README.md

CSE-11 Programming Assignment 4

Due Date: Wednesday, April 28, 11:59PM Pacific Time

Learning Goals

- practice writing interfaces and classes

Collaboration

Different assignments in this course have different collaboration policies. On this assignment, you **cannot** share code publicly on Piazza or with any other students in the course. If you need to share code, ask in a private Piazza post or in 1-on-1 hours. Still *do* ask any public code questions about lecture code, quizzes that are past, or other conceptual questions, just no code from this PA.

Resubmission/Late Policy

- We will not accept this PA late.

- If you did not submit this PA or did poorly (<75%), you can fix your PA and resubmit it to the PA resubsmission assignment in Gradescope to earn a maximum of 75% the points.
- If you scored higher than 75% on the origianl PA, we may grade the resubmission, but will not change your original grade.
- The resubmission will be open for 2 weeks after the PA's original due date.

You will upload *several* Java files for this assignment, detailed below.

Tweets

A Twitter-specific pattern is writing a "thread" by replying to one's own Tweets repeatedly. In a file `Tweets.java`, write an `interface` called `Tweet` with three methods:

- `public boolean isStartOfThreadBy(String author);`
- `public int totalLikes();`
- `public String unrollThread();`

Then, write two classes:

- `TextTweet`, implements `Tweet` and has three fields:
  - `contents`, a `String`
  - `likes`, an int
  - `author`, a `String`

  This class should implement the methods as follows:
  - `isStartOfThreadBy` should return `true` when the given author is the same as the `author` of this `TextTweet`, `false` otherwise.
  - `totalLikes` should return the number of likes on this `TextTweet` object
  - `unrollThread` should return a string in the following format:

```
<author>
<n> likes
<content>
```

where `<author>` is replaced by the author's name, `<n>` is replaced by the number of likes on the `TextTweet`, and `<content>` is replaced by the contents of the Tweet. The string should end in a new line (`"\n"` character).

- `ReplyTweet`, which **should implement** `Tweet` and has four fields:
  - `contents`, a `String`
  - `likes`, an int
  - `author`, a `String`
  - `replyTo`, a `Tweet`

  This class should implement the methods as follows:

  - `isStartOfThreadBy` should return `true` when the given author is the same as the `author` of this `ReplyTweet` **and** the `replyTo` tweet is also the start of a thread by the same author.
  - `totalLikes` should return the total number of likes on this `ReplyTweet` object **plus** the total likes of its `replyTo` Tweet. For example, a thread of tweets that is 4 replies long should sum the likes on all 4 of those tweets.
  - `unrollThread` should return a string in the following format:

    ```
    <replyTo content>
    <author>
    <n> likes
    <content>
    ```

    where the bottom three parts are the same format as in `TextTweet`, and the first part is the unrolled version of the `replyTo` `Tweet`. Like for `TextTweet`, this should end in a new line character.

You can, but don't have to, use an abstract class to avoid duplicated work as you see fit. Add constructors as appropriate to initialize the fields on objects of these classes (whether or not you use an abstract class).

For each method, write at least two tests for it in a class called `Tweets` using the `Tester` library. A “test” is a use of `checkExpect` that checks the results of the method call against an expected value.

Since there are 6 total method implementations, you should have at least 12 tests. Tests are graded manually, your implementation is graded automatically.

Numbers

This code will go in the file `ExamplesNumber.java`, any tests in a class called `ExamplesNumber` that you add to that file.

We saw in our reading that representing fractional numbers like 0.6 with doubles can be fraught. Some languages and libraries do support exact fractions, and we can implement classes that act like them in Java. We won't be able to use the built-in `+` and `*` operators, because these are only defined for numbers and strings, but we can define methods for the operations we care about. We can represent numbers with an interface:

```
interface Number {
  int numerator();
  int denominator();
  Number add(Number other);
  Number multiply(Number other);
  String toString();
  double toDouble();
}
```

(We could specify more methods, but for the purposes of this assignment, these six will be sufficient.)

Your task is to create two classes that implement the interface above. One should be called `WholeNumber` and represent whole integers (including negative numbers). The other should be called `Fraction` and represent mixed numbers.

`WholeNumber` should have:

- A field `int n` and a constructor that takes a single `int`
- An implementation of all the methods above.
  - `numerator` should return the value of `n`
  - `denominator` should return `1`
  - `add` should return a new `Number` that represents adding this whole number to the one provided as an argument. Note that the argument could be either a `Fraction` or a `WholeNumber`
  - `multiply` should return a new `Number` that represents multiplying this whole number to the one provided as an argument. Note that the argument could be either a `Fraction` or a `WholeNumber`
  - `toString` should return the value of `n` as a `String`, so if `n` is `500`, it should return `"500"`

    - `toDouble` should return the value of `n` as a `double`

`Fraction` should have:

- A field `int n` representing the numerator
- A field `int d` representing the denominator
- An implementation of all the methods above:
    - `numerator` should return the value of `n`
    - `denominator` should return the value of `d`
    - `add` should return a new `Number` that represents adding this fraction to the one provided as an argument. Note that the argument could be either a `Fraction` or a `WholeNumber`
    - `multiply` should return a new `Number` that represents multiplying this fraction by the one provided as an argument. Note that the argument could be either a `Fraction` or a `WholeNumber`
    - `toString` should return a `String` in the format `"n/d"` where `n` and `d` are the corresponding fields. So if `n` and `d` were `1` and `2`, this should be `"1/2"`
    - `toDouble` should return the value of `n/d` as a `double`. So if `n` is 1 and `d` is 2, this should return `0.5`

A reminder about arithmetic and fractions:

$$n = \frac{n}{1}$$

$$\frac{n}{d_1} + \frac{m}{d_2} = \frac{d_1 m + d_2 n}{d_1 d_2}$$

$$\frac{n}{d_1} \cdot \frac{m}{d_2} = \frac{nm}{d_1 d_2}$$

Exploration

At the end of the `ExamplesNumber` class in a place marked clearly with a comment that says `// Exploration`, write code to perform four calculations:

1. The result of `0.1 + 0.2 + 0.3` using built-in `double` arithmetic in Java
2. The result of `0.1 + (0.2 + 0.3)` using built-in `double` arithmetic in Java
3. The result of (1) using your exact fractions, showing the result via `toString()`
4. The result of (2) using your exact fractions, showing the result via `toString()`

Submission

Then you will submit *all* of your files to the `pa4` assignment on Gradescope:

- On the Gradescope upload screen, you can keep clicking “Browse Files” to select more than one file for your upload; you can select them one at a time or use your operating system's multi-select (Shift-Click usually works) to select them all and drag them onto the upload area, or other options that you find that work.
- You can also make a zip archive of all of your files and upload them all at once if you prefer.

We will automatically grade the correctness of the methods and classes you write. Tests and exploration sections will be graded manually. In addition, we may give you feedback on any part of the code, including automatically graded parts, that we want you to respond to after grading.

Extra Challenges (not for credit)

Challenge (not required for credit): Many fractions, like $$2/4$$ or $$27/6$$, are not in their simplest form. Make it so that the constructor for `Fraction` always creates a fraction object with numerator and denominator in their most reduced form.

Challenge (not required for credit): Create a `ReplyTweet` that is a reply to itself. Do you think this is possible on Twitter?

FAQ

1. Why is the autograder producing this error “…” for me?

- As a general reminder, it would help us a lot if your provide your submission link when it's a Gradescope-related question (just copy the URL from the URL bar when looking at your submission and other students won't be able to see your code from a Gradescope link).
- In addition, you should have your own tests, and you should write your own tests and try things out to make sure you understand what your code does before submitting it. You can share your tests with us privately on Piazza and we can discuss what's happening in them as a way to debug as well.

2. My code does not run on Gradescope because it’s still WIP. How can I check that the part I finished is correct?

- For the methods you haven’t finished yet, you can put implementations for them that intentionally return the wrong answer, like an empty string or false, to make it so all

the tests will run.

3. VSCode is red underlining the `import tester` statement. Is this an error?

- Before trying to assess the error, try compiling your code and running it. Sometimes VSCode will erroneously underline imports that actually do work. If the code does not compile, make sure that the file that is trying to import the tester is in the same folder as `tester.jar`.

4. For `unrollThread` methods in `Tweets.java`, I cannot pass the tests on Gradescope, but the expected result and my program result look exactly the same.

- This is probably because of a missing or extra newline character or space character that is hard to spot with our eyes. The autograder has been updated to give a hint:" (Hint) If your result looks the same as the reference output but still gets an error, there may be extra space characters or newline characters in your result. You may try to print the length of your output String locally to see if it matches your expectation."

5. I wrote test methods with the `Tester`, but `./run` is telling me that no tests ran.

- `Tester` methods have to start with "test" at the beginning! like `boolean testAdd(Tester t) { .... }`

6. Am I required to write tests in the `ExamplesNumber` class?

- We won't grade tests written there, but we encourage you to write them to gain confidence in your code!

7. Should the `unrollThread` method of `ReplyTweet` start with just the contents of the `replyTo` tweet, or the entire unrolled tweet?

- It should contain the entire unrolled tweet, not just the `contents` field.

Releases

No releases published
Create a new release

Packages

No packages published
Publish your first package

Contributors 2

 **ayonbiswas** Ayon Biswas

 **ucsd-miranda** Greg Miranda

Languages

Shell 54.3% **Batchfile** 39.2% **Java** 6.5%