**Exercise for Set**

* Implement set as Hashset, LinkedHashset and treeset . Use all methods e.g. add, remove, addall,removeall,retainall, isSubset, isempty, size and soon. Implement a iterator over a set.
* Test the website domain name against a list of blocked website domain name. suppose following is the list of blocked domains stored in text file called blockeddomains.txt

dontgothere.com

ohno.org

badstuff.com

badstuff.org

badstuff.net

whatintheworld.com

notinthislifetime.irg

letsnot.com

eeewwwwww.com

* + Create a Domainblocker class which keeps the track of blocked domain and checks against them as needed. Create a treeset for blocked domain set. Read the website name from text file and add website name to treeset. Create a Boolean method to see if specified website name has been blocked. @Param is website name and @returns true if name is in your treeset otherwise return false.
  + Create a DomainChecker class. This class creates a instance of DomainBlocker class and then allows the user to repeatedly read a website name and checks to see if domain is blocked or not
* In mathematics, several operations are defined on sets. The union of two sets A and B is a set that contains all the elements that are in A together with all the elements that are in B. The intersectionof A and B is the set that contains elements that  are in both A and B. The difference of A and B is the set that contains all the elements of A **except** for those elements that are also in B.

Suppose that A and B are variables of type set in Java. The mathematical operations on A and B can be computed using methods from the *Set* interface. In particular: A.addAll(B) computes the *union* of A and B; A.retainAll(B) computes the *intersection* of A and B; and A.removeAll(B) computes the *difference* of A and B. (These operations change the contents of the set A, while the mathematical operations create a new set without changing A, but that difference is not relevant to this exercise.)

For this exercise, you should write a program that can be used as a "set calculator" for simple operations on sets of non-negative integers. (Negative integers are not allowed.) A set of such integers will be represented as a list of integers, separated by commas and, optionally, spaces and enclosed in square brackets. For example: [1,2,3] or [17, 42, 9, 53, 108]. The characters +, \*, and - will be used for the union, intersection, and difference operations. The user of the program will type in lines of input containing two sets, separated by an operator. The program should perform the operation and print the resulting set. Here are some examples:

Input Output

------------------------- -------------------

[1, 2, 3] + [3, 5, 7] [1, 2, 3, 5, 7]

[10,9,8,7] \* [2,4,6,8] [8]

[ 5, 10, 15, 20 ] - [ 0, 10, 20 ] [5, 15]

To represent sets of non-negative integers, use sets of type *TreeSet<Integer>*. Read the user's input, create two *TreeSets*, and use the appropriate *TreeSet* method to perform the requested operation on the two sets. Your program should be able to read and process any number of lines of input. If a line contains a syntax error, your program should not crash. It should report the error and move on to the next line of input. (Note: To print out a *Set*, A, of *Integers*, you can just say System.out.println(A). I've chosen the syntax for sets to be the same as that used by the system for outputting a set.)