CS 262 | **Due: Wednesday, November 30 at 12 PM**
Fall 2022 | **20 points**
Lab 9
11/17/2022

Introduction: For this lab, you will get some good practice using generic structures in Java. The basic idea of what your program has to do is described as follows.

Let the user type in six input values. If the user types in a valid `double` for the first value, then your program may assume that the remaining five inputs will all be valid `double`s. Otherwise, if the first input value is not a valid `double`, then your program must read in all six input values as `String` types.

Your program should interpret the six input values as three pairs: the first two form a pair, the third and fourth form a pair and the fifth and sixth form a pair. Output the "largest" of these three pairs. Your program must determine the larger of the two pairs (a, b) and (c, d) as follows:

- If $a < c$, then (a, b) is less than (c, d).
- If $a > c$, then (c, d) is less than (a, b).
- If $a = c$, and $b < d$, then (a, b) is less than (c, d).
- Otherwise, (c, d) is less than (a, b).

Note that this is basically the same as the lexicographical ordering scheme. Also note that in the above list, "<" effectively represents testing whether a call to `compareTo` returns a negative value, assuming a is comparable with c, and b is comparable with d.

Here's a sample run, where the first line is user input and the second line is the output of the program:

```
1 1 2 1 1 99
(2.0, 1.0)
```

Here's another sample run:

```
purple monkey dish washer foo bar
(purple, monkey)
```

We will define a generic `MyPair` class to represent a pair of values, and then we'll have a `Lab9` driver program that handles the input and output functionality.

Getting started:

1. Create a new BlueJ project called "Lab9". Make sure you save your project to some kind of permanent storage, e.g., OneDrive.
2. Within your Lab9 project, create two classes: `MyPair` and `Lab9`.
3. Delete all the starter code in both of these classes so that you're starting fresh with empty classes.

Requirements: Ensure that your lab satisfies the following requirements:

1. Since `MyPair` is supposed to represent a pair of objects, make `MyPair` generic using two type parameters.
2. `MyPair` must encapsulate two `private` fields: `key` and `value`. These must be declared using the appropriate generic type and they each must be comparable, not necessarily to each other, but to other objects of the same type. Hint: you can bound the type of each generic type parameter separately and using the `extends` notation that we discussed in class.
3. Have a `MyPair` constructor that accepts two parameters and uses them to initialize the `key` and `value` fields, respectively.
4. Implement a proper `toString` method in `MyPair`. Mimic the format shown in the above sample output.
5. Make it so the `MyPair` class is itself comparable, restricted to being compared to other `MyPair` objects, where the comparing `MyPair` and the compared `MyPair` both have the same types of `key`s and `value`s. At this point, the class header for `MyPair` is probably getting pretty crazy (that's fine).
6. Implement the `compareTo` method of `MyPair` according to the above lexicographical ordering specification.
7. In the `Lab9` class, implement a generic `maxOfThreePairs` method that accepts three `MyPair` objects and returns the largest of the three, according to the ordering scheme that we previously defined. Basically, you should be calling the `compareTo` method on `MyPair` objects to determine which one is the largest.
8. Also in the `Lab9` class, have a `main` method that reads in six input values.
9. Don't prompt for the input.
10. You must use a `new Scanner(System.in)` to read the six input values from the user and you are only allowed to call the `Scanner` methods `nextDouble`, or `next`.
11. If the first input value happens to represent a valid `double` value, then you must assume that all six input values will be valid `double`s.
12. If the first input value doesn't represent a valid `double` value, then you must assume that all six input values will be `String`s.
13. You must use `Exceptions` to differentiate between reading in `double`s and `String`s. Hint: Try to read in six `double`s, and if you get an `InputMismatchException` on the first one, then read in six `String`s.
14. Interpret the six input values as three pairs: the first two values should be stored in a `MyPair` object, the third and fourth values should be stored in a `MyPair` object, and the fifth and six values should be stored in a `MyPair` object.

15. All `MyPair` objects must be instantiated correctly using proper generic notation.
16. Print out the largest of the three pairs. For this step, you must call your `maxOfThreePairs` method and print out its return value.
17. You must use the diamond notation somewhere in your program.

Submission: When you are done, zip up your entire BlueJ project folder. Upload your .ZIP file to the **Lab9** dropbox on Canvas. After you have uploaded the file, double-check to ensure your file was uploaded correctly. It is your responsibility to ensure your submission was done correctly. Programs that are not uploaded correctly are worth 0 points.