Programming Assignment 3

`DetailedTripSetter`

Overview

DukeDash has sold a large number of `TripSetter` applications. However, they're worried that they are not going to have a continuing source of revenues. So, they want to create a new product that has additional features. They are calling this upgrade the `DetailedTripSetter`.

The `DetailedTripSetter` will be given the current speed in mi/hr, the distance traveled during the current trip (in ft) and the duration of the trip (in hr). In addition to the information displayed by the `TripSetter`, it will also display the length of the trip in km, and the length of the trip in "whole miles" and "remaining feet".

Specifications

The product must comply with the following specifications:

1. The main class must be named `DetailedTripSetter`.
2. When executed, command-line argument 0 must contain a `String` representation of the current real-valued speed (in mi/hr), command-line argument 1 must contain a `String` representation of the real-valued trip distance (in ft), and command-line argument 2 must contain a `String` representation of the real-valued trip duration (in hr).
3. The modified `Converter` class must have the following additional methods:

   ```
   public static double milesToKilometers(double miles)
   public static int feetToWholeMiles(int feet)
   public static int feetToRemainingFeet(int feet)
   ```

   The purpose of the first method should be apparent from its name. The other two can be used to convert a distance in feet to a distance in "whole miles" and "remaining feet".

For example, the distance 100 feet corresponds to 0 whole miles and 100 remaining feet, the distance 5280 feet corresponds to 1 whole mile and 0 remaining feet, and the distance 5380 feet corresponds to 1 whole mile and 100 remaining feet.

4. As in the `TripSetter`, the product must display the current speed in km/hr, the average speed in km/hr, and the average speed in mi/hr. The `DetailedTripSetter` must also display the trip length in kilometers, and the trip length in whole miles and remaining feet.

Existing Components

`Dashboard`

As you know from your earlier work, 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 has the following method:

```
public static void setTripDistance(double km, int wmi, int rft)
```

Displays the trip distance in both kilometers (passed in using the parameter named `km`) and whole miles (passed in using the parameter named `wmi`) and remaining feet (passed in using the parameter named `rft`) . Note that this method does not ensure that the values are consistent, it simply displays the values it is given.

Recommended Process

1. Read and understand the entire assignment.
2. Create a directory/folder (e.g., named `pa3`) that will hold all of the files for this assignment.
3. Copy `Text.class` and `Dashboard.class` into the directory you just created.
4. Copy `Converter.java` into the directory you just created. You may use either your implementation or the solution that was provided to you.
5. Copy `TripSetter.java` into the directory you just created. You may use either your implementation or the solution that was provided to you.
6. Rename `TripSetter.java` to `DetailedTripSetter.java` and change the name of the class in the file.
7. By hand, convert the following values from miles to kilometers:

   0.0, 7.0, 75.0
8. Implement the `milesToKilometers()` method in the `Converter` class.
9. Test the `milesToKilometers()` method (using the tests above) and debug it if necessary.
10. By hand, convert the following distances (in feet) to whole miles and remaining feet:

0, 37, 9387, 10559, 10560, 10561

11. Implement the `feetToWholeMiles()` method in the `Converter` class.
12. Test the `feetToWholeMiles()` method (using the tests above) and debug if necessary.
13. Implement the `feetToRemainingFeet()` method in the `Converter` class.
14. Test the `feetToRemainingFeet()` method (using the tests above) and debug if necessary.
15. Implement the part of the `DetailedTripSetter` class
16. Test the `DetailedTripSetter` class.
17. Submit your implementation of `Converter.java` and `DetailedTripSetter.java` in a file named `pa3.zip` using Autolab. Do not include any other files in the `.zip` file.

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`:	75% (Partial Credit Possible)
`DetailedTripSetter`:	25% (All or Nothing)

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.