Storing Multiple Choice Quizzes

The text files:
- Each text file corresponds to answers for a multiple choice quiz.
- Each line contains one answer.
- For each quiz, answers.txt represents the correct answers.

MCQuiz Class

```java
public class MCQuiz {
    private String studentName;
    private String quizName;
    private List<String> correctAnswers;
    private List<String> studentAnswers;

    public MCQuiz(String filename) {
        // Process file
    }

    public String getName() {
        // Return student name
    }

    public String getStudent() {
        // Return student name
    }

    public int numberCorrect() {
        // Calculate correct answers
    }
}
```

We would like to do the following tasks:
- Print out the quizzes in worst-to-best order (e.g. sort the quizzes)
- Collect all quizzes of each particular student together and display them (still from worst-to-best)

Sorting Strings

Strings have a method called compareTo that works like < does on ints.
If we have two strings:
- String hello = "hello" and String bye = "bye"
To do the test "hello < bye", we do the following:
- Write what we want: hello < bye
- Subtract the right from both sides: hello - bye < 0
- Replace the subtraction with compareTo: hello.compareTo(bye) < 0
That's it!

Sorting A String List
```java
if (list.get(j).compareTo(list.get(minIndex)) < 0) {
    minIndex = j;
}
```

Comparing MCQuizzes

```java
public static void sortMCQuizList(List<MCQuiz> list) {
    for (int i = 0; i < list.size(); i++) {
        int minIndex = i;
        for (int j = i; j < list.size(); j++) {
            if (list.get(j).compareTo(list.get(minIndex)) < 0) {
                minIndex = j;
            }
        }
        String temp = list.get(minIndex);
        list.set(minIndex, list.get(i));
        list.set(i, temp);
    }
}
```
How do sort and TreeSet work?

How do sort and TreeSet KNOW the ordering?
If you were implementing sort for a type T, what would you need to be able to do with T and T?

We would need to be able to COMPARE a and b

That's just an interface! Java calls it "Comparable".

Comparable
The Comparable interface allows us to tell Java how to sort a type of object:

```
public interface Comparable<E> {
    1 public int compareTo(E other);
}
```

This says, "to be Comparable, classes must define compareTo".

MCQuiz: Defining compareTo

Attempt #1

```
public class MCQuiz implements Comparable<MCQuiz> {
    1 ... 3 public int compareTo(MCQuiz other) {
    4     return this.numberCorrect() - other.numberCorrect();
    5 }
```

This doesn't work, because if we have a quiz where someone got 1/10 and another where someone else got 1/5, we treat them as the same.

Attempt #2

```
public class MCQuiz implements Comparable<MCQuiz> {
    1 ... 3 public int compareTo(MCQuiz other) {
    4     return (double)this.numberCorrect()/this.correctAnswers.size() -
        (double)other.numberCorrect()/other.correctAnswers.size();
    5 }
```

This won't even compile! We need to return an int.

Attempt #3

```
public class MCQuiz implements Comparable<MCQuiz> {
    1 ... 3 public int compareTo(MCQuiz other) {
    4     Double thisPer = (double)this.numberCorrect()/this.correctAnswers.size();
    5     Double otherPer = (double)other.numberCorrect()/other.correctAnswers.size();
    6     return thisPer.compareTo(otherPer);
```

This still doesn't work, because it doesn't take the names of the students into account.
In particular, if two students both get 1/10 on a quiz, our compareTo method says "it doesn't matter which one goes first".

Attempt #4

```
public class MCQuiz implements Comparable<MCQuiz> {
    1 ... 3 public int compareTo(MCQuiz other) {
    4     Double thisPer = (double)this.numberCorrect()/this.correctAnswers.size();
    5     Double otherPer = (double)other.numberCorrect()/other.correctAnswers.size();
    6     if (result = 0) { result = this.studentName.compareTo(other.studentName); }
    7     return result;
```

This still doesn't work, but it's not as clear why. Let's try the second task.

Printing The Quizzes in Order

Client Code to Print The Quizzes

```
List<MCQuiz> quizzes = createQuizzes(2);
// First, let's get a sorted list of the quizzes
Collections.sort(quizzes);
for (MCQuiz quiz : quizzes) {
    System.out.println(quiz);
}
```

This doesn't work, because Java doesn't know how to sort MCQuizes.

Comparable
The Comparable interface allows us to tell Java how to sort a type of object:

```
public interface Comparable<E> {
    1 public int compareTo(E other);
}
```

This says, "to be Comparable, classes must define compareTo".

Comparable: Tricks #1 & #2

```
int Fields
If we have a field int x in our class, and we want to compare with it, our code should look like:
```
```
public class Sample implements Comparable<Sample> {
    1 public int compareTo(Sample other) {
    2     return ((Integer)other).compareTo(this.x);
    3 }
```

In other words, just use the existing compareTo on the field in the class!

Object Fields
If we have a field Object x in our class, and we want to compare with it, our code should look like:

```
public class Sample implements Comparable<Sample> {
    1 public int compareTo(Sample other) {
    2     return x.compareTo(other.x);
    3 }
```

Grouping the Quizzes by Student

What data structure should we use to group the quizzes? A Map!

Map Question: "Which quizzes were taken by this student?"

Keys: Strings (the student names)

Values: Set<MCQuiz> (all the quizzes that student took).

```
List<MCQuiz> quizzes = createQuizzes(2);
Map<String, Set<MCQuiz>> quizzesByStudent = new TreeMap<>();
for (MCQuiz quiz : quizzes) {
    // We want to loop over all the quizzes, adding thes one by one
    String name = quiz.getStudent();
    if (!quizByStudent.containsKey(name)) {
        quizByStudent.put(name, new TreeSet<>(new MCQuiz()));
    }
    quizByStudent.get(name).add(quiz);  
}
```

Now, we want to print out the quizzes student by student:

```
for (String student : quizByStudent.keySet()) {
    System.out.println(student + " : " + quizByStudent.get(student));
}
Grouping the Quizzes by Student

The output looks like this:

```
>> BarbaraHarris: [BarbaraHarris (quiz1): 3/11, BarbaraHarris (quiz0): 4/11]
>> JessicaHernan: [JessicaHernan (quiz1): 1/11, JessicaHernan (quiz0): 2/11]
>> TeresaHall: [TeresaHall (quiz0): 4/11]
```

Why does Teresa only have one quiz? She scored the same on both of her quizzes and compareTo said they were the same!

Final Attempt

```java
public class MCQuiz implements Comparable<MCQuiz> {
    ...
    public int compareTo(MCQuiz other) {
        Double thisPer = (double) this.numberCorrect() / this.correctAnswers.size();
        Double otherPer = (double) other.numberCorrect() / other.correctAnswers.size();
        int result = thisPer.compareTo(otherPer);
        if (result == 0) {
            result = this.studentName.compareTo(other.studentName);
        }
        if (result == 0) {
            result = this.quizName.compareTo(other.quizName);
        }
        return result;
    }
}
```

Lesson: When you write compareTo, make sure that a.compareTo(b) == 0 exactly when a.equals(b)

Some Comparable Tips

- Understand multi-level structures
- Use the most general interface as possible
- When implementing compareTo, make sure to use all the fields that make it different (to put another way: a.compareTo(b) == 0 exactly when a.equals(b))
- Remember that inside classes, you can look at the fields of other instances of that class