Programming and Data Structures
Active Learning Activity 3: Abstract Classes and Interfaces

Activity Objectives

At the end of this activity, students should be able to:

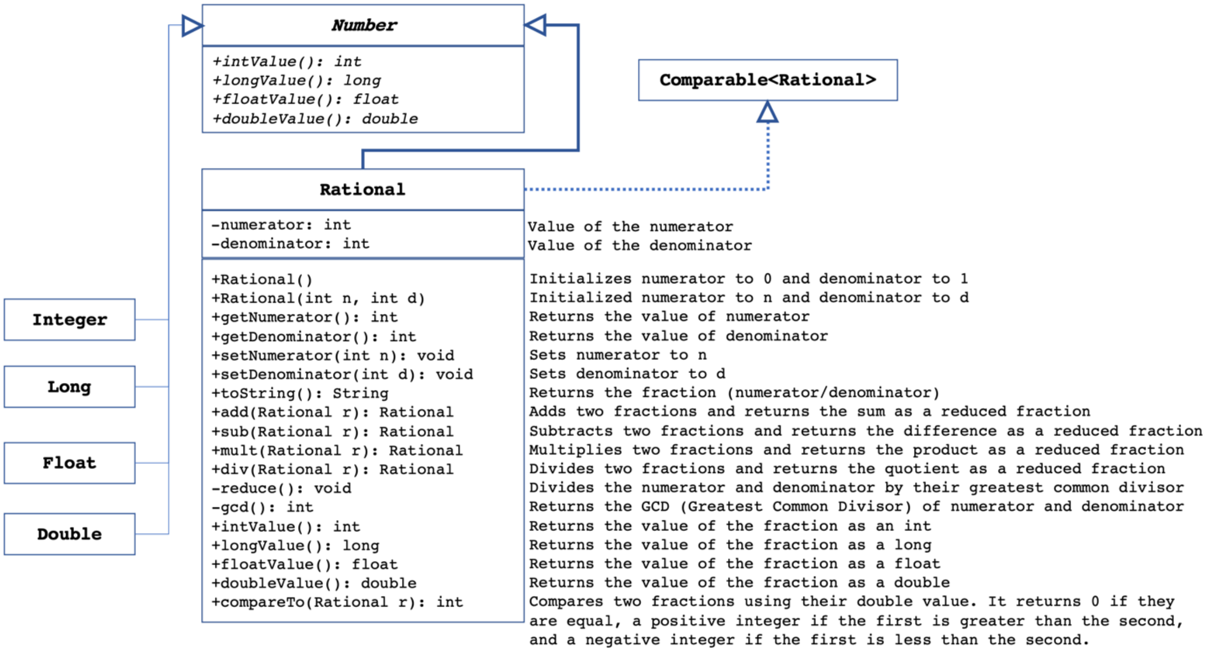
1. Use an abstract class to model common behavior between related classes
2. Create a concrete class ***Rational*** for type fraction that extends an abstract class and implement the abstract methods
3. Make the class ***Rational*** implement the interface ***Comparable*** to define how rational objects should be ordered
4. Write a Java program that manipulates fractions using the new type ***Rational***

Activity

You are asked to write a program that defines a new type for rational numbers (fractions) and perform basic arithmetic operations on fractions.

Your program should include the following:

1. The class ***Rational*** that extends the Java abstract class ***Number***. The two classes are described by the UML diagrams below. You do not need to create the class Number as it is already available in the package ***java.lang***. Note that the wrapper classes ***Integer***, ***Long***, ***Float*** and ***Double*** extend the abstract class ***Number***.



-numerator: int	Value of the numerator
-denominator: int	Value of the denominator
+Rational()	Initializes numerator to 0 and denominator to 1
+Rational(int n, int d)	Initialized numerator to n and denominator to d
+getNumerator(): int	Returns the value of numerator
+getDenominator(): int	Returns the value of denominator
+setNumerator(int n): void	Sets numerator to n
+setDenominator(int d): void	Sets denominator to d
+toString(): String	Returns the fraction (numerator/denominator)
+add(Rational r): Rational	Adds two fractions and returns the sum as a reduced fraction
+sub(Rational r): Rational	Subtracts two fractions and returns the difference as a reduced fraction
+mult(Rational r): Rational	Multiplies two fractions and returns the product as a reduced fraction
+div(Rational r): Rational	Divides two fractions and returns the quotient as a reduced fraction
-reduce(): void	Divides the numerator and denominator by their greatest common divisor
-gcd(): int	Returns the GCD (Greatest Common Divisor) of numerator and denominator
+intValue(): int	Returns the value of the fraction as an int
+longValue(): long	Returns the value of the fraction as a long
+floatValue(): float	Returns the value of the fraction as a float
+doubleValue(): double	Returns the value of the fraction as a double
+compareTo(Rational r): int	Compares two fractions using their double value. It returns 0 if they are equal, a positive integer if the first is greater than the second, and a negative integer if the first is less than the second.

2. Create the class Test with a main method to perform the following:
   a. **Part 1 (Manipulating Numbers)**
      i. Create an array of type ***`Number`*** and size ***`10`***.
      ii. Add a pair of random integers, long, float, double, and rational numbers to the array.
      iii. Display the list of numbers using the following methods: ***`toString()`***, ***`intValue()`***, and ***`doubleValue()`***.

   b. **Part 2 (Operations on numbers of type *`Rational`*)**
      i. Create an array of type ***`Rational`*** and size ***`8`***.
      ii. Create eight random ***`Rational`*** objects and store them in the array. Generate two random integers from 1 to 9 for the numerator and denominator of each ***`Rational`*** object.
      iii. Display the list of fractions stored in the array.
      iv. Display the sum of the first and second fractions.
      v. Display the difference between the third and fourth fractions.
      vi. Display the product of the fifth and sixth fractions.
      vii. Display the quotient of the division of the seventh fraction by the eighth fraction.
      viii. Use the method ***`java.util.Arrays.sort()`*** to sort the 8 fractions from the smallest to the largest.

   All fractions should be displayed in reduced form.

Test your program (see sample run below) and submit the files ***`Rational.java`***, and ***`Test.java`*** on **Github**. Make sure all your java files contain Javadoc comments.

Important note: ***`toString()`*** method in class ***`Rational`*** should consider the following:

- If the denominator is equal to ***`1`***, return only the numerator (fraction ***`2/1`*** should return ***`2`***)
- If the numerator is ***`0`***, return ***`0`*** (***`0/2`*** should return ***`0`***)
- If the denominator is negative, the negative sign should appear only on the numerator (***`1/-2`*** should return ***`-1/2`***)
- If the numerator is equal to the denominator, return ***`1`***

Here is a sample run of the program:

------------------------ Sample Run ----------------------------

```
List of numbers:
Value                    int value      double value
585                      585            585.00
111036                   111036         111036.00
902.82294                902            902.82
83712.82156522696        83712          83712.82
3/2                      1              1.50
91                       91             91.00
118433                   118433         118433.00
676.6151                 676            676.62
66087.5350023578         66087          66087.54
1/2                      0              0.50

Original list of fractions:
2
4/7
8/7
1
1/6
3/7
2/3
7/5

Operations on fractions:
2 + 4/7 = 18/7
8/7 − 1 = 1/7
1/6 ∗ 3/7 = 1/14
2/3 / 7/5 = 10/21

Sorted list of fractions:
1/6
3/7
4/7
2/3
1
8/7
7/5
2
```