Задание 1. Int.

Напишите класс, реализующий «правильные с объектно-ориентированной точки зрения целые числа”, В классе должны быть определены

- методы increment() и decrement(), соответственно увеличивающие и уменьшающие число на 1;
- методы add(Int n) и substract(Int n), увеличивающие и уменьшающие число на n;
- метод toString()

Примечание 1: в методах add и substract передаются значения типа Int (с большой буквы), а не int.
Примечание 2: для сдачи основного задания нельзя придумывать свои методы и конструкторы, нужно использовать указанные выше.
Примечание 3: применить инкрементирование 1000 раз - плохой вариант.
Примечание 4: в доп. задании можно создать конструктор Int(int value).

При создании нового объекта должно создаваться число, равное 0.

Напишите наиболее короткую программу, которая используя только класс Int, выводит на экран число 1000. Программа должна быть чисто объектно-ориентированной. В частности, в ней нельзя использовать оператор присваивания.

Задание 2. Матрица.

Напишите класс Matrix, реализующий квадратные матрицы. В нем должны быть определены

- конструктор с целочисленным параметром --- размером матрицы, создающий единичную матрицу;
- методы **Matrix** sum(**Matrix**) и **Matrix** product(**Matrix**), вычисляющие сумму и произведение матриц
- матрицы setElement(int row, int column, int value) и getElement(int row, int column), для обращения к элементам матрицы;
- метод toString() *(Примечание: необходимо использовать StringArray или StringBuilder).*

Во всех методах предполагается, что передаваемые параметры всегда корректны.
Напишите программу, выводящую **первые 10** степеней матрицы:

```
[1 1]
[1 0]
```

Задание 3. Наследование.

Напишите класс Matrix, реализующий матрицы и расширяющий его класс SquareMatrix, реализующий квадратные матрицы. В классах должны быть определены:

- конструкторы с параметрами размерами матриц, создающие нулевую матрицу для Matrix и единичную для SquareMatrix;
- методы **Matrix** sum(**Matrix**) и **Matrix** product(**Matrix**), вычисляющие сумму и произведение матриц; метод sum должен быть переопределен в SquareMatrix;
- методы setElement(int row, int column, int value) и getElement(int row, int column), для обращения к элементам матрицы;
- метод toString().

Напишите собственный класс исключения, расширяющий *(наследующий)* класс RuntimeException. Во всех конструкторах и методах должны бросаться исключения в тех случаях, когда соответствующая операция невозможна (например, при сложении матриц разных размеров). Исключения должны содержать информацию о том, какая именно проблема возникла. Достаточно хранить эту информацию в виде строки, возвращаемой методом getMessage().

Примените к написанной программе:

1. Разложите классы по пакетам.
2. Напишите слово **final** в тех случаях, где оно разумно.
3. Реализуйте для матриц метод equals().

Задание 4. LinkedList.

Напишите класс SortedIntegerList, который хранит отсортированный в порядке возрастания список целых чисел. Внутри класса список должен храниться с помощью LinkedList. У SortedInteger List должны быть определены:

- Конструктор с булевским параметром; если этот параметр принимает значение true, то в создаваемом списке разрешены повторяющиеся элементы, иначе --- нет;
- Методы add(int) и remove(int), которые, соответственно, добавляют число в список и удаляют число из списка; если добавление (удаление) невозможно --- метод не делает ничего. Операции добавления/удаления должны требовать **не более чем одного** прохода по списку;
- Метод equals();
  *Примечание: использовать **везде** итератор.*

Напишите программу, проверяющую работу класса SortedIntegerList. Постарайтесь реализовать возможно полный набор проверок.

Задание 5. Интерфейс.

Напишите интерфейс IMatrix с несколькими реализациями --- UsualMatrix и расширяющий его SquareMatirx из предыдущих заданий и SparseMatrix для разреженных матриц. SparseMatrix должен быть реализован с помощью LinkedList (возможно, вам потребуется создать какие-то еще дополнительные классы, которые должны быть вложенными/внутренними). Все общие методы должны быть представлены в интерфейсе IMatrix.

Напишите программу, создающую 2 случайные матрицы размером 1000x1000 с 1000 ненулевых элементов в каждой двумя способами --- с помощью обычных и разреженных матриц. Проверьте, что сложение и умножение для разных видов матриц дает одинаковые результаты.

Задание 6. Ввод-вывод.

Нужно реализовать две небольшие программы.

1. Реализовать класс FormattedInput с двумя статическими функциями:
Object[] scanf(String format). Читает с System.in.
Object[] sscanf(String format, String in). Читает из строки in.

format --- строка со спецификацией формата ввода (может быть несколько спецификаторов в одной строке, например, «%d %d %f»). Список спецификаторов:
%d --- целое int
%f --- дробное double
%s --- строка
%c --- символ

Если ввод пользователя не соответствует спецификации, то функция запрашивает ввод повторно.
Пример:

```
main(..) {
……...
   Object vals[] = scanf(“%d %s %c”);
……..
}
```

Ввод пользователя: 10 ten v

2. Реализовать программу EncodingConverter для перекодирования текстовых файл из одной кодировки в другую. Программа должна получать параметры из командной строки и контролировать их корректность.

Пример вызова: java EncodingConverter in.txt out.txt utf8 cp1251

Задание 7. Map.

Реализуйте класс для хранения настроек Settings, в котором хранятся пары «имя параметра, значение». «Имя параметра» задается строкой, а «значение» целым числом. Реализация должна использовать класс HashMap. В классе Settings должны быть определены:

- toString() и equals()
- put(String, int)
- int get(String)
- delete(String)
- loadFromBinaryFile(String filename)
- saveToBinaryFile(String filename)
- loadFromTextFile(String filename)
- saveToTextFile(String filename)

Задание 8. Thread.

Реализовать класс ParallelMatrixProduct для многопоточного умножения матриц UsualMatrix. В конструкторе класс получает число потоков, которые будут использованы для перемножения (число потоков может быть меньше, чем число строк у первой матрицы).

В функции main сравнить время перемножения больших случайных матриц обычным и многопоточным способом. Получить текущее время можно с помощью методов класса System.

Задание 9. Synchronized.

Написать программу, приводящую к ситуации взаимной блокировки (deadlock).