```java
import java.util.HashSet;

/**
 * Simple class to experiment with the Java Collections Framework and how it
 * compares with the OSU CSE collection components.
 *
 * @author Put your name here
 *
 */
public final class JCFExplorations {

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

    /**
     * Raises the salary of all the employees in {@code map} whose name starts
     * with the given {@code initial} by the given {@code raisePercent}.
     *
     * @param map
     *            the name to salary map
     * @param initial
     *            the initial of names of employees to be given a raise
     * @param raisePercent
     *            the raise to be given as a percentage of the current salary
     * @updates map
     * @requires [the salaries in map are positive] and raisePercent > 0
     * @ensures <pre>
     * DOMAIN(map) = DOMAIN(#map)  and
     * [the salaries of the employees in map whose names start with the given
     *  initial have been increased by raisePercent percent (and truncated to
     *  the nearest integer); all other employees have the same salary]
     * </pre>
     */
    public static void giveRaise(components.map.Map<String, Integer> map,
            char initial, int raisePercent) {
        assert map != null : "Violation of: map is not null";
        assert raisePercent > 0 : "Violation of: raisePercent > 0";

        Iterator<Pair<String, Integer>> itr = map.iterator();
        Pair<String, Integer> object = null;

        while (itr.hasNext()) {
            object = itr.next();

            if (object.key().charAt(0) == initial) {
                int newValue = object.value() + (object.value() * raisePercent / 100);
                map.replaceValue(object.key(), newValue);
            }
        }

    }

    /**
     * Raises the salary of all the employees in {@code map} whose name starts
     * with the given {@code initial} by the given {@code raisePercent}.
     *
     * @param map
     *            the name to salary map
     * @param initial
     *            the initial of names of employees to be given a raise
     * @param raisePercent
     *            the raise to be given as a percentage of the current salary
     * @updates map
     * @requires <pre>
     * [the salaries in map are positive] and raisePercent > 0 and
     * [the dynamic types of map and of all objects reachable from map
     *  (including any objects returned by operations (such as
     *  entrySet() and, from there, iterator()), and so on,
     *  recursively) support all optional operations]
     * </pre>
     * @ensures <pre>
     * DOMAIN(map) = DOMAIN(#map)  and
     * [the salaries of the employees in map whose names start with the given
     *  initial have been increased by raisePercent percent (and truncated to
     *  the nearest integer); all other employees have the same salary]
     * </pre>
     */
    public static void giveRaise(java.util.Map<String, Integer> map,
            char initial, int raisePercent) {
        assert map != null : "Violation of: map is not null";
        assert raisePercent > 0 : "Violation of: raisePercent > 0";

        java.util.Iterator<Entry<String,Integer>> itr = map.entrySet().iterator();

        while (itr.hasNext()) {
            Entry<String, Integer> object = itr.next();

            if (object.getKey().charAt(0) == initial) {
                int newValue = object.getValue()
                        + (object.getValue() * raisePercent / 100);
                map.replace(object.getKey(), newValue);
            }
        }
    }

    /**
     * Increments by 1 every element in the given {@code Set}.
     *
     * @param set
     *            the set whose elements to increment
     * @updates set
     * @ensures <pre>
     * DOMAIN(map) = DOMAIN(#map)  and
     * [set is the set of integers that are elements of #set incremented by 1]
     * </pre>
     */
    public static void incrementAll(components.set.Set<NaturalNumber> set) {
        assert set != null : "Violation of: set is not null";

        NaturalNumber currentNum = null;
        components.set.Set<NaturalNumber> newSet = set.newInstance();
        int setSize = set.size();

        for (int i = 0; i < setSize; i++) {
            currentNum = set.removeAny();
            currentNum.increment();
            newSet.add(currentNum);
        }

        set.transferFrom(newSet);
    }

    /**
     * Increments by 1 every element in the given {@code Set}.
     *
     * @param set
     *            the set whose elements to increment
     * @updates set
     * @requires <pre>
     * [the dynamic types of set and of all objects reachable from set
     *  (including any objects returned by operations (such as iterator()), and
     *  so on, recursively) support all optional operations]
     * </pre>
     * @ensures <pre>
     * DOMAIN(map) = DOMAIN(#map)  and
     * [set is the set of integers that are elements of #set incremented by 1]
     * </pre>
     */
    public static void incrementAll(java.util.Set<NaturalNumber> set) {
        assert set != null : "Violation of: set is not null";

        Iterator<NaturalNumber> itr = set.iterator();
        java.util.Set<NaturalNumber> newSet = new HashSet<NaturalNumber>();
        int setSize = set.size();
        NaturalNumber currentNum = null;

        for (int i = 0; i < setSize; i++) {
            currentNum = itr.next();
            currentNum.increment();
            newSet.add(currentNum);
        }

        set = newSet;

    }

}

```