2XB3 Assignment 1 – Implementing Abstract Data Types

Department of Computing and Software
McMaster University
Instructor: Dr. Reza Samavi

Posted: January 31st, 2020

Due on: **February 14, *2020* at 23:59** (strict deadline)

1. Assignment Problem Part 1
1.1 An ADT for Set Operations

This assignment requires you to create your own abstract data type for set operations on a set composed of String elements. The ADT needs to be an **immutable type** with the following methods (Assume R and S are two finite sets):

Set Union	Return the union of this set with another set (i.e. R ∪ S).
Set Intersection	Return the intersection of this set with another set (i.e. R ∩ S).
Set Difference	Return the difference of this set with another set (i.e. R − S, the set of all elements in R that do not belong to S) (note: *this* set is R).
Set Product	Return the Cartesian product (cross product) of this set by another set (i.e. R X S, the set of all possible pairs of concatenated elements of the form *rs* where *r* ∈ R and *s* ∈ S.
isEqual	Are these sets equal? Return Boolean. Remember, sets are **reflexive, symmetric, and transitive**. By definition, a set cannot have duplicate elements.
isSubset	Is this set a subset of another set? (e.g., thisSet ⊆ otherSet)
getCount	Return the number of elements in this set.
toString	Return the elements of the set in the following format: {$elem_1$, $elem_2$, …, $elem_n$}

1.2 Implementation:
Before you start implementing this ADT, you should create a project in your workspace in Eclipse called 'CAS2xb3_A1_*lastname_initials'*. Replace *lastname* with your last name and *initials* with your initials. Your entire implementation should be included in this project.

You will need to:

1. Build an API. What would the API look like? The above table looks similar, but you have to decide on the return type and parameters of each method. Also the above list of operations is not comprehensive and you should add common operations such as adding an element to, deleting an element from the set. Include the API as a pdf document in the root folder of the project. If you made any specific design decision or made any assumption you can include them in this pdf file too.

2. Implement the abstract data type that you create. Keep in mind the required instance variables, constructors, accessor methods and instance methods.
3. Make sure encapsulation is conserved.

Note: *You cannot use Java Set Data type for this assignment.*

The ADT should be called "Set" and contain an array of Strings. Each element of the array is considered an element of the set. Assume the input sets will have a maximum of 20 elements consisting of *unit strings* (strings of length one). Hint: *you may end up producing larger sets with longer strings*.

Your task is to create the proper constructors, accessor methods, and instance methods based upon the table provided above. Proper commenting and formatting is required.

1.3 Testing:

Upon completion of the ADT, create a new Java class to test the ADT that you have implemented. This class should be called testSet. Within this class, you need to create test methods to ensure that the ADT is working properly. Use the following method signatures to create test methods that test each of the methods in your ADT. Each test method must test not only the expected behaviour, but also edge cases (for example what happens if you try to add a duplicate element). You may implement the beahviour of your methods in different ways when an edge case occurs (for example throwing an exception, skipping adding an element when adding a duplicate element). You should use the API pdf to document your design decisions.

Note: Using Junit testing is optional.

public static void testUnion()
public static void testIntersection()
public static void testDifference()
public static void testProduct()
public static void testIsEqual()
public static void testIsSubset()
public static void testGetCount()
public static void testToString()

The test cases used in the test methods must be **read from a txt file** called **input.txt** located in the root directory of your Eclipse project. Each input set will appear on a new line where elements of the set are separated by a comma. Each test case should display appropriate message at the beginning, during, and at the end of execution. The results of the test methods must be **written to a text file** in the root directory called **output.txt**.

Sample Output

Entering testUnion…
Test case 1 passed.
Test case 2 passed.

…
testUnion complete.

Assignment 1 Submission

In every class that you create, include the following header with the appropriate information:

```
/*            Student Information
 *            ------------------
 *            Student Name: Lastname, Firstname
 *            Student Number: 000000000
 *            Course Code: CS/SE 2XB3
 *            Lab Section: 00
 *
 *            I attest that the following code being submitted is my own individual
work.
 */
```

In order to submit your Eclipse project to the relevant assignment dropbox in Avenue, you first need to save everything and then export your project.

1. In Eclipse, right-click on the name of the project, select Export->General->Archive File.
2. Ensure that just your project has a check-mark beside it, and select a path and filename to export the project to. Ensure that your export project has a file extension of '.zip'.
   IMPORTANT: You MUST export the FULL Eclipse project. Submitting individual files (e.g. **Java**/class files) will NOT be counted towards your submission. Click 'Finish' to export.
3. Verify the zip file by opening it and ensuring that it has the same folder structure as in Eclipse (it may have some extra files or folder such as 'bin', which is okay).
4. Go to Avenue and upload your zipped project to 'Assignment 1 Submission' Dropbox.

Assignment 1 Marking

Assignment 1 is worth 15% of the course marks. Your grade for this assignment will be determined based on the following criteria:

- **A submitted assignment that does not compile or run gets 0 credit.**
- A solution that runs but is partially correct gets partial credit (depending on the progress towards a full solution).
- Providing adequate, concise, and meaningful comments throughout your code is part of the solution grade (i.e., a piece of code that correctly solves a problem without (or with inadequate) comments will score less than a well-commented piece of code that does the same).
- The work you submit must be your own. If you include libraries from any sources other than your own or from the course material (course lecture notes and lab notes/instructions) you must acknowledge them and explicitly give proper credit with meaningful and concise comments inside your code (when using methods from the external libraries). The included libraries should not be a substantial part of your assignment. Copying full or partial codes from other resources and including them inside your code is strictly forbidden. Your work will be

checked for plagiarism to account for this. Both copying assignments and allowing others to copy your assignments are strictly forbidden and will be treated as an academic offence.

- Every hour after an assignment deadline 2% will be deducted from the assignment mark. After 48 hours, the student will get 0 credit for the missing assignment.
- If you MSAF the assignment, you will get 3-day extension.

VERY IMPORTANT: ONLY THE LAST SUBMISSION WILL BE GRADED. LATE SUBMISSION WILL BE APPLIED BASED ON THE TIMESTAMP OF THE LAST SUBMISSION.