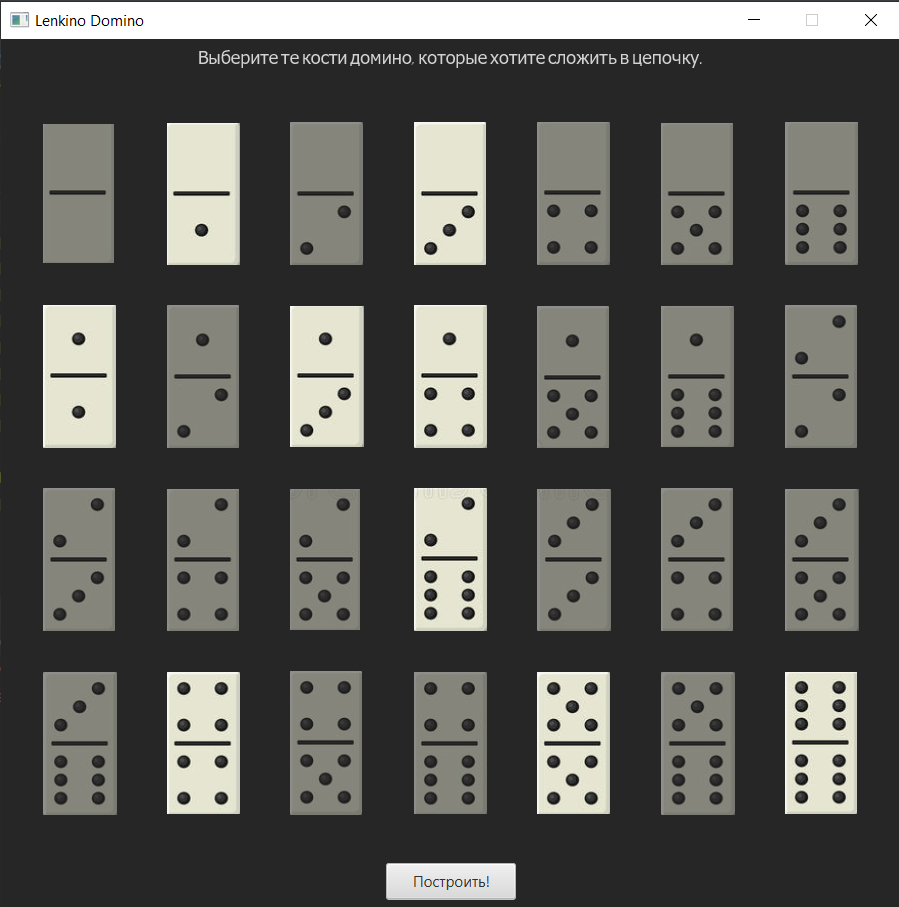


Рис.1 Техническое задание.

Ссылка на репозиторий проекта GitHub : <https://github.com/RemnevaElena/Domino.git>

# **МЕТОД РЕШЕНИЯ**

В проекте код разделен на 2 пакета: sample, test, res. В пакете sample находятся все классы, которые отвечают за логику программы и визуализации, в пакете test, соответственно находятся unit- тесты, которые проверяют правильность работы методов, определенных в классах пакета sample, в пакете res находятся изображения костяшек домино.

Содержимое пакета sample:

* Main – запуск приложения.
* Controller – данный класс отвечает за корректный ввод данных пользователем и визуализацию первого окна.
* ResController – данный класс отвечает за связь между классами Controller и Solver, а так же за вывод результата (самой длинной цепочки) во второе окно.
* Bone – данный класс определяет тип объекта, с которым будет работать программа.
* BoneClickHandler – данный класс обрабатывает события, которые происходят в результате взаимодействия пользователя с мышью (клик по костяшке домино), неразрывно связан с классом Controller.
* Solver – данный класс реализует построение самой длинной цепочки.

Работа любого приложения начинается с интерфейса, и в первую очередь мы оцениваем приложение по визуализации его рабочих компонентов.

За реализацию ввода данных, с которыми потом будет работать программа, отвечали ToggleButton (кнопка – переключатель) , имеющая два устойчивых состояния on/off. Для каждого ToggleButton назначалось свое изображение кости домино.

В методе initialize() класса Controller находится реализация метода BoneClickHandler, определенного в классе HandleFunction, являющегося функциональным.

Здесь же есть и отсылка на метод getButton, который также находится в классе Controller, он возвращает нам ToggleButton, для того, чтобы потом мы могли вернуть кнопке прозрачность на 100%, в случае, если она нажата, и 50 %, если она не была выбрана пользователем.

Все кнопки ToggleButton размещены на GridPane, это помогло их упорядочить и упростить возможность обращения к ним.

В этом же классе находятся и методы OpenNewStage и onResultClick, они создают новое окно для вывода результата действия программы.

В классе ResController определен метод initialize(), который так же как и метод initialize() в классе Controller, отвечает за вывод результата на экран. Если кнопки на первом окне располагались в GridPane, то здесь был использован Hbox, располагающий все дочерние элементы в одну горизонтальную строку.

Из этого же метода происходит и запуск главного алгоритма.

Класс Bone является одним из самых важных классов, так как именно он определяет объект, с которым выполняется вся работа.

В данном классе есть только два поля, first и second, единственным образом определяющие каждую костяшку домино. Здесь, естественно, есть проверка на возможность существования данной доминошки, то есть костяшка с индексами (7; 9) не пройдет.

Так как одна костяшка может выдавать себя за две, меняя полярность своих концов, здесь предусмотрен переопределенный метод equals(), который не позволяет дважды использовать одну и ту же костяшку, и , конечно, сравнивать разные костяшки.

Так же присутствуют методы first() и second(), возвращающие первое значение поля костяшки и второе значение поля костяшки.

Метод, возвращающий количество костяшек, getHowManyBone; метод getAllBone, возвращающий все костяшки, тоже находятся в классе Bone.

В классе ResController создается экземляр класса Solver, а именно ArrayList , и запускается метод makeChain, определенный в классе Solver. Класс Solver и строит самую длинную цепочку. В нем находятся три метода exist(), отвечающий за определение, есть ли в данном наборе костяшка, отвечающая требованиям построения цепочки, makeChain(), который определяет, является ли построенная цепочка лучшей. Данный метод неразрывно связан со следующим методом makeCep, который и строит самую длинную цепочку из костяшек, заданных ему пользователем.

За основу для алгоритма программы был взят видоизмененный алгоритм многопузырьковой сортировки. Основной принцип предлагаемого метода сортировки заключается в том, что из массива случайным образом выбираются четные пары соседних чисел, которые затем сравниваются (В конструктор класса Solver передается массив bones. И, например, в публичном методе makeChain() из 8 элементов на одном такте одновременно могут сравниваться от 1 до 3 пар). После сравнения данная пара чисел временно удаляется для того, чтобы избежать их повторного сравнения на равенство. Если числа были переставлены в результате сравнения, то данная пара удаляется, так как имеет смысл выполнять сравнение для не переставленных костяшек домино. В случае если сравниваемая пара чисел не была переставлена, то есть левый элемент костяшки домино не равен значению правого элемента предыдущей костяшки домино, то блокировка пары чисел снимается. Алгоритм заканчивает свою работу в случае запрета на сравнение для всех пар чисел массива solved.

Так же присутствуют два класса Junit – тестов для проверки правильности работы методов программы.

# **СКРИНШОТЫ РАБОТЫ ПРОГРАММЫ**

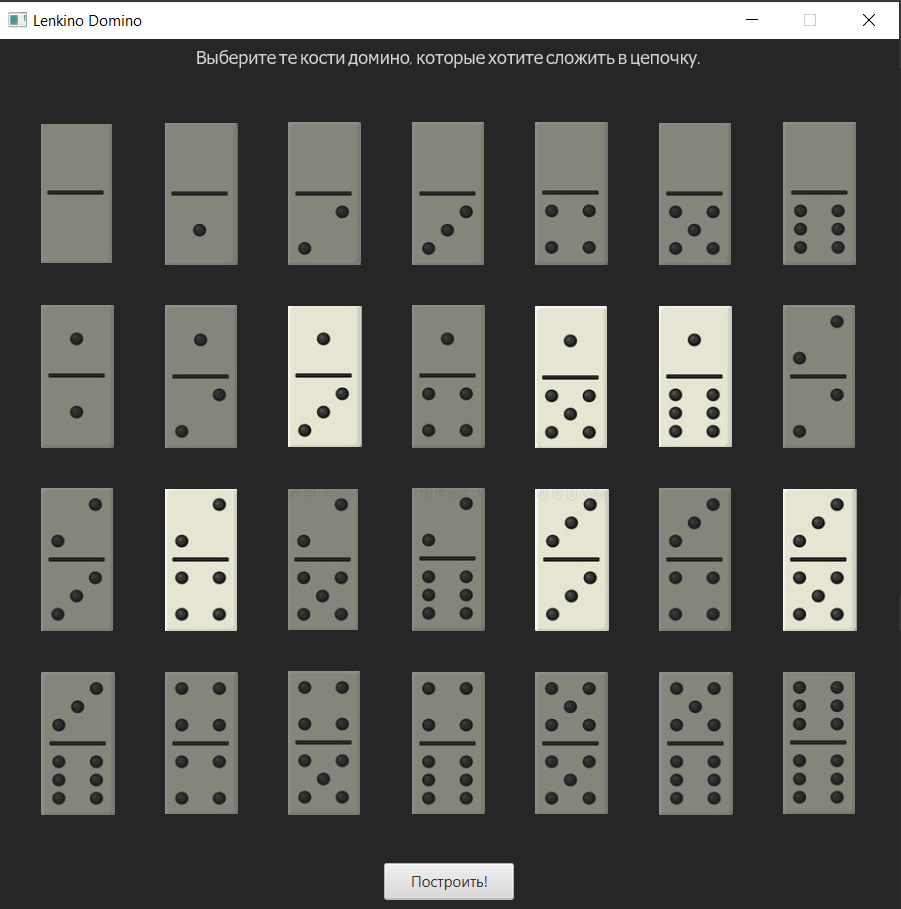


Рис.2 Скриншот работы программы. Выбор костей домино.

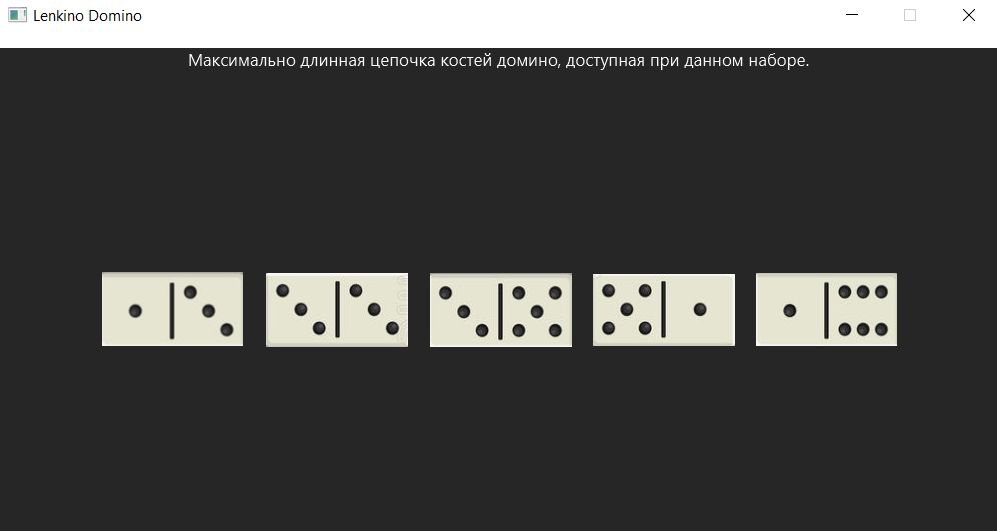


Рис.3 Скриншот работы программы. Вывод результата.

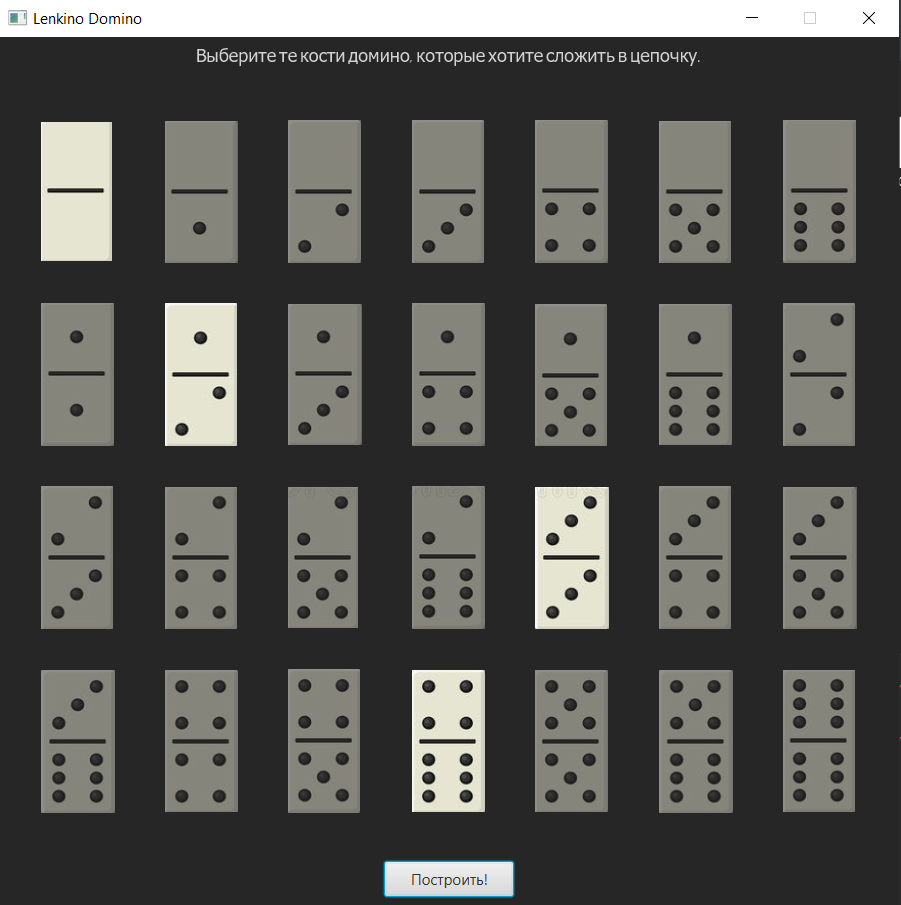


Рис.4 Скриншот работы программы. Выбор костей домино.

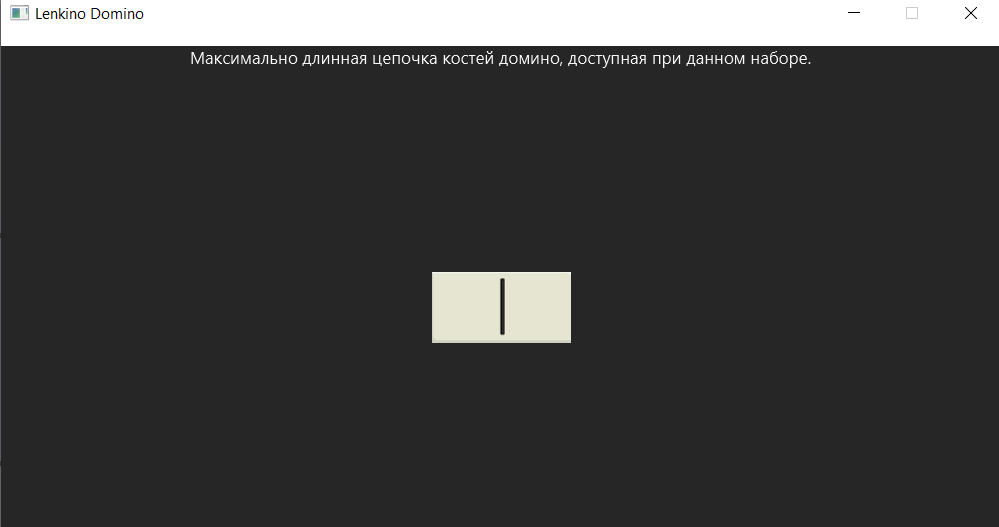


Рис.5 Скриншот работы программы. Вывод результата.

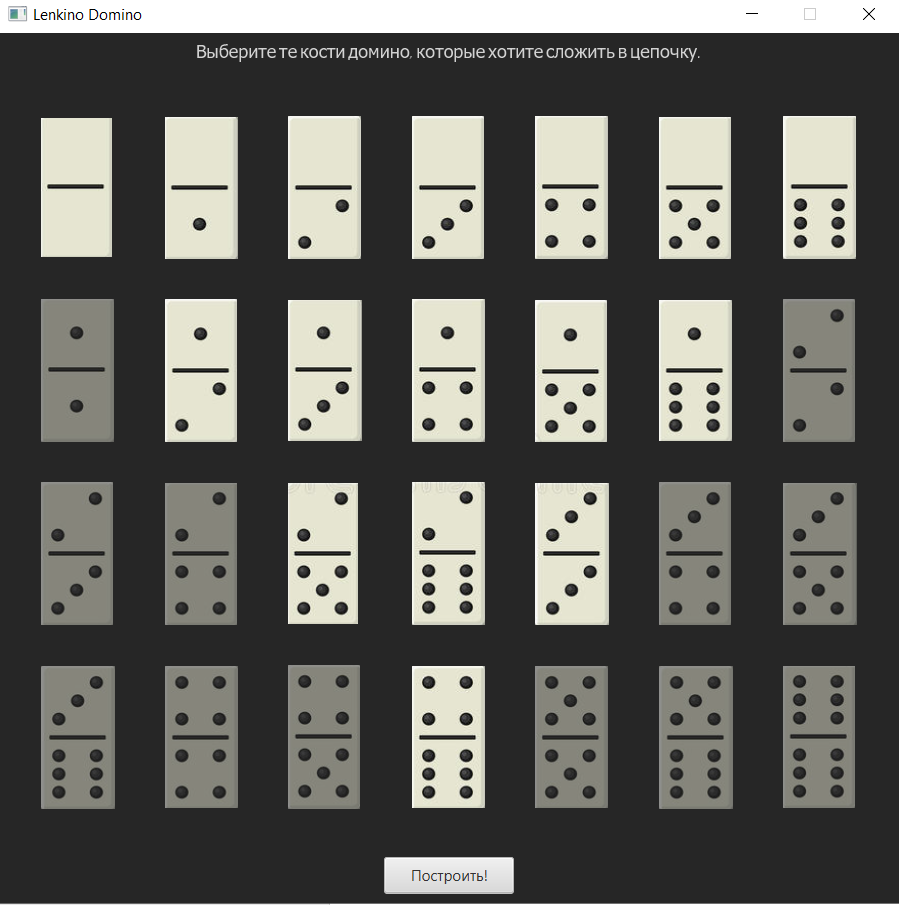


Рис.6 Скриншот работы программы. Выбор костей домино.



Рис.7 Скриншот работы программы. Вывод результата.

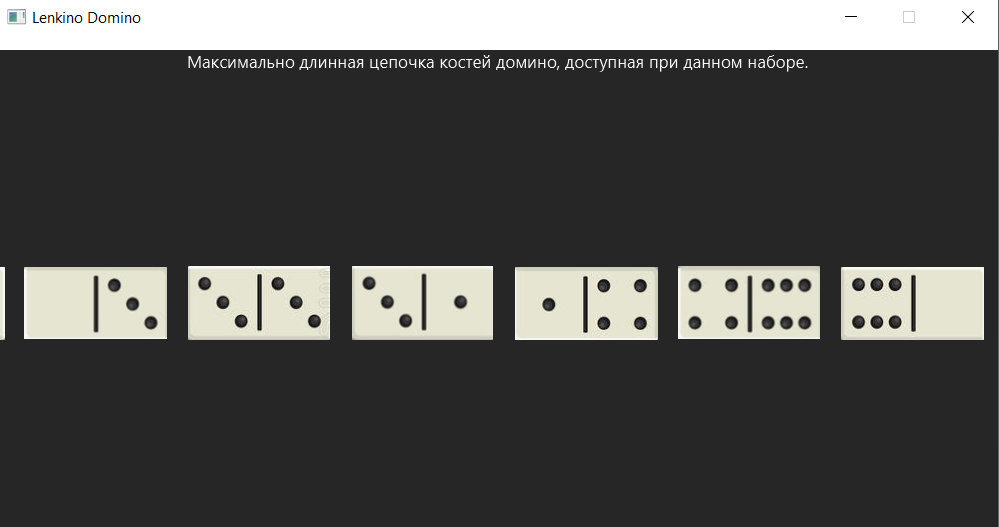


Рис.8 Скриншот работы программы. Вывод результата. Продолжение.

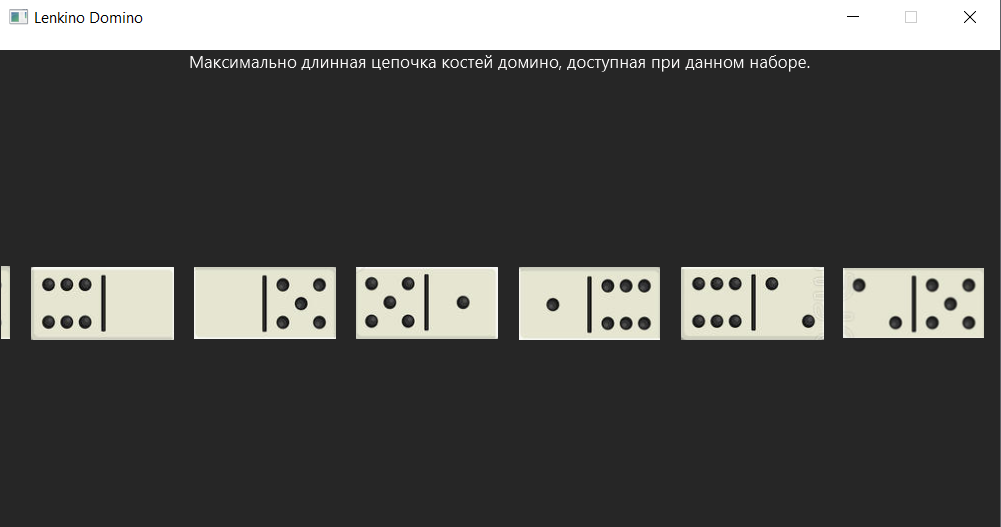


Рис.9 Скриншот работы программы. Вывод результата. Продолжение.

# **ЛИТЕРАТУРА**

<https://e-maxx.ru/bookz/files/cormen.pdf>

<https://javarush.ru>

<http://developer.alexanderklimov.ru>

<https://metanit.com>

https://habr.com