Programming Assignment 2

`TripSetter`

Overview

The project manager at *DukeDash* who reviewed your work on the `SpeedSetter` is very happy and wants you to create a new product called `TripSetter` that both adds capabilities and makes the code more reusable.

As with `SpeedSetter`, `TripSetter` will be given the current speed in mi/hr and convert it to km/hr. However, it will also be given the distance traveled during the current trip (in ft) and the duration of the trip (in hr) and must calculate the average trip speed in both km/hr and mi/hr.

Specifications

The product must comply with the following specifications:

1. The main class must be named `TripSetter`.
2. When executed, command-line argument 0 must contain a `String` representation of the current real-valued speed (in mi/hr), argument 1 must contain a `String` representation of the real-valued trip distance (in ft), and argument 2 must contain a `String` representation of the real-valued trip duration (in hr).
3. The product must have a class named `Converter` that performs all of the necessary unit conversions. In particular, this class must have the following methods:

   ```
   public static double feetToMiles(double feet)

   public static double mphToKPH(double mph)
   ```

   The purpose of each method should be apparent from its name.
4. The `Converter` class must have appropriate “class constants” that are used in its methods.

5. The product must display the current speed in km/hr, the average speed (also called the trip speed) in km/hr, and the average speed in mi/hr.

Existing Components

`Dashboard`

As you know from your earlier work on `SpeedSetter`, the `Dashboard` class contains the graphical user interface for an in-vehicle electronic dashboard. In addition to the methods you have used in the past, it also has the following method:

```
public static void setTripSpeed(double kph, double mph)
```
Displays the average trip speed in both km/hr and mi/hr.

Recommended Process

1. Read and understand the entire assignment.
2. Create a directory/folder (e.g., named `pa2`) that will hold all of the files for this assignment.
3. Copy `Text.class` and `Dashboard.class` into the directory you just created.
4. Implement the `mphToKPH()` method in the `Converter` class. (Hint: This should mostly involve moving code from the `SpeedSetter` class to the `Converter` class.)
5. Implement the part of the `TripSetter` class that uses argument 0 and displays the speed in mph.
6. Test the `mphToKPH()` method using the same values you used when testing the `SpeedSetter` class.
7. By hand, convert the following values from feet to miles:

   0.0, 1000.0, 1320.0, 5280.0, 2138958.0
8. Implement the `feetToMiles()` method in the `Converter` class.
9. Test the `feetToMiles()` method and debug it if necessary.
10. By hand, convert the following distances (in feet) and durations (in hr) to mi/hr:

Distance	Duration
200.0	1.00
5280.0	1.00
5280.0	2.00
316800.0	1.25

11. Implement the part of the `TripSetter` class that uses arguments 1 and 2 and displays the average speed in mph and kph.
12. Test this portion of the `TripSetter` class.
13. Submit your implementation of `Converter.java` and `TripSetter.java` in a file named `pa2.zip` using Autolab. Do not include any other files in the `.zip` file.

Hints

1. Be careful about units. Specifically, pay attention to the units of the command-line arguments and the units of the values to be displayed in the `Dashboard`.
2. If you have questions about the assignment, ask them before you start writing any code. In other words, make sure you completely understand what you need to do before you start typing.
3. If you're confused about the relationship between the `TripSetter` class (the main class) and the `Converter` class (the utility class), you should look at the lecture on "Methods and Modularity", specifically the example with the `AreaCalculator` class (the main class) and the `Geometry` class (the utility class). Remember, however, that the `main()` method in the `AreaCalculator` uses the `JMUConsole` class to present information to the user while the `main()` method in the `TripSetter` class must use the `Dashboard` class to present information to the user.

Grading

Your code will first be graded by Autolab and then by the Professor. The grade you receive from Autolab is the maximum grade that you can receive on the assignment.

Autolab Grading

Your code must compile (in Autolab, this will be indicated in the section on "Does your code compile?") and all class names and method signatures comply with the specifications (in Autolab, this will be indicated in the section on "Do your class names, method signatures, etc. comply with the specifications?") for you to receive any points on this assignment.

Autolab will then grade your submission as follows:

Conformance to the Course Style Guide:	**20 points** (Partial Credit Possible)
Correctness:	**80 points**
`Converter`:	50% (Partial Credit Possible)
`TripSetter`:	50% (Partial Credit Possible)

Manual Grading

After the due date, the Professor may manually review your code. At this time, points may be deducted for inelegant code, inappropriate variable names, bad comments, etc.