```java
import components.set.Set;

/**
 * Lets the user test the {@code hashCode(String)} method, by reading text lines
 * from a file (whose name is supplied by the user), and then outputting the
 * distribution of lines into buckets.
 *
 * @author Put your name here
 *
 */
public final class HashingExploration {

    /**
     * Private constructor so this utility class cannot be instantiated.
     */
    private HashingExploration() {
    }

    /**
     * Computes {@code a} mod {@code b} as % should have been defined to work.
     *
     * @param a
     *            the number being reduced
     * @param b
     *            the modulus
     * @return the result of a mod b, which satisfies 0 <= {@code mod} < b
     * @requires b > 0
     * @ensures <pre>
     * 0 <= mod  and  mod < b  and
     * there exists k: integer (a = k * b + mod)
     * </pre>
     */
    public static int mod(int a, int b) {
        assert b > 0 : "Violation of: b > 0";

        if (a > 0 && b > 0) {
            while (a >= b) {
                a = a - b;
            }
        } else if (a < 0 && b > 0) {
            while (-a >= b || a <= b) {
                a = a + b;
            }
        } else if (a < 0 && b < 0) {
            while (a <= b) {
                a = a - b;
            }
        }

        return a;
    }

    /**
     * Returns a hash code value for the given {@code String}.
     *
     * @param s
     *            the {@code String} whose hash code is computed
     * @return a hash code value for the given {@code String}
     * @ensures hashCode = [hash code value for the given String]
     */
    private static int hashCode(String s) {
        assert s != null : "Violation of: s is not null";

        int hash = 0;

        for (int i = 0; i < s.length(); i++) {
            hash = hash + s.charAt(i);
        }

        return 0;
    }

    /**
     * Main method.
     *
     * @param args
     *            the command line arguments
     */
    public static void main(String[] args) {
        SimpleReader in = new SimpleReader1L();
        SimpleWriter out = new SimpleWriter1L();
        /*
         * Get hash table size and file name.
         */
        out.print("Hash table size: ");
        int hashTableSize = in.nextInteger();
        out.print("Text file name: ");
        String textFileName = in.nextLine();
        /*
         * Set up counts and counted. All entries in counts are automatically
         * initialized to 0.
         */
        int[] counts = new int[hashTableSize];
        Set<String> counted = new Set1L<String>();
        /*
         * Get some lines of input, hash them, and record counts.
         */
        SimpleReader textFile = new SimpleReader1L(textFileName);
        while (!textFile.atEOS()) {
            String line = textFile.nextLine();
            if (!counted.contains(line)) {
                int bucket = mod(hashCode(line), hashTableSize);
                counts[bucket]++;
                counted.add(line);
            }
        }
        textFile.close();
        /*
         * Report results.
         */
        out.println();
        out.println("Bucket\tHits\tBar");
        out.println("-------\t----\t---");
        for (int i = 0; i < counts.length; i++) {
            out.print(i + "\t" + counts[i] + "\t");
            for (int j = 0; j < counts[i]; j++) {
                out.print("*");
            }
            out.println();
        }
        out.println();
        out.println("Total:\t" + counted.size());
        in.close();
        out.close();
    }

}
```