1. В компьютерной игре герой (класс Hero) может перемещаться между двумя точками (метод move) различными способами: идти пешком, ехать на лошади, лететь и т. п. Реализовать классы, позволяющие выбирать и менять в ходе выполнения программы способ перемещения героя, используя паттерн “стратегия” (strategy). Продемонстрировать работу реализованных классов.
2. Написать консольное приложение, которое:
    a. Считывает из текстового файл размерность матрицы N*N.
    b. Создаёт и заполняет матрицу случайными числами от -N до N.
    c. Последовательно поворачивает матрицу на 90, 180 и 270 градусов против часовой стрелки и делит каждый элемент на сумму соседних.
    d. Каждую из трёх получившихся матриц вывести в общий файл

    Требования к обработке исключительных ситуаций:
    a. контролировать состояние потоков ввода/вывода (отсутствие записи в файле, недопустимые значения, etc);
    b. генерировать и обрабатывать исключение при некорректных математических операциях;
    c. выбрасывать исключение при нехватке памяти;
    d. реализовать собственные классы исключений для случаев
        - деление на 0
        - файл не существует
        - N > 1_000_000
3. Создайте иерархию животного царства на свое усмотрение - выбираем один тип (простейшие/губки/хордовые и т. д.), а далее наследуйте классы, отряды, семейства, роды и виды) - должна получиться иерархия в 6 уровней, выбор животных на ваш вкус, особо много создавать не нужно. Затем, для получившейся иерархии, выполняем следующее:
    a. Создаем обобщенный класс Queue, представляющий из себя очередь фиксированного размера со стандартными методами очереди - add и get
    b. Создаем методы produce и consume: первый метод должен возвращать upper bound generic очередь (например, наследники позвоночных) из n животных, которая будет генерироваться на ваше усмотрение и подаваться во второй метод. Consume будет их распределять в 2 или более lower bound очереди - например, родители кошек и родители змей, выбор типов также остаётся за вами.

c. Демонстрируем работу всех методов на конкретных собственных кейсах

4. Написать аннотацию с целочисленным параметром. Создать класс, содержащий только приватные методы (3–4 шт.), аннотировать любые из них. Вызвать из другого класса все аннотированные методы столько раз, сколько указано в параметре аннотации.
5. С использованием StreamAPI реализовать следующие методы:
   - метод, возвращающий среднее значение списка целых чисел;
   - метод, приводящий все строки в списке в верхний регистр и добавляющий к ним префикс «_new_»;
   - метод, возвращающий список квадратов всех встречающихся только один раз элементов списка;
   - метод, принимающий на вход коллекцию строк и возвращающий все строки, начинающиеся с заданной буквы, отсортированные по алфавиту;
   - метод, принимающий на вход коллекцию и возвращающий её последний элемент или кидающий исключение, если коллекция пуста;
   - метод, принимающий на вход массив целых чисел, возвращающий сумму чётных чисел или 0, если чётных чисел нет;
   - метод, преобразовывающий все строки в списке в Map, где первый символ – ключ, оставшиеся – значение;
6. Создать супервизор (управляющую программу), которая контролирует исполнение абстрактной программы.

   Абстрактная программа работает в отдельном потоке и является классом с полем перечисляемого типа, который отражает ее состояние (UNKNOWN, STOPPING, RUNNING, FATAL ERROR) и имеет поток-демон случайного состояния, который в заданном интервале меняет её состояние на случайное.

   У супервизора должны быть методы остановки и запуска абстрактной программы, которые меняют ее состояние. Супервизор является потоком, который циклически опрашивает абстрактную программу, и если ее состояние UNKNOWN или STOPPING, то перезапускает ее. Если состояние FATAL ERROR, то работа абстрактной программы завершается супервизором. Все изменения состояний должны сопровождаться соответствующими сообщениями в консоли. Использовать конструкции с wait/notify.
7. Создать очередь сообщений, в которую пишут N потоков (количество потоков задается через args), и читают N потоков, использовать только пакет java.util.concurrent. Имена потоков должны быть осмыслены, использовать конструкции с wait/notify запрещено.