Abstract Methods and Interfaces

CMPT220L
Due on Apr 29, 2022 by 11:59 PM
Points: 100

Problems

1. Define a class named `Time` for encapsulating a time. The class contains the following:

    - A data field of the `long time` that stores the elapsed time since midnight, Jan 1, 1970.
    - A no-arg constructor that constructs a `Time` for the current time.
    - A constructor with the specified hour, minute, and second to create a `Time`.
    - A constructor with the specified elapsed time since midnight, Jan 1, 1970.
    - The `getHour()` method that returns the current hour in the range 0-23.
    - The `getMinute()` method that returns the current minute in the range 0-59.
    - The `getSecond()` method that returns the current second in the range 0-59.
    - The `getSeconds()` method that returns the elapsed total seconds.
    - The `toString()` method that returns a string such as "1 hour 2 minutes 1 second" and "14 hours 21 minutes 1 second".
    - Implement the `Comparable<Time>` interface to compare this `Time` with another one based on their elapse seconds. The `compareTo` method returns the difference between this object's elapse seconds and the another's.

    Write a test program that produces the following sample run:

```
Enter time1 (hour minute second): 331 34 674
19 hours 45 minutes 14 seconds
Elapsed seconds in time1: 1194314
Enter time2 (elapsed time): 93889345
16 hours 22 minutes 25 seconds
Elapsed seconds in time2: 93889345
time1.compareTo(time2)? -92695031
time3 is created as a clone of time1
time1.compareTo(time3)? 0

Enter time1 (hour minute second): 1 2 3
1 hour 2 minutes 3 seconds
Elapsed seconds in time1: 3723
Enter time2 (elapsed time): 193032
5 hours 37 minutes 12 seconds
Elapsed seconds in time2: 193032
time1.compareTo(time2)? -189309
time3 is created as a clone of time1
time1.compareTo(time3)? 0
```

2. The `Animal` class is given in Listing 13.7 in the book. Modify this class by implementing the `Comparable` interface and `Cloneable` interface. Add the `weight` property in the `Animal` class with getter and setter methods. The two animals are compared based on their weights. Use the following `main` method to run your test program.

```
public static void main(String[] args) {
    Animal[] list = new Animal[5];
    list[0] = new Chicken();
    list[0].setWeight(4.5);
    list[1] = new Tiger();
    list[1].setWeight(46.6);
    list[2] = new Chicken();
    list[2].setWeight(1.5);
    list[3] = (Animal)(list[0].clone());
    list[3].setWeight(7.5);
    list[4] = (Animal)(list[1].clone());

    java.util.Arrays.sort(list);
    for (int i = 0; i < list.length; i++) {
        System.out.println("weight: " + list[i].getWeight());
    }
}
```

3. *(Parse rational numbers)* Add the following method in the `Rational` class defined in Listing 13.13.

```
public static Rational parseRationalNumber(String s)
```

The method returns a `Rational` object from a string that represents a rational number. Here are some examples of parsing rational numbers:

```
Rational r1 = Rational.parseRationalNumber("3 / 15");
Rational r1 = Rational.parseRationalNumber("-3/15"); // This is OK
Rational r2 = Rational.parseRationalNumber("34"); // Denominator is 1
```

Write a test program that prompts the user to enter two rational numbers as strings and displays their addition. Here are some sample runs:

```
Enter the first rational number: 3/ 21
Enter the second rational number: 2
1/7 + 2 = 15/7
```

Submission

Make sure you create one Java file per project. Place your `.java` files under the corresponding folder in your local copy of the GitHub repository, commit and push it to the remote repository. Make sure that the professor has access to the repository (`jfac65-marist`).

```
cmpt220lastname\
   hw10\
        Problem1.java
        Problem2.java
        Problem3.java
```