1. Заменить класс **Shape** на интерфейс **Shape** (или **Volume**).
2. Shape должен наследоваться от **Comparable**. То есть, shapes должно быть сравнимы по объему.
   Можно воспользоваться методом *Arrays.sort()*, чтобы убедится, что массив фигур сортируется верно.
3. Дан массив строк. Отсортируйте их все разными способами:
   По алфавиту
   По возрастанию длины
   По количеству вхождений цифр в строку
   Для сортировки можно использовать *Arrays.sort()* с параметром **Comparator**.
   Для вывода строк из массива можно использовать *Arrays.toString()*.
4. Напишите метод *fill*, который принимает массив объектов, и экземпляр интерфейса **Generator** для получения нового значения по индексу.
5. * Стандартная библиотека Java содержит интерфейс **Iterator.** Интерфейс Iterator имеет следующее определение:

```
public interface Iterator <E>{
    E next();
    boolean hasNext();
    // Some other methods (default)
}
```

   Реализация интерфейса предполагает, что с помощью вызова метода *next()* можно получить следующий элемент. С помощью метода *hasNext()* можно узнать, есть ли следующий элемент. И если элементы еще имеются, то *hasNext()* вернет значение **true**. Метод *hasNext()* следует вызывать перед методом *next()*, так как при достижении конца коллекции метод *next()* выбрасывает исключение **NoSuchElementException**.
   Задача:
   5.1. Написать **Iterator** по массиву.
   5.2. Написать «склеивающий» итератор, принимающий в конструкторе два итератора (можно расширить до N), и проходящий по ним последовательно.
   5.3. Дан итератор, метод *next()* которого возвращает либо экземпляр **String**, либо итератор такой же структуры (то есть возможна «рекурсия»). Написать итератор, который проходит по этому итератору перебирая только экземпляры **String**, то есть делая итератор «плоским».

Для тестов можно воспользоваться таким кодом:

```
private static Iterator getRecursiveIterator() {
  List finalList = List.of(
        "1",
        List.of("2", "3", "4").iterator(),
        "5",
        "6",
        List.of("7",
              List.of(
                    List.of("8").iterator(),
                    "9",
                    "10",
                    List.of().iterator()).iterator(),
              "11").iterator(),
        "12");

  return finalList.iterator();
}
```

```
DeepIterator myIterator = new DeepIterator(getRecursiveIterator());
while (myIterator.hasNext()) {
  System.out.println(myIterator.next());
}
```

На выходе должна быть последовательность чисел 1-12