## Лабораторная работа №2

**Тема:** Разработка через тестирование

**Цель:** продемонстрировать способ разработки через тестирование при реализации типовых задач

**Навыки и умения:** TDD, моделирование предметной области, алгоритмы, абстрагирование, использование языков графического моделирования

**Условия выполнения:** лабораторная работа предназначена для выполнения одним студентом

## Описание задания

Приложение должно отображать изменение некоторой величины, максимальное и минимальное значение для данного набора из N значений. Приложение будет состоять из:

- генератора значений случайной величины;

- класса очереди, необходимого для решения задачи;

- простейшего UI для отображения текущего состояния программы (консоль, GUI – на Ваше усмотрение).

Необходимо выполнить разработку класса очереди через тестирование. Тесты можно писать как с использованием готовых библиотек и фреймворков (например, Junit, Mockito в Java), так и в виде отдельного приложения. Каждый тест должен содержать входные/выходные данные и собственно проверку. Результат каждого теста: пройден/не пройден. При сдаче работы необходимо продемонстрировать исходное состояние: набор тестов и класс очереди ДО реализации (все тесты должны быть не пройдены), и конечный результат: пройдены все тесты, работающая программа.

## Требования к программе

Программа должна обеспечивать возможность выполнения перечисленных ниже функций:

- отображение текущего набора значений (консоль, GUI);

- отображение минимального и максимального значений из текущего набора значений.

Отказы программы вследствие некорректных действий пользователя не допустимы.

Требования к языкам программирования и используемым программным средствам не предъявляются.

## Требования к реализации

Необходимо отображать набор из N **целочисленных** значений.

Шаг программы должен выполняться при нажатии любой клавиши. При каждом шаге необходимо:

- сгенерировать случайное значение;

- обновить текущий набор значений;

- отобразить текущий набор значений, минимум и максимум.

Текущий набор значений необходимо хранить в очереди. Класс очереди должен содержать, как минимум, следующие методы **enqueue(int value), dequeue(), minimum(), maximum()**. Сложность поиска максимального и минимального значения должна быть **O(1)**. Требования к очереди по пямяти: не более N элементов + возможно, вспомогательные переменные.

## Прием работ

* демонстрация разработки через тестирование:
* написанные тесты до реализации очереди
* реализация очереди и пройденные тесты
* та же очередь в коде программы
* демонстрация работы программы
* обсуждение
* дополнительные вопросы
* в случае успешной сдачи выставляется фиксированный балл.

## Задание повышенной сложности

Выполнение задания повышенной сложности даёт дополнительные баллы при условии сдачи всех лабораторных, аналогичное повышению оценки с 3 до 4, и -1 вопрос на экзамене (по умолчанию при выполнении всех лабораторных работ можно получить 3 без сдачи экзамена). ЛР№3 также будет содержать задание повышенной сложности.

**Условие:**

- программа должна работать в реальном времени вместо выполнения шага при нажатии клавиши. Потребуется либо применение многопоточности, либо модель продвижения времени со списком событий без многопоточности, либо что-то ещё. Например: раз в 0.1с проверяется список событий. Если на данное время запланировано какое-то событие (генерация нового значения, обновление UI, проверка очереди на наличие устаревших значений), то выполняем его. Если нет – засыпаем ещё на 0.1с и затем снова проверяем список событий;

- генератор должен выдавать значения в случайное время каждые [0.1 – 5] с;

- обновлять UI в части состояния текущего набора значений 1 раз в 3 секунды. Не обновлять UI чаще, например, при поступлении нового значения от генератора;

- отображать отдельно последние 10 значений, выданных генератором. Этот список необходимо обновлять сразу после поступления значения.

## Список литературы

1. Кент Бек. Экстремальное программирование: разработка через тестирование. — «Питер», 2003.