```java
import java.io.IOException;
import java.nio.file.Files;
import java.nio.file.Path;
import java.security.SecureRandom;
import java.util.*;
import java.util.stream.Collectors;
import java.util.stream.IntStream;
import java.util.TreeMap;

public class A6Operations {

    public static Queue<Character> stackToQueue(final Stack<Character> stackOriginal) {
        Queue<Character> newQueue=stackOriginal.stream().collect(Collectors.toCollection(LinkedList::new));
        return newQueue;
    }
    public static List<Integer> uniqueSortedRandoms(final int min, final int max, final int poolSize) {
        SecureRandom s = new SecureRandom();
        return s.ints(poolSize, min, max).boxed().sorted().collect(Collectors.toCollection(ArrayList::new));
    }
    public static int rangeSum(final List<Integer> listInt, final int fromIndex, final int toIndex) throws IllegalArgumentException {
        if (fromIndex > toIndex || fromIndex < 1 || fromIndex > listInt.size() || toIndex > listInt.size()) {
            throw new IllegalArgumentException("Incorrect parameters");
        }
        return listInt.subList(fromIndex -1, toIndex).stream().reduce(0,Integer::sum);
    }

    public static Map<Character, Long> fileCharSummary(final String fileName) throws IOException {

        return (Map<Character, Long>) Files.readString(Path.of(fileName)).chars().mapToObj(c -> (char) c)
                .collect(Collectors.toMap(k -> k, v->1L, Long::sum, TreeMap::new));
    }


    public static void main(String[] args) throws IOException {

        // Test 1, converting a stack to queue
        Stack<Character> stackChar = new Stack<>();
        IntStream.rangeClosed(65, 90)
                .mapToObj(i -> (char) i)
                .forEach(stackChar::add);

        System.out.printf("%s%s%n", "Stack before queue: ", joinStream(stackChar));
        System.out.printf("%s%s%n", "stackToQueue returned: ", joinStream(stackToQueue(stackChar)));
        System.out.printf("%s%s%n%n", "Stack after queue: ", joinStream(stackChar));

        // Test 2, Random number generation. Your results will vary
        List<Integer> listOrderedRands = uniqueSortedRandoms(25, 200, 20);
        System.out.printf("%s%s%n%n", "Ordered list of random numbers without repetition: ", joinStream(listOrderedRands));

        // Test 3, summing a list's elements using different ranges
        List<Integer> listSumTest = Arrays.asList(13, 10, 20, 5, 1, 9, 16, 2, 4, 18, 19, 12, 16, 17, 8);
        System.out.printf("%s%,d%n", "Sum of listSumTest between 0 and 15: ", rangeSum(listSumTest, 0, 15));
        System.out.printf("%s%,d%n", "Sum of listSumTest between 15 and 0: ", rangeSum(listSumTest, 15, 0));
        System.out.printf("%s%,d%n", "Sum of listSumTest between 0 and 14: ", rangeSum(listSumTest, 0, 14));
        System.out.printf("%s%,d%n", "Sum of listSumTest between 14 and 0: ", rangeSum(listSumTest, 14, 0));
        System.out.printf("%s%,d%n", "Sum of listSumTest between 0 and 13: ", rangeSum(listSumTest, 0, 13));
        System.out.printf("%s%,d%n", "Sum of listSumTest between 5 and 4: ", rangeSum(listSumTest, 5, 4));
        System.out.printf("%s%,d%n", "Sum of listSumTest between 3 and 7: ", rangeSum(listSumTest, 3, 7));
        System.out.printf("%s%,d%n", "Sum of listSumTest between 14 and 14: ", rangeSum(listSumTest, 14, 14));

        // Test 4, Letter counter from file
        System.out.printf("%n%s%s", "Character counts: ", joinStream(fileCharSummary("TestFile.txt").entrySet()));
    }

    /**
     * Don't modify. This is a helper function for joining a collection's elements as a String object
     *
     * @param collection the collection with elements to join
     * @return a String object representing all elements in the list sparated by a ', '
     */
    private static <E> String joinStream(Collection<E> collection) {

        return collection.stream()
                .map(String::valueOf)
                .collect(Collectors.joining(", "));
    }
}
```