**Порядок выполнения действий:**

1. При запуске приложения необходимо ввести температуру окружающей среды.
2. Далее выбираем тип двигателя. Вводим «1» без кавычек для выбора ДВС.
3. Далее выбираем тип теста для выбранного типа двигателя. Вводим «1» без кавычек для выбора теста на перегрев двигателя.

* В случае ввода некорректных данных или данных не соответствующих набору допустимых пользователь получит уведомление о необходимости внести корректные данные.

**Правила оформления конфигурационного файла для теста на перегрев:**

Значений соответствующие переменным должны быть введены в строгом порядке **через пробел**:

**I**(Момент инерции двигателя)   
**T\_Overheat** (Температура перегрева)  
**arraySize** (Размер массивов M и V, данные массивы должны содержать равное количество элементов)   
**М** (крутящий момент)  
**V** (скорость вращения)  
**Hm** (Коэффициент зависимости скорости нагрева от крутящего момента)  
**Hv** (Коэффициент зависимости скорости нагрева от скорости вращений коленвала)  
**С** (Коэффициент зависимости скорости охлаждения от температуры двигателя и окружающей среды)

Десятичные числа вводятся с **использованием запятой**!

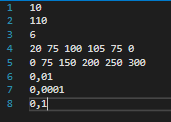
Пример ввода данных в конфигурационный файл представлен на рисунке №1. В данном примере все переменные находятся в разных строках для наглядности.

Рис1. Пример ввода данных в конфигурационный файл.

* В случае, если конфигурационный файл найден и считаные из него значения соответствуют необходимому типу данных, мы получим соответствующее уведомление об успешном считывание.
* В случае, если конфигурационный файл не найден или считанные из него данные не соответствуют необходимому типу данных, мы получим уведомление об ошибке и данные для симуляции будут выставлены значениями по умолчанию.

**Логические блоки приложения:**

**1). Симуляция двигателя внутреннего сгорания.**

За симуляция ДВС ответственен класс InternalCombustionEngine , являющийся наследником класса Engine.

Данный класс содержит поля (параметры симуляции, а также рассчитываемые значения) и методы: StartEngine() – запуск и симуляция работы двигателя, StopEngine() – остановка и прекращений симуляции двигателя, а также геттеры для необходимых полей.

**2). Логика тестирования двигателя на перегрев.**

За логику тестирования двигателя отвечает класс TestingStand. На вход объект данного класса получает объект двигателя, необходимого для проведения теста.

Данный класс содержит один метод: getTimeBeforeOverheat(), который возвращает время, прошедшее от начала симуляции до момента перегрева в секундах. Данный метод запускает симуляцию ДВС в отдельном потоке, а в основном отслеживает показатель температуры двигателя и при заданной температуре перегрева отключает двигатель и прекращает симуляцию. Если двигатель не успел перегреться, то метод вернёт «-1».

**3). Консольный ввод-вывод, задание исходных данных и запуск теста.**

Данный логический блок реализован в главном классе. Здесь происходит получение данных от пользователя и считывание данных для симуляции из конфигурационного файла, а также вывод всех сообщений пользователю.

Метод printOverheatTestResult() выводит результат о перегреве, полученный при вызове метода getTimeBeforeOverheat объекта тестового стенда.

Если я правильно понял, то при старте симуляции двигатель должен отработать по заданному в ТЗ графику. Если начать тест на низкой температуре, то двигатель может не успеть перегреться к концу графика и вылетит ошибка по причине того, что двигатель все еще работает, но переходы состояния кончились (график) и мы улетаем за пределы значений вектора, поэтому будем считать, что двигатель отработал без перегрева и выключать его.