## Практическая работа №11

## Задания на очереди

**Теоретические сведения ( что нужно знать)**

1. Классы

* **Инвариант класса**
* **Задачи инкапсуляции**

1. Интерфейсы

* **Интерфейс как синтаксический контракт**
* **Интерфейс как семантический контракт**

1. Абстрактные базовые классы и наследование
   * Устранение дублирования

* **Вынос изменяемой логики в наследников**

**Задания**

**Задание 1. Очередь на массиве**

* Найдите инвариант структуры данных «[очередь](http://ru.wikipedia.org/wiki/%D0%9E%D1%87%D0%B5%D1%80%D0%B5%D0%B4%D1%8C_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5))». Определите функции, которые необходимы для реализации очереди. Найдите их пред- и постусловия.
* Реализуйте классы, представляющие циклическую очередь с применением массива.
  + Класс ArrayQueueModule должен реализовывать один экземпляр очереди с использованием переменных класса.
  + Класс ArrayQueueADT должен реализовывать очередь в виде абстрактного типа данных (с явной передачей ссылки на экземпляр очереди).
  + Класс ArrayQueue должен реализовывать очередь в виде класса (с неявной передачей ссылки на экземпляр очереди).
  + Должны быть реализованы следующие функции(процедуры) / методы:
    - enqueue – добавить элемент в очередь;
    - element – первый элемент в очереди;
    - dequeue – удалить и вернуть первый элемент в очереди;
    - size – текущий размер очереди;
    - isEmpty – является ли очередь пустой;
    - clear – удалить все элементы из очереди.
  + Инвариант, пред- и постусловия записываются в исходном коде в виде комментариев.
  + Обратите внимание на инкапсуляцию данных и кода во всех трех реализациях.
* Напишите тесты реализованным классам.

**Задание 2. Очереди**

1. Определите интерфейс очереди Queue и опишите его контракт.
2. Реализуйте класс LinkedQueue — очередь на связном списке.
3. Выделите общие части классов LinkedQueue и ArrayQueue в базовый класс AbstractQueue.

**Дополнительные задания**

**Задание 3. Вычисление выражений**

1. **Разработайте классы Const, Variable, Add, Subtract, Multiply, Divide для вычисления выражений с одной переменной.**
2. **Классы должны позволять составлять выражения вида**

new Subtract(

new Multiply(

new Const(2),

new Variable("x")

),

new Const(3)

).evaluate(5)

 При вычислении такого выражения вместо каждой переменной подставляется значение, переданное в качестве параметра методу evaluate (на данном этапе имена переменных игнорируются). Таким образом, результатом вычисления приведенного примера должно стать число 7.

1. **Для тестирования программы должен быть создан класс Main, который вычисляет значение выражения x2−2x+1, для x, заданного в командной строке.**
2. **При выполнение задания следует обратить внимание на:**

* **Выделение общего интерфейса создаваемых классов.**
* **Выделение абстрактного базового класса для бинарных операций.**

**Задание 4**

1. **Доработайте предыдущее задание, так что бы выражение строилось по записи вида**

x \* (y - 2)\*z + 1

 Для этого реализуйте класс ExpressionParser с методом TripleExpression parse(String).

1. **В записи выражения могут встречаться: умножение \*, деление /, сложение +, вычитание -, унарный минус -, целочисленные константы (в десятичной системе счисления, которые помещаются в 32-битный знаковый целочисленный тип), круглые скобки, переменные (x, y и z) и произвольное число пробельных символов в любом месте (но не внутри констант).**
2. **Приоритет операторов, начиная с наивысшего**

* **унарный минус;**
* **умножение и деление;**
* **сложение и вычитание.**

1. **Для выражения 1000000\*x\*x\*x\*x\*x/(x-1) вывод программы должен иметь следующий вид:**

**x f**

0 0

1 division by zero

2 32000000

3 121500000

4 341333333

5 overflow

6 overflow

7 overflow

8 overflow

9 overflow

10 overflow

1. **Результат division by zero (overflow) означает, что в процессе вычисления произошло деление на ноль (переполнение).**
2. **Разбор выражений рекомендуется производить** [**методом рекурсивного спуска**](https://ru.wikibooks.org/wiki/%D0%A0%D0%B5%D0%B0%D0%BB%D0%B8%D0%B7%D0%B0%D1%86%D0%B8%D0%B8_%D0%B0%D0%BB%D0%B3%D0%BE%D1%80%D0%B8%D1%82%D0%BC%D0%BE%D0%B2/%D0%9C%D0%B5%D1%82%D0%BE%D0%B4_%D1%80%D0%B5%D0%BA%D1%83%D1%80%D1%81%D0%B8%D0%B2%D0%BD%D0%BE%D0%B3%D0%BE_%D1%81%D0%BF%D1%83%D1%81%D0%BA%D0%B0)**. Алгоритм должен работать за линейное время.**
3. **При выполнении задания следует обратить внимание на дизайн и обработку исключений.**
4. **Человеко-читаемые сообщения об ошибках должны выводится на консоль.**
5. **Программа не должна «вылетать» с исключениями (как стандартными, так и добавленными).**