Computing and Algorithms I
Project 5 Sets and Arrays

CS-101 **Fall, 2017**

In this project you will be implementing sets via a Set class. You will implement sets using Set objects. For this project you will write at least two classes, a Set class, and a "Driver" class or "Main" class. The Main class, or Driver class, is the class which contains the main method. The Set class will have only storage for a single set, and its methods are operations on sets. You will use an array of int as the storage mechanism to store the elements in the Set object. In the Main or Driver class, you will have an array variable to reference 100 Set objects (this array variable can be a local variable of some method or a class variable).

1 Set Operations

You will implement the following operations, whose signatures are shown here, as methods of Set objects:

- Set() - a constructor with no parameters, this will instantiate, or construct, a new empty set object.
- makeEmpty() - the set is made empty. Note that this does not construct a new empty set, it just removes all the elements in this set.
- isEmpty() - return the value `true` for an empty set, `false` for a set with elements.
- add(`int`) - this method will add an int value to the set which owns the add method. Note that if this Set object already has that int value, this Set object will not change. That is, duplicate values are not stored. (The set {1,2,1} is exactly the same set as {2,1}).
- remove(`int`) - this method will remove the specified int value from the set which owns the remove method. Handle the case of removing the last element in the set appropriately. Note that if the int value is not already in this set, this method does nothing.
- elementOf(`int`) - this boolean method returns `true` if the `int` value is in the set which owns the elementOf method.
- size() - this method returns the number of elements in this set.
- union(Set) - this method returns a new Set object containing all elements in the union of this Set object and the argument Set object.

- intersection(Set) - this method returns a new Set object containing all elements in the intersection of this Set object and the argument Set object. Note that this may be the empty set.
- setDifference(Set) - this method returns a new set object containing all elements in this Set object which are not also in the argument Set object.
- toString - this method returns a String listing all the elements in the set. The String returned should be of the form "{2,1}".

Note that there is no method in the Set class which calls a print method.

2 Input/Output

You will be reading input from a file. The name of that file will be supplied as a command line argument, in particular, as args[0]. All your output will be written to a file as well. The name of the output file will be supplied as a command line argument, args[1].

Each line of your input file will correspond to an operation on a set. The first character of the line will specify the operation, the remaining values on the line will specify values for that operation. The input line commands will be:

- C num

  C will mean construct an empty set. The num will represent a number from 0 to 99 and that is the set to make empty. For example

  C 50

  set number 50 is to be constructed as a new, empty set. A line of output will be written, but not from the Set class, stating that set number 50 has been constructed and is empty. Note that this number (50) is the slot number of the Set object stored in the array of Set objects in main.

- I num

  check to see if set numbered num is empty. For example

  I 10

  check set number 10. Print a line of output, but not in the Set class, stating whether set number 10 is empty or not. If element 10 of your Set array is null, print a message stating that there is no set number 10. Note that if the element of the Set array is null (in this example, element 10), there is no set to check for the condition empty.

- S num

  determine the number of elements in set numbered num. For example

  S 17

check set number 17. Print a line of output, but not in the Set class, stating that set number 17 contains size elements, where size is the number of elements in set number 17. Note that if set number 17 is empty, the size is 0. Also note that if element 17 of your Set array is null, there is no set; in that case print a message stating that there is no set number 17.

- X num

  The set num will be made an empty set. For example

  X 27

  set number 27 will be made empty, no matter how many elements were in the set to begin with. This does not construct a new set, it merely removes all elements from this set, if there is a Set reference at slot 27 of the Set array. If element 27 of your Set array in main does not contain a Set reference (if the value is null), this command will not do anything. Print a message to the output file stating that set number 27 has been emptied, or that there was no set to empty.

- A setNum value

  Add the value to set numbered setNum. For example

  A 9 234

  add the value 234 to set number 9. Note that if 234 is already in that set, the value will not be stored again. If your Set array has the value null in slot 9, print a message stating that there is no Set to add a value to.

- R setNum value

  Remove the value from set numbered setNum. For example

  R 8 522

  remove the value 522 from set number 8. If set 8 does not have a value 522, this does not change the set. If 522 is the only element in set 8, set 8 is to be made empty. If your Set array has the value null in slot 8, print a message stating that there is no Set to remove a value from.

- F setNum value

  Find the value as an element of set numbered setNum. For example

  F 7 123

  find whether or not 123 is an element of set 7. A line of output will be written to state if 123 is in the set or not. If your Set array has the value null in slot 7, print a message stating that there is no Set to in which to find a value.

- U set1 set2 set3

  Take the union of set1 and set2, set3 will be the union set. For example

  U 5 6 7

Take the union of sets 5 and 6, create set 7 to be the union of sets 5 and 6. If slot 5 or slot 6 of your Set array contains the value null, print a message stating that you cannot take the union due to a nonexisting set. Note that the sets 5 and 6 are not changed in any way in this operation.

- N set1 set2 set3

  Create set3 to be the set intersection of sets set1 and set2. For example:

  N 8 9 10

  create set 10 to be the intersection of sets 8 and 9. If slot 8 or slot 9 of your Set array contains the value null, print a message stating that you cannot take the intersection due to a nonexisting set. Note that the sets 8 and 9 are not changed in any way in this operation.

- D set1 set2 set3

  Create set3 to be the set difference of sets set1 and set2. For example:

  D 11 4 2

  Create set 2 to be the set formed by removing elements of set 4 from set 11. If slot 11 or slot 4 of your Set array contains the value null, print a message stating that you cannot take the set difference due to a nonexisting set. Note that the sets 11 and 4 are not changed in any way in this operation.

- P setNum

  Display, or print, the elements of set setNum. For example:

  P 17

  prints the elements in set 17. Note that your program will accomplish this by printing the value returned by the set object's `toString()` method. If slot 17 of your Set array is null print a message stating that there is no set to print.

- M setnum value1 value2 ... valuen

  Create set setNum and add values value1, value2, ..., valuen to the set. For example

  M 93 1 55 32 191

  Creates the set numbered 93 as an empty set, and then adds the values 1, 55, 32, and 191 to that set.

- # message

  # means comment, your program will print this line of input as is with no changes.

3 Requirements

You will write a correct, modularized and conforming program to read commands from the input file and write output to the output file as described above. This project has a design part and an implementation part.

4 Procedure

For this project, you will do the following:

1. Create a design document. The document will have a UML class diagram for each class in your solution. Note that there must be at least two classes in your solution. Each of these classes will have methods. Each method will have a data table and an algorithm. The algorithms will combine to solve the problem stated in this document.

2. Create a test file containing enough lines of input to create at least 10 sets and test each of the operations. There will be enough print commands to show that the operations work correctly.

3. Write code which satisfies the design and conforms to the requirements.

4. Test your code sufficiently to establish that the code satisfies the requirements.

5. Run your program on your test file.

5 Design

Your design documentation must include the following:

1. A UML class diagram for each class. (15 points)

2. A UML class interaction diagram. (10 points)

3. A data table for each class containing instance variables. (5 points)

4. Your design will have multiple methods, all properly indicated in the UML class diagrams.

5. An algorithm and data table for each method (except accessor methods). (10 points)

6 Code

The code will be graded as follows:

1. Program satisfies requirements. (25 points)

2. Program has good style including small methods, good variable names, etc. (10 points)

3. All required comments are included in code (including algorithms and data tables) (10 points)

4. Input file is constructed to easily show that program is correct. (10 points)

5. Output file is created from running program on input file. (5 points)

7 Deliverables

Your project is due at the beginning of class on Tuesday of week 10 (December 5). Be sure to submit the zipped Java code, input and output files, (zipped into project5.zip) into the Blackboard area project5b. Your design (in the original file(s) and the pdf file(s) you create from it) will be uploaded into the Blackboard area for project5a.

If you turn in the project early you will receive bonus points (5 points if turned in by Monday December 4).