Assignment 2

**Due:** Friday, February 5 by 11:59 PM

**Objective**

This assignment should help you gain practice with creating and editing basic Java classes based on given speciﬁcations.

**Task**

Do the following three exercises each in a different ﬁle. Your ﬁlenames should be

* IntegerSet.java
* Fraction.java

**Each ﬁle should have a comment including your name at the top of the ﬁle. Each ﬁle should also have appropriate comments throughout the program. When adding to a pre-supplied ﬁle, clearly indicate, using comments, which parts are your additions.**

**Integer Set**

Filename: IntegerSet.java

* Create class IntegerSet in its own ﬁle
* An IntegerSet object holds integers in the range 0-100
* This is represented by an array of booleans. Array element *i* is set to true if the integer *i* is in the set and false otherwise.
* Create these methods for the class (these should be the only public methods) :

IntegerSet()

public IntegerSet union(IntegerSet iSet)

public IntegerSet intersection(IntegerSet iSet)

public IntegerSet insertElement(int data)

public IntegerSet deleteElement(int data)

public boolean isEqualTo(IntegerSet iSet)

public String toString()

* The constructor (no arguments) initializes the array to represent the “empty set” (no integers in the set)
* Method union creates and returns a new set that is the set-theoretic union of the two existing sets (the calling object and the parameter). An element is in the union if it is in either of the two starting sets
* Method intersection creates and returns a new set that is the set-theoretic intersection of the two existing sets. An element is in the intersection if it’s in both of the starting sets.
* Method insertElement adds the argument(an integer) to the set (the calling object) and should also return that set (this allows calls to be cascaded)
* Method deleteElement removes the argument from the set and should also return the set, to allow cascading
* Method isEqualTo returns true if the two sets are equal (if they have all the same elements) and false otherwise
* Method toString returns a string containing the set elements as a list of numbers in ascending order, separated by spaces. Include only the elements present in the set. Use “—” to represent an empty set

**Fraction Class**

Filename: Fraction.java

Begin with this Fraction class, posted on Canvas within the Fraction.java file within the Code files for Lecture 4, and add the following features:

* You will create methods with the following signatures:

public Fraction simplify()

public Fraction add(Fraction f)

public Fraction subtract(Fraction f)

public Fraction multiply(Fraction f)

public Fraction divide(Fraction f)

* The simplify method will return a simpliﬁed version of the calling object. This method should return a new Fraction (simpliﬁed), but not change the original one. The fractions in the form 0/N should have a simpliﬁed form of 0/1. Any other fraction has the usual mathematical deﬁnition of “simpliﬁed form”. This will require ﬁnding the GCD of the numerator and denominator. One useful algorithm for doing so is Euclid’s algorithm (the Euclidean algorithm).
* Methods add, subtract, multiply, divide should take in a Fraction as a parameter and perform the given computation between the calling object and the parameter object (The calling object is always the ﬁrst operand). The result of each operation should always be a fraction returned in simpliﬁed form. Example calls: f1.add(f2) means to return the value *f1+ f2*, f1.divide(f2) means to return the value *f1/f2*.
* In divide, if an attempt is made to divide by a fraction with the numerator 0, default the result to 0/1. This division is actually undeﬁned, but we need to return something from the method and this is a “sane” value
* Be sure that your new methods enforce the same rules on the data as the original methods do – the denominator must always be non-negative (negative fractions have the negative sign in the numerator) and the denominator must never be zero

**Testing**

I’ve provided a ﬁle to help you get started with testing. This is not a comprehensive set of tests (so make sure you do some of your own), but will get you started. Also, you will need to include the HW2Tester class (unchanged) in your jar ﬁle when you submit, as indicated at the end of this.

The HW2Tester.java ﬁle, which contains some tests for both the IntegerSet and Fraction classes, is posted on within Assignment 2 Canvas site.

**Sample Run of HW2Tester**

After set1.insertElement(10), set1 = 0 2 8 10

default IntegerSet is = ----

set1 = 0 2 4 6 8 10 12 95 100

set2 = 0 3 6 9 12

set1.union(set2) = 0 2 3 4 6 8 9 10 12 95 100

set1.intersection(set2) = 0 6 12

set1.deleteElement(2) = 0 4 6 8 10 12 95 100

set1.isEqualTo(set1) = true

set1.isEqualTo(set2) = false

Fraction tests:

4/6 simplified = 2/3

75/175 simplified = 3/7

-6/17 simplified = -6/17

f1 = 4/6

f2 = 75/175

f3 = -6/17

4/6 + 75/175 = 23/21

4/6 - 75/175 = 5/21

4/6 \* 75/175 = 2/7

4/6 / 75/175 = 14/9

75/175 + -6/17 = 9/119

75/175 - -6/17 = 93/119

75/175 \* -6/17 = -18/119

75/175 / -6/17 = -17/14

75/175 / 0/1 = 0/1

**Submitting**

Pack all of your ﬁles (class ﬁles and source code) into **a fully runnable JAR** ﬁle called hw2.jar (this will be discussed in class, see the links on the class website for more details). The main program that the jar ﬁle should execute is the unchanged HW2Tester program downloaded above. I should be able to run the HW2Tester main() method from your jar ﬁle with the command:

java -jar hw2.jar

Submit your jar ﬁle via the Canvas submission link for Assignment 2.