Занятие 2

Задания

Самостоятельное изучение

- Изучить введение в Maven. (Часть 4. Основы Maven) Разобрать понятия pom, parent pom, effictive pom, dependency
- Разобрать понятия SOLID (Принципы SOLID на примерах)
- Принципы ООП Java / Объектно-ориентированное программирование
- Разобраться, что такое пакет
- Разобрать и запомнить модификаторы доступа для членов класса
- Повторить, как в Java реализованы абстракция, полиморфизм, наследование.
- Разобрать методы класса java.lang.Object
- Разобраться в интерфейсах и абстрактных классах, чем они отличаются и чем похожи
- Varargs, правила статического полиморфизма Java Challengers #1: Перегрузка методов в JVM

Задачи

Для каждой задачи реализовать **отдельный** класс с методом main, в котором демонстрируется работа решения.
Пример для Sequences.
Интерфейс называется Sequences
Реализация - SequencesImpl
Демонстрация - SequencesTest. В данном классе есть метод main, в котором создается экземпляр объекта и вызываются методы

Sequences

Последовательности A - J заданы в виде нескольких значений следующим образом

```
A. 2,  4,  6,   8,  10...
B. 1,  3,  5,   7,   9...
C. 1,  4,  9,  16,  25...
D. 1,  8, 27,  64, 125...
E. 1, -1,  1,  -1,   1,  -1...
F. 1, -2,  3,  -4,   5,  -6...
G. 1, -4,  9, -16,  25....
H. 1,  0,  2,   0,   3,   0,  4....
I. 1,  2,  6,  24, 120, 720...
J. 1,  1,  2,   3,   5,   8, 13, 21…
```

Необходимо найти закономерности, по которым эти последовательности сформированы и реализовать следующий интерфейс, каждый метод которого

принимает число N и выводит в консоль N элементов соответствующей последовательности. Каждый элемент можно выводить с новой строки

```
public interface SequenceGenerator {
 /**
  * Выводит в консоль n первых членов последовательности A
  * @param n число членов последовательности для вывода
  */
 void a(int n);
 /**
  * Выводит в консоль n первых членов последовательности B
  * @param n число членов последовательности для вывода
  */
 void b(int n);
 /**
  * Выводит в консоль n первых членов последовательности C
  * @param n число членов последовательности для вывода
  */
 void c(int n);
 /**
  * Выводит в консоль n первых членов последовательности D
  * @param n число членов последовательности для вывода
  */
 void d(int n);
 /**
  * Выводит в консоль n первых членов последовательности E
  * @param n число членов последовательности для вывода
  */
 void e(int n);
 /**
  * Выводит в консоль n первых членов последовательности F
  * @param n число членов последовательности для вывода
  */
 void f(int n);
 /**
  * Выводит в консоль n первых членов последовательности G
  * @param n число членов последовательности для вывода
  */
 void g(int n);
 /**
  * Выводит в консоль n первых членов последовательности H
  * @param n число членов последовательности для вывода
  */
 void h(int n);
 /**
  * Выводит в консоль n первых членов последовательности I
  * @param n число членов последовательности для вывода
  */
 void i(int n);
 /**
  * Выводит в консоль n первых членов последовательности J
  * @param n число членов последовательности для вывода
  */
 void j(int n);
}
```

Complex Numbers

Реализовать класс, описывающий комплексное число (действительная и мнимая часть должны иметь точность double). Должны быть доступны следующие операции:

1. Создание нового числа по действительной части (конструктор с 1 параметром)
2. Создание нового числа по действительной и мнимой части (конструктор с 2 параметрами)
3. Сложение
4. Вычитание
5. Умножение
6. Операция получения модуля
7. преобразование в строку (toString)
   (арифметические действия должны создавать новый экземпляр класса)

Написать код, демонстрирующий работу с созданными классами

RateLimitedPrinter

Реализовать класс RateLimiterPrinter. Класс имеет конструктор, в который передается **interval** и метод print(), в который передается строка. Класс функционирует по следующему принципу: на объекте класса вызывается метод print(). Далее идет проверка, когда был последний вывод в консоль. Если интервал (в миллисекундах) между последним состоявшимся выводом и текущим выводом больше значения **interval**, переданного в конструктор - то происходит вывод значения. Иначе - не происходит, и сообщение отбрасывается. То есть класс ограничивает частоту вывода в консоль. Другими словами, сообщение не будет выводится чаще чем 1 раз в **interval** милисекунд. Реализовать описанный класс.

```
public class RateLimitedPrinter {
 public RateLimitedPrinter(int interval) {
   // code here
 }

 public void print(String message) {
   // code here
 }
}
```

Пример использования. Задается вывод не чаще 1 раза в секунду, далее запускается цикл.

```
public static void main(String[] args) {
 RateLimitedPrinter rateLimitedPrinter = new RateLimitedPrinter(1000);
 for (int i = 0; i < 1_000_000_000; i++) {
   rateLimitedPrinter.print(String.valueOf(i));
 }

 long currentTime = System.currentTimeMillis(); // возвращает текущее
                                                // время
}
```

Snils Validator

Номер СНИЛС состоит из 11 цифра, и валидация номера СНИЛС выполняется по следующим правилам: (Валидация и проверка контрольного числа СНИЛС)

Алгоритм проверки контрольного числа

1. Вычислить сумму произведений цифр СНИЛС (с 1-й по 9-ю) на следующие коэффициенты — 9, 8, 7, 6, 5, 4, 3, 2, 1 (т.е. номера цифр в обратном порядке).
2. Вычислить контрольное число от полученной суммы следующим образом:
    1. если она меньше 100, то контрольное число равно этой сумме;
    2. если равна 100, то контрольное число равно 0;
    3. если больше 100, то вычислить остаток от деления на 101 и далее:
        1. если остаток от деления равен 100, то контольное число равно 0;
        2. в противном случае контрольное число равно вычисленному остатку от деления.
3. Сравнить полученное контрольное число с двумя младшими разрядами СНИЛС. Если они равны, то СНИЛС верный.

Реализовать интерфейс **SnilsValidator**

```
public interface SnilsValidator {

 /**
  * Проверяет, что в строке содержится валидный номер СНИЛС
  * @param snils снилс
  * @return результат проверки
  */
 boolean validate(String snils);

}
```

Который возвращает true если номер СНИЛС валидный, false - в противном случае. Можно считать, что номер передается в виде строки, содержащей исключительно цифры от 0 до 9.

```
int x = Character.digit('7', 10);// конвертация символа в число. x == 7
boolean isDigit = Character.isDigit('7'); // true
```

Пример:

```
System.out.println(new SnilsValidatorImpl().validate("01468870570")); //false
```

```
System.out.println(new SnilsValidatorImpl().validate("90114404441")); //true
```

Обратить внимание, что переданная строка может быть произвольной. Метод должен возвращать true тогда и только тогда, когда в строке валидный СНИЛС

StatsAccumulator

Необходимо реализовать интерфейс StatsAccumulator

```
public interface StatsAccumulator {

 /**
  * Добавляет число в аккумулятор. Вызывается многократно
  * @param value число
  */
 void add(int value);

 /**
  * Возвращает минимальное из всех добавленных чисел
  * @return минимальное из всех добавленных чисел
  */
 int getMin();

 /**
  * Возвращает максимальное из всех добавленных чисел
  * @return максимальное из всех добавленных чисел
  */
 int getMax();

 /**
  * Возвращает количество всех добавленных чисел
  * @return количество добавленных чисел
  */
 int getCount();

 /**
  * Возвращает среднее арифметическое всех добавленных чисел
  * @return среднее арифметическое всех добавленных
  */
 Double getAvg();
}
```

Объект данного класса, будучи созданным, может принимать значения через метод add. Приняв значение, объект меняет свое внутреннее состояние, чтобы в любой момент времени предоставить данные о количестве переданных ему элементах, минимальному их них, максимальному из них, а также о среднем арифметическом всех переданных ему элементов

Пример использования

```
StatsAccumulator s = …; // как то создается

s.add(1);
s.add(2);
System.out.println(s.getAvg()); // 1.5 - среднее арифметическое
элементов
s.add(0);
System.out.println(s.getMin()); // 0 - минимальное из переданных
значений
s.add(3);
s.add(8);
System.out.println(s.getMax()); // 8 - максимальный из переданных
System.out.println(s.getCount()); // 5 - количество переданных элементов
```

Написать решение, использующее O(1) памяти. Другими словами - нельзя хранить все переданные в метод add числа