Командные кости

Компания друзей отправляется в казино, чтобы поиграть в кости. В выбранном заведении в кости играют командами по три человека. Команда победителей получает приз – от ¥1 000 000 до ¥10 000 000.

Правила в заведении следующие:

1. `t` команд по 3 игрока играют в кости. `t` – параметр командной строки, целое число от 1 до 10;
2. У каждой команды свой стол с 6-ю костями;
3. Каждый участник команды подходит к столу и бросает каждую из 6 костей;
4. Кость выдаёт очки в диапазоне от 1 до 6 включительно случайным образом;
5. Перед началом игры (при запуске потока) игрок представляется: приветствует всех, говорит свои имя и команду. Например, пишет в консоль «Hello there, I'm John from Winx Club»;
    a. Имя игрока генерируется любым образом, главное, чтобы оно было уникально. Например, можно выбирать случайное из заранее определенного набора имён. То же самое касается имени команды
6. Одновременно за столом может находиться только один участник команды.

Бросив 6 костей, участник команды подходит к общей для всех таблице с результатами и прибавляет число набранных очков к сумме очков своей команды. После этого игрок спит. Проснувшись, он снова пытается получить доступ к столу, чтобы бросить кости. Длительность сна игрока определяется случайным образом в диапазоне от 100 до 1000 миллисекунд.

Крупье

В игре есть крупье. Он каждые десять секунд в консоль пишет: текущего лидера игры – команду, набравшую наибольшее количество очков, собственно количество очков у команды-лидера и оставшееся время до окончания игры. Если таких команд несколько, то крупье выводит информацию о каждой из них. Во время объявлений команд-лидеров ни один игрок не может добавить количество очков к сумме очков своей команды.

Конец игры

Игра длится 35 секунд.

В конце игры крупье прерывает всех игроков, объявляет победителя, генерирует случайным образом приз в размере от ¥1 000 000 до ¥10 000 000, вручает приз победившей команде и показывает (в консоли) итоговую таблицу с результатами. Если победителей несколько, то приз делится между командами-победителями поровну.

Каждый игрок команды-победителя выводит в конце игры количество денег, которые получит лично он, пропорционально количеству заработанных им очков.

Пример: игрок *A* команды “Клуб Винкс” принёс команде 10 очков, члены его команды – 90 в сумме, сумма очков – 100, приз – ¥1 000 000, тогда игрок *A* сообщает, что его личный выигрыш составил ¥100 000. Точность вычисления – два знака после запятой.

После этого каждый игрок прощается, написав об этом в консоль, и покидает казино (поток завершает работу). После чего крупье предлагает запустить игру заново с другими участниками (новые потоки). При этом количество команд остается тем же.

Детали реализации

1. Каждый игрок представляется потоком – экземпляром класса `java.lang.Thread`, крупье – главный поток программы (main);
2. Для синхронизации разрешено использовать только intrinsic-lock механизм: ключевое слово `synchronized`, методы `Object::notify`, `Object::notifyAll`, `Object::wait`;
    a. Другие механизмы синхронизации,то есть – все понятия, определенные в пакете java.util.concurrent и его подпакетах (например, `java.util.concurrent.locks.Lock`), ключевое слово `volatile`, синхронизованные коллекции и т.п., использовать запрещено.

Тестирование

Логика игры в методах и классах должна быть покрыта тестами с помощью JUnit 5. Требуется покрытие тестами строк кода не менее чем на 60%. Покрытие многопоточности тестами не обязательно

Формат сдачи

Имя архива – по формуле LastName_FirstName_hw2.zip.

В архиве:

1. Java 17 maven-проект с реализацией игры и тестами;
2. Исполняемый jar-файл в корне проекта, собранный maven;

3. README-файл с инструкцией по запуску приложения.

Пример запуска jar

```
$ java -jar application.jar 10
```

10 - `t` – количество команд в игре

Оценивание

- За отсутствие потоков максимальная оценка 3;
- При грубых ошибках синхронизации потоков максимальная оценка 6;
- При незначительных ошибках синхронизации максимальная оценка 8;
- Оценка 9 -10 ставится при отсутствии ошибок синхронизации и за качественный код. В понятие «качественного кода» входит и его эффективность / производительность (с позиций полезного использования процессора).