**JSP API.**

**Задание-09.**

**Игра Lines.**

По традиции, в каждой новой технологии нужно написать что-нибудь сложное, чтобы прочувствовать эту технологию. Не будем отступать от этой традиции и реализуем при помощи JSP классическую игру **Lines-98**.

Если вы никогда раньше в нее не играли, то пример этой игры можно увидеть вот здесь - <https://www.lines98.com>.

Перед тем как мы опишем правила, пара очень важных пояснений. Ребята, основная цель этой задачи:

Показать вам, что JSP – такое же унылое @#$%, как и сервлеты!

И если в вашем арсенале будут только JSP, то реализовать хоть сколь-нибудь сложную задачу вы не сможете! Поэтому, не ставьте перед собой цель выполнить эту работу полностью. На самом деле, будет идеально, если вы

1. Начнете ее
2. Выполните процентов 30 от общего объема, например:
   1. Отрисовку игрового поля при открытии JSP страницы
   2. Генерирование новых шариков рандомного цвета и в рандомных (свободных) клетках.
   3. И, для особо отважных, - передвижение шарика – но это уже не обязательно.
3. Увидите, что код получается еще хуже чем у сервлетов.
4. Произнесете – «Какая же гадость эта ваша заливная JSP!»
5. И больше никогда не допустите мысли, что писать на чистых JSP – это хорошая идея ☺

Но все же для порядка опишем правила:

1. Пользователю отображается поле размером 9\*9
2. При начальном отображении поля игра генерирует 5 шариков разных цветов в случайных клетках поля.
3. После каждого хода игрока игра генерирует по три дополнительных шарика разных цветов.
4. Ход игрока – это перемещение любого (но одного) шарика из его текущей клетки в любую свободную клетку.
5. **Внимание:** ход считается возможным, если шарик может переместиться из начальной клетки в конечную двигаясь только по горизонтали или вертикали. То есть, шарик может ходить в любую соседнюю клетку вверх, вниз, влево или вправо, но не может ходить по диагонали.
6. Так же шарик в процессе движения не может перепрыгивать через другие шарики.
7. Если выбранный игроком маршрут невозможен, то шарик должен остаться в своей клетке, а над игровым полем должно появиться сообщение – «Движение невозможно! Выбери другой маршрут!»
8. **Внимание:** Каждый ход игрока – это новый запрос на одну и туже JSP с игровым полем.
9. Задача игрока сложить шарики одного цвета в линию по вертикали, горизонтали или диагонали. Минимальная длинна линии – 5 шаориков.
10. Если линия достигла минимальной длинны, шарики исчезают с игрового поля, занимаемые ими клетки освобождаются, а пользователю начисляется по одному очку за каждый «сожженный» шарик.
11. Игра прекращается проигрышем игрока, когда на поле не остается свободных клеток для генерации новых шариков.

Из всего этого условия, наиболее интересным моментом является алгоритм определения:

1. Возможно ли движение шарика из точки A в точку B.
2. Ну и само движения.

Для этого вы можете погуглить алгоритмы прохождения лабиринтов:

<https://habr.com/ru/post/198266>

<https://habr.com/ru/post/445378>

<https://myrobot.ru/articles/logo_mazesolving.php>

<https://intellect.icu/poisk-vykhoda-iz-labirinta-75>

<https://ru.wikipedia.org/wiki/%D0%90%D0%BB%D0%B3%D0%BE%D1%80%D0%B8%D1%82%D0%BC_%D0%9B%D0%B8>

И попытаться написать код для одного из них. При всей убогости этой работы – алгоритм прохождения лабиринта – действительно интересное и полезное задание. Ведь роботов-пылесосов в нашей жизни меньше не становится… И кому-то в будущем потребуется их программировать… А потом программировать роботов-менеджеров, чтобы управлять роботами-пылесосами, когда их станет неуправляемо много. А потом перепрограмировывать робота T-800, для того, чтобы он помог бороться с восставшими пылесосами, и прочей нечистью, которую мы успели напрограммировать до этого ☺