**Задание №1. Оптимизация работы светофоров на перекрестке.**

Передо мной стояла цель обеспечить водителям минимальное время для проезда на перекрестке.

Дополнительные условия для задачи

Для решения поставленной задачи я начал разрабатывать алгоритм, для которого справедливы следующие уточнения.

1. Все виды транспорта имеют одинаковое время проезда перекрестка, имеется ввиду время, затрачиваемое автомобилем для преодоления расстояния между двумя светофорами.
2. Работа программы начинается с установки зеленого цвета светофоров вертикальной составляющей перекрестка.
3. Зеленый свет светофора означает, что машина во время пересечения перекрестка может повернуть в любое направление кроме обратного.

UML диаграмма моей программы

Изображение выглядит как стол

Автоматически созданное описание

Изображение выглядит как текст

Автоматически созданное описание

Реализация и пояснения  
Итак, для работы программы я решил использовать 3 потока, каждый из которых имеет свой функционал.  
MyThreadPaint – класс, наследуемый от класса Thread, реализующий логику для вывода на экран “снимка улицы”. Снимок улицы – это консольный вывод состояний светофоров, нагруженности каждой дороги.   
MyThreadRoadAutoWorker – класс, наследуемый от класса  Thread, имеющий реализацию для работы с показателями загруженности дороги.  
MyThreadTrafficLightState – класс, наследуемый от класса Thread, имеющий реализацию для работы со светофорами.

Light – класс, идея которого состоит в том, чтобы хранить состояние светофора.  
LightState -  enum класс, состоящий из возможных состояний светофора.  
Car -  pojo класс для модели машин.

Реализация. В главном потоке программы создаются коллекции и объекты светофоров, такое решение принято прежде всего из-за того, что есть вероятность создания GOD object - огромного класса, хранящего в себе статические поля и методы для работы с данными нашего перекрестка. Поскольку необходимо работать с данными из всех перечисленных потоков, я решил обеспечить потокобезопасность при получении и обработке. Для этого использовался статический метод Collections.synchronizedList(new ArrayList<>()), возвращающий потокобезопасную коллекцию.

Далее в каждый поток добавляются данные и начинается их обработка, в зависимости от функционала.

MyThreadTrafficLight Реализация. Содержит в себе изменение состояний светофоров. Всего их 5, согласно заданию должно быть 4, но я решил добавить еще одно состояние, оно может быть использовано при поломке и вызвать службу для устранения неисправности. Также имеет логику для показа мигающего зеленого, реализующую условие с поочередным показом зеленого и “бесцветного” состояния, но для удобного использования программы присутствует код объединяющий эти состояния, на данный момент он закомментирован. Оптимизация времени светофоров реализована достаточно простым способом. Сравниваем нагрузку на дороги и ставим приоритет по времени той, на которой ожидает большее количество машин.  
  
MyThreadPrint Реализация. Содержит в себе вывод на экран снимка улицы каждую секунду. Также присутствует функция получения строки по состоянию светофора.  
  
MyThreadRoadAutoWorker Реализация. Задача этого класса случайным образом добавлять и удалять машины из коллекций. Если на светофоре горит зеленый или же мигающий зеленый, то каждую секунду мы удаляем машину, считая, что она пересекла перекресток.

Стоит отметить, так как моя реализация не поддерживает изменение потока автомобилей, программа через некоторое время будет показывать полную загруженность.

Чтобы освободить все наши ресурсы мы добавляем в систему  shutdownHook – поток, который вызывается перед выходом из программы. Там мы закрываем все наши потоки в методе close.