Principles of Object Oriented Programming - 2017/Spring Exercise 2

TAs in charge: Guy Danielli and Matan Kintzlinger.

Published:23/04/2017 **Due**: 07/05/2017

The assignment should be submitted in groups of two students.

General note: although it is possible to implement this assignment without objects, you must implement it in an object-oriented manner, according the principles learned in the lectures and practical sessions. In addition, you must write your programs according to the coding-standards document that is published on the Assignments page of the course website.

We advise you to read the whole assignment before starting to implement it.

Any question about the assignment should be asked in the assignment's forum.

1 General Description

The goal of this assignment is to practice the following concepts:

- Class design
- Collections
- Inheritance and substitution

In this assignment, you are required to implement a Set Calculator. The Set Calculator supports set operations such as *union*, *intersection*, *difference* and *Cartesian product* of two sets. The design should be general, and allow easy modification. To this end, you need to implement the following classes (Some of the classes may be Abstract or Interface). You can provide additional methods, if you wish.

- **Element**
  Represents a general element. This class should include the following operations:
  - ○ Element transformAdd(Numeric n)
    accepts a numeric argument and returns an element which is the sum of the current element with n.
  - ○ Element transformMul(Numeric n)
    accepts a numeric n and returns a new element which is the multiplication of the current element with n

Two classes that extend/implement Element

- ❑ **Numeric**
  This class supports the following operations:
  - Numeric transformAdd(Numeric n)
    accepts a numeric argument and returns a Numeric which is the sum of the current element with n.
  - Numeric transformMul(Numeric n)
    accepts a numeric n and returns a new Numeric which is the multiplication of the current numeric with n

  Two **classes that extend/implement** Numeric
  - **Real:** For real numbers.
  - **Rational:** For Rational numbers. A rational number is a number a/b, where and b are integers, b ≠0. It should be represented using two integer fields.

- **Set**

  A subclass of Element that represents a (finite) set of elements. This class supports the following operations:

  - Set insert(Element e)
    accepts an element e and returns a Set which is the current set with the argument e. (i.e. s1.insert(e) == s1 ∪ { e })

  - Set remove(Element e)
    accepts an element e and returns a Set which is the current set without e (i.e. s1.remove(e) == s1\{ e })

  - int size()
    Returns the cardinality of the set.

  - Set union(Set s)
    accepts a set argument s and returns a Set which is the union of the current set with the set s. (i.e. s1.union(s2) == s1 ∪ s2)

  - Set intersect(Set s)
    accepts a set argument s and returns a Set which is the intersection of the current set with the set s.(i.e. s1.intersect(s2) == s1 ∩ s2)

  - Set difference(Set s)
    accepts a set argument s and returns a Set which is the difference between the current set and s. (i.e. s1.difference(s2) == s1\s2)

  - Set power()
    returns a Set which is the powerset of the current set.

  - boolean contains(Set s)
    accepts a set argument s and returns true if the current set contains s
    (i.e. s1.contains(s2) == s2 ⊆ s1)

  - boolean member(Element e)
    accepts an element e and returns true if e is a member of the current set.

  - boolean deepExistence(Element e)
    accepts an element e and returns true if e is a member of the current set or is a member (recursively) of one of its member elements

  - Set transformAdd(Numeric n)
    accepts a numeric argument n and returns a Set which is the current set after recursively applying transformAdd with the argument n on all current sets’ elements.

❏ Set transformMul(Numeric n)
accepts a numeric argument n and returns a Set which is the current set after recursively applying transformMul with the argument n on all currents sets' elements

Note: Similar to the typing rules in most programming languages , the resulting type should be the more common one.
For example:

$\frac{4}{5} * 5.25 = 4.2$ **not** $\frac{21}{5}$

$\frac{1}{2} + 0.3 = 0.8$ **not** $\frac{4}{5}$

- Calculator: This class includes the main method. Inside the main method the user will be asked for input. The Calculator should support the method void **`calc(String instruction)`** that takes a user instruction (represented as String) , performs the requested instruction and returns the result.

In addition, every class should include getters and setters, toString. and equals methods The toString method should return a friendly representation of the object

2 Input and Output Formats

Input:

Element Form:

Rational Form: Rational number always appears in the form a / b (without spaces)

Real Form: Real numbers should appear with a dot if is needed.

Set Form: Sets appear in the form '{e1,e2,...,en}', (without spaces) where ei is a rational, a real or a set.

Commands Form:

```
size            <set>
contains        <set> <set>
member          <set> <element>
deepexistance   <set> <element>
equals          <element> <element>
insert          <set> <element>
remove          <set> <element>
union           <set> <set>
intersect       <set> <set>
difference      <set> <set>
```

```
power             <set>
transformAdd      <element> <numeric>
transformMul      <element> <numeric>
help
bonus
exit
```

The number of separation spaces is arbitrary. Your program should work for any number of separation spaces

Example:

```
deepExistance {1,2,{3},{4{5,6}}}         5
true
```

Output:

Rational: Similar to the input form. In the case that the the denominator equals to 1 (b =1), rationals should appear without the denominator part. In any case, the rationals should appear in reduced form.
Real and Set: Similar to their input form

3 Running Examples:

The program should never crash, even if a user tries to perform an illegal operation. Instead, in case of an exception, you should catch it, show the user an appropriate message saying that the input is incorrect, and close the program.

Example 1

```
>java -jar hw2.jar

Sets Calculator
==========================
>help
size                <set>
contains            <set> <set>
member              <set> <element>
deepexistance       <set> <element>
equals              <element> <element>
insert              <set> <element>
remove              <set> <element>
union               <set> <set>
intersect           <set> <set>
difference          <set> <set>
power               <set>
```

```
transformAdd        <element> <numeric>
transformMul        <element> <numeric>
help
bonus
exit

>size {1,2,3,4}
4

>size 2
Error: 2 is not a set!

>contains {1,2,3,4}            2
Error: 2 is not a set!

>contains {1,2,3,4}   {3}
True

>contains {1,2,3,4} {3
Error: cannot parse {3

>contains {1,2,{3},{4,{5,6}}} {3}
False

>member {1,2,3,4} 2
True

>member {,1,2,{3},{4,{5,6}}} 4
False

>deepExistance {1,2,3,4} 2
True

>deepExistance {1,2,{3},{4,{5,6}}} 5
True

>equals {1,2,3,4} {1,2,3,4}
True

>equals {1,2,3,4} {1,2,{3},{4,{5,6}}}
False

>exit
```

Example 2:

```
>java -jar hw2.jar

Sets Calculator
==========================
>insert {1,2,3,4} {}
{1,2,3,4,{}}

>add {1,2,3,4,{}} 3/8
{1,2,3,4,{},3/8}

>remove {1,2,3,4,{},3/8} {}
{1,2,3,4,3/8}

>remove {1,2,3,4,{},3/8} 5
{1,2,3,4,{},3/8}

>union {1,2,3,4} {3,6,7,8}
{1,2,3,4,6,7,8}

>union {1,2,3,4} {1,2,3,4}
{1,2,3,4}

>union {1,2,3,4} 3/8
Error: 3/8 is not a set!

>intersect {1,2,3,4} {3,6,7,8}
{3}

>difference {1,2,3,4} {3}
{1,2,4}

>difference {1,2,3,4} {1,2,3,4}
{}

>power {1,2,3,4}
{{},{1},{2},{3},{4},{1,2},{1,3},{1,4},{2,3},{2,4},{3,4},{1,2,3
},{1,2,4},{1,3,4},{2,3,4},{1,2,3,4}}

>power {}
{}

>exit
```

Example 3:

```
>java -jar hw2.jar

Sets Calculator
==========================
>transformAdd {1,2/4,3,4} 3/8
{1.375,7/8,3.375,4.375}

>transformAdd {1,2/4,3,4} 1.5
{2.5,2,4.5,5.5}

>transformMul {1,2/4,3,4} 3/8
{0.375,3/16,1.125,1.5}

>transformMul {1,2/4,3,4} 0.375
{0.375,0.1875,1.125,1.5}

>exit
```

4 Testing

In this assignment you are required to write unit tests for Set operations. For each of the operations below you are required to write at least one unit test.

- insert
- remove
- size
- union
- intersect
- difference
- power
- member
- contains
- deepExistence
- transformAdd
- transformMul

5 Bonus

After finishing the assignment, you can earn up to 5 more points by adding a bonus command called `bonus`, which prints an ASCII art of your choice. You will be scored for creativity and impressiveness.

(you can read about ASCII art here: https://en.wikipedia.org/wiki/ASCII_art)

Bonus Example:

```
>java -jar hw2.jar

Sets Calculator
==========================
>bonus
                                           /
                        _,.------....___,.' ',.-.
                     ,-'          _,.--"        |
                   ,'         _.-'              .
                  /   ,     ,'                   `
                 .   /     /                     ``.
                 |  |     .                       \.\
       ____      |___._.  |       __               \ `.
     .'    `---""       ``"-.--"'`  \               .  \
    .  ,            __               `              |   .
    `,'         ,-"'  .               \             |    L
   ,'          '    _.'                -._          /    |
  ,`-.    ,".   `--'                      >.      ,'     |
 . .'\'   `-'       __    ,  ,-.         /  `.__.-      ,'
 ||:, .           ,'  ;  /  / \ `        `.    .      .'/
 j|:D  \          `--'  ' ,'_  . .         `.__, \   , /
/ L:_  |                 .  "' :_;                `.'.'
.    ""'                  """""'                    V
 `.                                 .    `.   _,..  `
   `,_   .    .                _,-'/    .. `,'   __  `
    ) \`._        ___....----"'  ,'   .'  \ |   '  \  .
   /   `. "`-.--"'         _,' ,'     `---' |    `./  |
  .   _  `""'--.._____..--"   ,             '         |
  | ." `. `-.                /-.           /          ,
  | `._.'    `,_            ;  /         ,'          .
 .'          /| `-.        . ,'         ,           ,
 '-.__ __ _,','    '`-..___;-...__   ,.'\ ____.___.'
 `"^--'..'   '-`-^-'"--    `-^-'`.''"""""`.,^.`.--' mh

>exit
```

6 Submission Instructions and Requirements

You need to submit a **single** zip file called hw2.zip to the CS submission system which contains the following:

1. Part 1: Design - to be submitted in a file named **hw2.pdf**:

    (a) Define the components that take part in the system and their responsibilities.

    (b) Design an appropriate UML class diagram.

2. Part 2: implementation - you need to submit **your source code AND a jar file called hw2.jar**

    (a) Implement the program according to your design.

    (b) Your program should support input stream directly from the user (see examples above).

Good Luck!