



Rush 00 - Piscine Java

Консольная игра и мавен

Резюме: Сегодня вы будете реализовывать довольно сложный игровой бизнес-процесс с помощью инструмента сборки Maven

Содержание

I	Предислов ие	2
II	Инструкци и	3
III	Упражнение Сдавайтесь, вы окружены ОО :	5

Глава I

Предисловие

В период с 2000 по 2010 год в программном обеспечении контроллера дроссельной заслонки известной автомобилестроительной компании были обнаружены многочисленные проблемы, которые привели к 89 серьезным авариям.

Проблема заключалась не в конструкции автомобиля, а в некачественном программном обеспечении. Это был код-спагетти, не поддающийся никаким методам тестирования.

К расследованию проблемы подключилось даже НАСА.

Ниже приведено количество дефектов, обнаруженных в программном обеспечении контроллера:

<i>Table A.8-8. Variable Scope</i>		
Camry05	Type	Description
1,872	b	Uninitialized static (file-local)
2,800	C	Uninitialized common (extern)
108	d	Initialized static (file-local)
6,473	D	Initialized common (extern)
5	r	Read-only static (file-local)
91	R	Read-only common (extern)
914	t	Text, static (file-local)
3,710	T	Text, common (extern)

И этот контент небезопасен для просмотра:

<i>Table A.8-9. Use of Global Scope</i>		
Camry05	Type	Description
9,273	C+D	Externally visible variables
1,980	b+d	File-local variables

Глава II


Инструкции

- Используйте эту страницу как единственную ссылку. Не слушайте никаких слухов и домыслов о том, как приготовить раствор.
- Теперь для вас существует только одна версия Java - 1.8. Убедитесь, что компилятор и интерпретатор этой версии установлены на вашей машине.
- Вы можете использовать IDE для написания и отладки исходного кода.
- Код чаще читают, чем пишут. Внимательно прочитайте [документ](#), в котором приведены правила форматирования кода. При выполнении каждой задачи убедитесь, что вы следуете общепринятым [стандартам Oracle](#)
- Комментарии не допускаются в исходном коде вашего решения. Они затрудняют чтение кода.
- Обратите внимание на разрешения ваших файлов и каталогов.
- Для оценки ваше решение должно находиться в вашем GIT-репозитории.
- Ваши решения будут оценивать ваши товарищи по аквариуму.
- Вы не должны оставлять в своем каталоге никаких других файлов, кроме тех, которые явно указаны в инструкциях к упражнению. Рекомендуется изменить свой .gitignore во избежание несчастных случаев.
- Когда вам нужно получить точный вывод в ваших программах, запрещено выводить предварительно рассчитанный вывод вместо правильного выполнения упражнения.
- У вас есть вопрос? Спросите своего соседа справа. В противном случае попробуйте поговорить с соседом слева.
- Ваше справочное пособие: товарищи / Интернет / Google. И еще кое-что. На любой ваш вопрос есть ответ на Stackoverflow. Узнайте, как правильно задавать вопросы.
- Внимательно прочитайте примеры. В них могут потребоваться вещи, которые не указаны в предмете.
- Используйте "System.out" для вывода

- И да пребудет с вами Сила!
- Никогда не оставляйте на завтра то, что вы можете сделать сегодня ;)

Глава III

Упражнение 00: Сдавайтесь, вы окружены

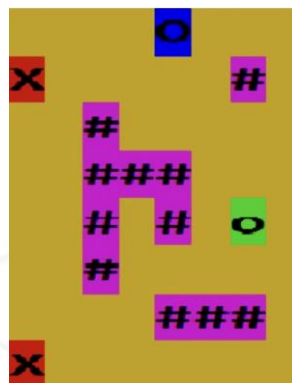
	Упражнение 00
Сдавайтесь, вы окружены	
Входящий каталог :	
<i>exoo/</i>	
Файлы для сдачи : Game-folder, ChaseLogic-folder	
Разрешенные функции : Все	

Помните ли вы старые добрые Java-игры? В начале 2000-х они были в каждом телефоне. Сейчас Java-разработчики проектируют масштабируемые корпоративные системы, но в то время...

Ваша цель сегодня - немного поностальгировать и реализовать игру, в которой вы убегаете от сущностей искусственного интеллекта по квадратному полю.

Программа должна генерировать случайную карту с препятствиями. И игрок, и его враги располагаются на карте случайным образом. Каждый элемент карты должен иметь определенный цвет.

Пример сгенерированной карты:



Обозначения:

o - положение игрока (пользователя программы) на карте. # - препятствие
x - враг (искусственный интеллект)

О - целевая точка, до которой игрок должен добраться до того, как враги достигнут игрока. Считается, что игрок достиг целевой клетки, если он наступил на ее место.

Правила игры:

1. Каждый участник (игрок и противники) может сделать один ход. Затем наступает очередь другого участника. Противник считается достигшим игрока, если он может наступить на позицию игрока, сделав текущий ход.
2. Доступные направления движения: влево, вправо, вниз и вверх.
3. Если враг не может двигаться вперед (вокруг него есть препятствия или другие враги, или достигнут край карты), он пропускает ход.
4. Целевая точка является препятствием для противника.
5. Если игрок не может двигаться вперед (окружен препятствиями, врагами или достиг края карты), он проигрывает игру.
6. Игрок проигрывает, если враг обнаружит его до того, как он достигнет целевой точки.
7. Игрок начинает игру первым.

Требования к реализации:

1. Размер поля, количество препятствий и количество врагов вводятся в программу с помощью параметров командной строки (их наличие гарантировано):

```
$ java -jar game.jar -- enemiesCount=10 --wallsCount=10 --size=30 --profile=production
```

2. Проверяется, можно ли разместить указанное количество врагов и препятствий на карте заданного размера. Если входные данные неверны, программа выбросит непроверенное исключение `IllegalParametersException` и выключится.
3. Враги, препятствия, игрок и целевая точка располагаются на поле случайным образом.
4. При генерации карты враги, игрок, препятствия и целевая точка не должны пересекаться.
5. В начале игры карта должна быть сгенерирована таким образом, чтобы игрок мог добраться до целевой точки (игрок не должен быть заблокирован стенами и краем карты в исходном положении).
6. Чтобы сделать ход, игрок должен ввести на консоли число, соответствующее направлению движения A, W, D, S (влево, вверх, вправо, вниз).
7. Если игрок не может сделать ход в указанном направлении, вводится другое число (направление).
8. Если в начале или середине игры игрок понимает, что целевая точка недостижима, он должен закончить игру, введя 9 (игрок проигрывает).
9. Как только игрок сделал ход, наступает очередь его противника сделать ход в сторону игрока.
10. В режиме развития каждый шаг противника должен быть подтвержден игроком путем ввода 8.
11. При каждом шаге любого участника карта должна быть перерисована в консоли. В режиме разработки карта должна отображаться без обновления экрана.
12. Алгоритм преследования должен учитывать местоположение целевого объекта на каждом шаге.

Требования к архитектуре:

1. Должны быть реализованы два проекта: `Game` (содержит игровую логику, точку входа в приложение, функциональность вывода и т.д.) и `ChaseLogic` (содержит

реализацию алгоритма преследования).

2. Оба проекта являются maven-проектами, и ChaseLogic должен быть добавлен в качестве зависимости в pom.xml внутри Game.

3. Архив Game.jar должен быть переносимым: JCommander и JCDP должны быть непосредственно включены в архив. В то же время, все библиотеки, связанные с проектом, должны быть объявлены как maven-зависимость. Для сборки такого архива должны быть использованы следующие плагины.

Также необходимо создать файл конфигурации под названием application-production.properties. В этом файле вы укажете настройки вашего приложения. Пример этого файла показан ниже:

```
enemy.char = X
player.char = o
wall.char = #
goal.char = O
empty.char =
враг.цвет = КРАСНЫЙ
игрок.цвет =
ЗЕЛЕНый стена.цвет =
МАДЖЕНТА ворота.
цвет = СИНИЙ
пустота.цвет =
ЖЕЛТЫЙ
```

Этот файл конфигурации будет расположен в папке resources запущенного jar-архива.

Кроме того, должен быть реализован файл application-dev.properties. Структура этого файла аналогична структуре файла application.properties. Здесь вы можете указать параметры для различать запуск приложения в режиме разработки (например, разные цвета/символы для компонентов карты).

Необходимо помнить, что программа может быть запущена и в других режимах. Для этого в исходный проект может быть добавлен соответствующий файл свойств, а сам режим передается через параметр --profile.

