CSE 142 - Su 02
Homework 5
Assigned: Wednesday, July 31
Due: Wednesday, August 7, BEFORE MIDNIGHT

** General Comments about the Homework **

All homework is turned in electronically. Go to the class web site and use the link on the homework page to do the turnin. Don't be late! Late homeworks will not be accepted.

This file describes the Homework 5 Programming Project. There are other files that describe the practice problems and the graded problems.

** Homework 5 Programming Project **

This project is based on the King County Metro Transit bus system. As you saw in class, Metro and the University of Washington make quite a bit of information about the buses and their locations available in real-time. In this homework project, you will be defining Java classes that we can use to track and manipulate that information.

Your task is to implement two new classes and upgrade another.

++ List of classes ++

As provided to you, homework 5 does not work because it is missing two key classes which you will implement. The overall concept of the program is that it creates a number of TransitBus objects, each representing one bus in the TransitSystem, and then it analyzes the list of buses to find out various interesting tidbits such as which bus is closest to a Landmark like the UW HUB, how many route 7 buses are running, and so on. Each TransitBus has a Vehicle Identification Number (vin), a route number, and a Location. Each Landmark has a Location and a name.

The files involved in this program are the following.

* Metro.java *

This is the main program. It contains the "static void main(String[] arg)" method where Java starts running the program. This method creates a new TransitSystem object and then adds several TransitBus objects to the system. Then it uses TransitSystem methods to process the list of buses.

Metro.java is complete as written. During development, you may want to put comment characters "//" in front of the lines that use parts of the code that you haven't written yet. For example you could hide all the lines that refer to Landmark objects (the second half of the main method) until after you have finished your TransitBus class and gotten that working.

* TransitSystem.java *

This class manages the list of buses and performs various analysis actions. The constructor is provided to you, as well as a few methods to add TransitBuses to the list and print the list. Your job (described below) is to write additional methods to process the list.

* TransitBus.java *

TransitBus is the primary class of interest in this program. TransitBus objects have a unique vehicle identification number, they have a current

Location, and they are assigned to a route. The Vehicle and LocatedVehicle classes implement some of these capabilities. Your job (described below) is to implement a functioning TransitBus class.

* Landmark.java *

A Landmark is an object that implements the Locatable interface and has a name. There is an example of how to do this provided in LocatedVehicle, and your implementation of LandMark can follow that example very closely. Your job (described below) is to implement a functioning Landmark class.

* Locatable.java *

This file defines the Locatable interface. Any class that wants to provide the capability of being located on the surface of the Earth needs to implement this class. Locations are defined in terms of latitude and longitude. Any class that implements Locatable must have a reference to a Location object, and provide some methods for accessing and using that Location. The required methods are defined in this Locatable interface. You can look at LocatedVehicle.java for an example of how to implement this interface. No change is needed in this class.

* LocatedVehicle.java *

This file defines the LocatedVehicle class that implements the Locatable interface. In order to do this, it extends the Vehicle class and adds the methods that are required by the Locatable interface. Read this class carefully, because the Landmark class that you must implement is very similar to LocatedVehicle. No change is need in this class.

* Vehicle *

This file defines the basic Vehicle class. This is where the Vehicle Identification Number is defined and stored. This class also implements the Comparable interface, which makes it possible to sort Vehicles (and all subclasses of Vehicle) by vehicle ID number. No change is needed in this class.

* Location.java *

This file defines the Location class that manages the details associated with storing a location on the face of the Earth in terms of latitude and longitude. There is a simple constructor, and a couple of methods to update the actual position after the object is created. This is also a method that calculates the distance on the surface of the Earth between this Location object and any other Location object. This method is used in implementations of the distanceTo(Location other) method that is required in the Locatable interface. Again, refer to LocatedVehicle.java for an example of how to do this. No change is needed in this class.

++ Project Requirements ++

1. Download the cse142-hw5.zip file and unzip it. The project skeleton is in the directory hw142\hw5. Your task is to open the project and create two new class files (TransitBus.java and Landmark.java) and update another class file to add capabilities (TransitSystem.java).

2. You can use BlueJ if you like, or you can use any other program editor and compiler combination. I have provided simple batch files that compile and run this project outside of BlueJ. In order to run the program from inside BlueJ, just right-click on the Metro class and select the main method. This will

start the program running and it will produce some printed output. If you run the program from the command line using runMetro.bat, it will print the output to the screen window.

3. There are examples to guide you in the implementation of required features. The TransitBus class is a simple extension of the LocatedVehicle class. The Landmark class is very similar to the LocatedVehicle class. Landmark implements the Locatable interface and so does LocatedVehicle. Landmark does not extend any other class (other than Object). And finally, the methods that you should add to TransitSystem are similar to the printBusList() method that is provided.

4. The specific implementation requirements are as follows.

4a. Write a new class called TransitBus that extends LocatedVehicle and adds the capability of storing a route number for each bus. There is not much code required to do this.

You need to have a constructor for the TransitBus class that takes an integer vehicle identification number, a double latitude value,a double longitude value, and an integer route number. Your constructor should call the constructor of its superclass (namely LocatedVehicle) using the syntax "super (vin, lat, lon)" that I showed in class on Monday July 29. After doing that, your constructor also needs to save the route number in a private instance variable defined in TransitBus.

For TransitBus you also need to write a getRoute() method that returns the route number as an integer, and a setRoute method that takes a new integer route number as a parameter and updates the instance variable in the TransitBus object.

The javadoc comments for this class are given in TransitBus.txt.

4b. Write a new class called Landmark that implements the Locatable interface and also stores a name for this particular Landmark. There is not much code required to do this.

You need to have a constructor that takes a String name, a double latitude and a double longitude. The constructor stores the name of the Landmark in a private String instance variable, and also creates a new Location(lat,lon) and stores a reference to that in a private Location instance variable. The class must implement all the methods of the Locatable interface. Refer to LocatedVehicle for examples of exactly how to do this.

The javadoc comments for this class are given in Landmark.txt.

4c. Update the methods in the TransitSystem class so that all the functions needed by the main method in Metro.java are satisfied.

You need to implement printBusOnRouteList, findClosestBus, and findClosestBusOnRoute. The comments describing these methods are given in TransitSystem.txt.

5. There is a batch file named makeDoc.bat that will run the javadoc tool for you and generate your program documentation in subdirectory doc. You might want to run it and look at the output, starting with index.html. The comments in your code are the source for these web pages.

6. There is a batch file names runMetro.bat that will run the program for you. The output will be something like this when the program is running correctly:

```
C:\home\finson\cse142\hw5>java -classpath . Metro

The buses currently in the system are:
TransitBus: vin: 4000
TransitBus: vin: 3227
TransitBus: vin: 4106
TransitBus: vin: 4010
TransitBus: vin: 4037
TransitBus: vin: 3501

The buses currently on route 43 are:
TransitBus: vin: 4000
TransitBus: vin: 4010

The bus closest to the Hub is: TransitBus: vin: 4037

The route 43 bus closest to the Hub is: TransitBus: vin: 4010

C:\home\finson\cse142\hw5>pause
Press any key to continue . . .
```