

Другая Монополия

(Домашнее задание 1)

1. Игра

Игра представляет собой консольное приложение (Java-приложение, запускаемое из командной строки) и использует консольный ввод-вывод для взаимодействия с пользователем в режиме диалога.

Игровая доска (игровое поле) представляет собой прямоугольник, составленный из клеток, при этом последняя клетка примыкает к первой (см. рис.1). При реализации допускаются различные способы изображения игрового поля с использованием скромных графических средств консольного ввода. В простейшем варианте вместо изображения клетки может быть выведено лишь ее символьное обозначение. Перечень обозначений и отображений для всех разновидностей клеток указан ниже (см. пункт 2: *Генерация поля и обозначения его клеток*). Поощряется реализация (средствами консольного вывода¹) более выразительных способов изображения клеток с их символьными обозначениями.

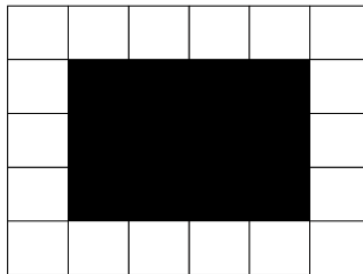


Рис.1. Концептуальный (примерный) вид игрового поля.

Количество клеток по ширине (*width*) и по высоте (*height*) задается аргументами командной строки при запуске приложения.

В начале игры у каждого игрока есть одинаковый стартовый капитал, величина которого (*money*) тоже является аргументом командной строки. Начальное положение игрока – клетка в левом верхнем углу (0, 0) игрового поля, которая является нейтральной (*EmptyCell*) – см. описание разновидностей клеток ниже.

¹ Консольный вывод может работать по-разному на разных платформах; например, при использовании Windows для вывода разноцветных символов в консоль может понадобиться изменение информации в Windows registry. При разработке Java-приложений следует избегать использования средств, зависящих от платформы (оно должно одинаково работать на всех).

Игроки делают свои ходы по очереди; при этом они перемещаются по игровому полю по часовой стрелке, тратя и получая деньги. Очередность ходов определяется случайно в начале игры и далее не изменяется. Игра завершается, когда остаётся только один игрок, который не обанкротился. Он и объявляется победителем.

В данном задании игра рассчитана на двух игроков, одним из которых является бот. В задании требуется реализовать бота, который будет ходить аналогично игроку, но не будет пользоваться услугами офисов банка. При этом решения о покупке или улучшении магазина принимаются ботом случайно. Особые примечания, касающиеся поведения бота, даны по тексту ниже.

2. Генерация поля и обозначения его клеток

Поле генерируется случайным образом, со следующими условиями:

1. На четырех углах карты находятся нейтральные клетки (*EmptyCell*). Нейтральные клетки обозначаются символом 'Е'.
2. В игре участвует один банк (*Bank*) с многими офисами. Клетки офисов банка присутствуют на каждой линии в единственном экземпляре. Офисы банка обозначаются символом '\$'.
3. На каждой линии есть до 2 такси (*Taxi*). Такси обозначаются символом 'Т'.
4. На каждой линии есть до 2 штрафных клеток (*PenaltyCell*). Штрафные клетки обозначаются символом '%'.
5. Все остальные клетки - магазины (*Shop*). Магазины обозначаются в зависимости от того, когда и для кого выполняется отображение игрового поля:
 - магазин, не имеющий владельца, отображается символом 'S';
 - магазин, принадлежащий игроку, для которого выполняется отображение игрового поля, отображается для него символом 'М' (т.е. как My Shop);
 - магазин, принадлежащий сопернику, отображается символом 'О' (т.е. как Opponent Shop).

3. Виды клеток

При попадании игрока на любую клетку в консоль должны выводиться сообщения, содержащие информацию о происходящем в игре (например, может быть выведен тип клетки, изменения в капитале). Важно, чтобы сообщение было понятно игроку и давало четкую информацию о любых изменениях.

Подробности – см. далее.

- Магазин (Shop)

- Если у магазина нет владельца, игроку предоставляется возможность купить этот магазин по стоимости N и стать его владельцем, либо отказаться, т.е. пройти мимо (соответствующий диалог описан в пункте 4: *Ход игрока*).
- Если этот магазин – собственность попавшего на него игрока, то владелец может улучшить свой магазин или отказаться, т.е. пройти мимо (см. пункт 4: *Ход игрока*). За один ход улучшить магазин можно только один раз, но при этом за всю игру улучшать его можно неограниченное количество раз.
- За один ход можно ИЛИ купить магазин, ИЛИ улучшить его, уже являясь его владельцем, ИЛИ ничего не делать.
- Если игрок попадает на клетку магазина, принадлежащую другому игроку, игрок обязан выплатить владельцу магазина компенсацию в размере K . Если игрок не располагает нужной суммой, игра заканчивается его поражением.
- Улучшение магазина - это разовое изменение стоимости магазина и размера компенсации за его посещение по указанным ниже правилам.
- Каждое улучшение магазина влияет на него следующим образом:
 - Стоимость магазина повышается на $improvementCoeff * N$ (т.е., новая стоимость магазина равна $N + improvementCoeff * N$);
 - Размер компенсации увеличивается на $compensationCoeff * K$ (т.е., новая компенсация для этого магазина становится равной $K + compensationCoeff * K$);
 - Коэффициенты $compensationCoeff$ и $improvementCoeff$ должны генерироваться для каждого магазина отдельно до начала игры; в течение игры эти коэффициенты не изменяются. Каждый раз коэффициенты применяются к текущим значениям N и K (то есть $N = N + improvementCoeff * N$; $K = K + compensationCoeff * K$).

Здесь K - текущий размер компенсации, N – цена магазина; $compensationCoeff$ – коэффициент увеличения компенсации, $improvementCoeff$ – коэффициент увеличения стоимости улучшения магазина.

- Банк (Bank)

- Если попавший на ячейку офиса банка игрок является должником банка, с его счета списывается требуемая сумма. Если игрок не располагает нужной суммой, игра заканчивается его поражением.
- Если попавший на ячейку офиса банка игрок не является должником банка, он может получить от банка сумму, но не больше, чем $creditCoeff * (сумма, которую игрок потратил на покупку и улучшение всех своих магазинов)$. При этом он становится должником банка на сумму, которая в $debtCoeff$ раз больше, чем полученная от банка сумма (соответствующий диалог описан в разделе 4: *Ход игрока*). Значение $debtCoeff$ генерируется единожды за

партию по правилу $1.0 < debtCoeff \leq 3.0$. Значения *creditCoeff* и *debtCoeff* обязательно выводятся в консоль для сведения игрока в начале партии.

- Штрафная клетка (PenaltyCell)
 - С игрока взимается плата в размере *penaltyCoeff* * (количество денег игрока). Значение *penaltyCoeff* генерируется единожды за партию. Обязательно выводится в консоль для сведения игрока в начале партии.
- Такси (Taxi)
 - Игрок перемещается на *taxiDistance* клеток вперед. Значение *taxiDistance* генерируется для каждого попадания на соответствующую клетку. Игроку выводится сообщение в консоль: «*You are shifted forward by <taxiDistance> cells*»
- Нейтральная клетка (EmptyCell)
 - В консоль выводится сообщение: «*Just relax there*».

4. Ход игрока

Игра начинается с того, что:

- Генерируется и выводится игровое поле
- Выводятся значения коэффициентов
- Определяется очередность ходов игроков с помощью генератора случайных чисел (в дальнейшем игроки ходят по очереди).
- Происходит ход первого игрока.

При ходе игрока:

- Генерируются два числа от 1 до 6, и игрок перемещается вперед на сумму сгенерированных чисел.
- В зависимости от типа клетки, на которую попал игрок, в консоль выводится информация о доступных на текущей клетке действиях и предоставляются соответствующие возможности диалогового взаимодействия.

Примеры:

- «*You are in the bank office. Would you like to get a credit? Input how many you want to get or 'No'*»
- «*You are in <shop cell <X><Y>. This shop has no owner. Would you like to buy it for <price>\$? Input 'Yes' if you agree or 'No' otherwise*».
- «*You are in your shop <X><Y>. Would you like to upgrade it for <price>\$? Input 'Yes' if you agree or 'No' otherwise*».

Если пользователь вводит недопустимое число или набор слов, он должен быть уведомлен соответственно, и процедура ввода должна быть повторена.

После каждого действия игрока, в консоль должна выводиться соответствующая информация.

5. Завершение хода

После окончания хода в консоль выводится игровое поле (см. рис.1 выше). Обозначения для объектов указаны ранее в разделе “Генерация поля и обозначение клеток”.

Ниже выводятся положения игроков и информация о текущем местоположении игрока, балансе и долге игрока (в формате «*You are in the cell (<X>, <Y>). <Значения, которые изменились в течение завершившегося хода>*». Например, “*You balance: <balance>\$.*” и другие. Все такие значения выводятся отдельными строками).

6. Как играет бот

Бот ходит как игрок.

Бот может покупать и улучшать магазины, решение о покупке или улучшении магазина принимается случайно. Бот может пользоваться такси.

Бот НЕ может пользоваться услугами банка (ни в одном из его офисов).

7. Входные данные

Входные данные задаются аргументами командной строки:

- ширина поля `width`; $6 \leq \text{width} \leq 30$;
- высота поля `height`; $6 \leq \text{height} \leq 30$;
- стартовый капитал игрока `money`; $500 \leq \text{money} \leq 15000$.

Порядок аргументов командной строки при запуске программы следующий: `<height> <width> <money>`.

8. Дополнительная информация

Далее представлена таблица значений, о которых шла речь выше.

Переменная	Минимальное начальное значение (включая)	Максимальное начальное значение (включая)	Примечание
<code>height</code>	6	30	Задается пользователем при запуске (параметр командной строки приложения)
<code>width</code>	6	30	Задается пользователем при запуске (параметр

			командной строки приложения)
N	50	500	Стоимость магазина. Генерируется случайно в указанном интервале для каждого магазина отдельно в начале игры.
K	$0.5 * N$	$0.9 * N$	Начальная компенсация магазина. Генерируется случайно в указанном интервале для каждого магазина отдельно в начале игры.
compensationCoeff	0.1	1	Генерируется случайно в указанном интервале для каждого магазина отдельно. Коэффициент не меняется в течение игры.
improvementCoeff	0.1	2	Генерируется случайно в указанном интервале для каждого магазина отдельно. Коэффициент не меняется в течение игры.
creditCoeff	0.002	0.2	Значение генерируется случайно в указанном интервале один раз в начале игры и выводится в консоль.
debtCoeff	1.0	3.0	Значение генерируется случайно в указанном интервале один раз в начале игры и выводится в консоль.

taxiDistance	3	5	Целочисленное значение генерируется случайно в указанном интервале для каждого попадания на любую клетку taxi.
penaltyCoeff	0.01	0.1	Значение генерируется случайно в указанном интервале один раз в начале игры и выводится в консоль.

Если параметры, которые должны передаваться в качестве аргументов командной строки, отсутствуют, им должны присваиваться default-значения в соответствующих пределах (по усмотрению студентов) или должны выдаваться сообщения об ошибке запуска приложения.

9. Информация об оценивании

Каждый исходный файл должен в начале содержать javadoc-комментарий с указанием автора, аналогичный указанному ниже:

```
/**
 * @author <a href="mailto:isidorov@edu.hse.ru"> Ivan Sidorov</a>
 */
```

Все методы должны иметь javadoc-комментарии. Приветствуются также комментарии по ходу решения.

Текст программы должен быть отформатирован согласно общепринятому стилю кодирования (можно использовать <https://google.github.io/styleguide/javaguide.html> или тот, что указан и используется в IntelliJ IDEA by default).

Методы должны быть достаточно короткими, чтобы помещаться на экран. Исключения возможны только в том случае, когда метод невозможно разделить на более короткие (спойлер: у вас не выйдет).

10. Результат выполнения домашнего задания

Результат выполнения домашнего задания загружается в SmartLMS в виде одного результирующего архива с именем: **HW1_<номер_группы>_<фамилия>_<имя>.zip**. Например: **HW1_195_Sidorov_Ivan.zip**

В этом архиве должен быть упакован отдельный проект, выполненный в IntelliJ IDEA с использованием JDK11 (проекты, выполненные с использованием других JDK не засчитываются). В проекте должны содержаться все настройки, исходные файлы и

прочие необходимые файлы для компиляции и сборки работающего приложения в исполняемый JAR.

Приложение должно работать в контексте IntelliJ IDEA и автономно при запуске его с использованием исполняемого JAR'а приложения. Исполняемый JAR, настройки артефакта для генерации которого должны присутствовать в проекте IntelliJ IDEA, должен быть собран, присутствовать в результирующем архиве задания и иметь имя, совпадающее с именем проекта, например: **HW1_195_Sidorov_Ivan.jar**. Пример его запуска из командной строки:

```
java -jar HW1_195_Sidorov_Ivan.jar 30 30 15000
```

Допускается (и приветствуется) наличие командных файлов, запускающих исполняемый JAR приложения на разных платформах (MAC, Linux, Windows).

Все предоставленные решения подвергаются проверке на плагиат, выявление которого строго наказывается.

Deadline: 19 октября 2020, 23:00 Moscow Time.