## SmartCard

You are being asked to write a program to help manage a certain subway system which charges its passengers through SmartCards. Immediately, it seems we need to design a program with a Station class, a SmartCard class, and perhaps a Passenger class.

Let’s write the Station class. Each station has a name and is in one of five zones numbered 1 through 5. We can guess that the Station class has two private fields, a two-arg constructor, and two accessor methods. It is good practice to write every class with a default constructor, modifier methods, and a toString method. Your Station class MUST have the following public methods, though you are always welcome to write additional helper methods:

* getZone() must return an int representing the zone.
* getName() must return a String representing the name of the Station.
* a constructor with 2 parameters, a String for the name and an int for the zone.

Here are some example calls to the constructors of Station:

Station downtown = new Station("Downtown", 1);

Station center = new Station("Center City", 1);

Station uptown = new Station("Uptown", 2);

Station suburbia = new Station("Suburb", 1);

Write the Station class on a separate sheet of paper.

Let’s plan the SmartCard class. SmartCards keep track of the money on the card, the zone in which zone the passenger boarded, and whether the SmartCard was presented (true or false) to board or to exit the system.

When people board a train and present their SmartCard, the boarding information is saved on their SmartCard. When they exit, they present their SmartCard and are charged for their trip according to the following rates:

* Travel within the same zone is charged $0.50.
* In addition, for each zone outside the starting zone travelers go through, they pay $0.75.
* To get from any zone to another zone, travelers must pass through all zones in between. For example, to travel from zone 2 to zone 5, a traveler must pass through zones 3 and 4.

For example, a trip from zone 1 to another station in zone 1 (or zone 3 to zone 3, or zone 4 to zone 4) is $0.50. A trip from zone 1 to zone 2 would cost $1.25 ($0.50 for travel in zone 1 plus $0.75 to go from zone 1 to 2). A trip from zone 4 to zone 1 would cost $2.75 ($0.50 in zone 4 plus $0.75 for each of zones 3, 2, and 1).

Your SmartCard class MUST have the following public methods at a minimum (you are always welcome to write additional helper methods):

* a constructor with only one double parameter that loads the card with money. Set the other fields to the standard default values: 0 for numbers, null for objects, empty string for String, false for boolean.
* addMoney(double d) – this method adds to the balance carried by this SmartCard.
* getBalance()– this method returns the balance on the SmartCard as a String formatted as dollars and cents (Ex: $10.00).
* isBoarded()– this method returns a boolean value if the traveler is on board or not
* board(Station s)– this method saves the station where the traveler boards. If the traveler tries to board without having previously exited, it prints "Error: already boarded?!" and returns. If the traveler has less than $0.50 (minimum fare), it prints "Insufficient funds to board. Please add more money." and returns. Do not use System.exit(0). If it passes those two checks, this method prints the name of the boarding station and the balance on the SmartCard, as shown in the run below.
* cost(Station s) – calculates and returns the cost to exit at this station; the details of the calculation are described above.
* exit(Station s) – if the traveler tries to exit without having previously boarded, it prints "Error: Did not board?!" and returns. If the cost exceeds the balance on the SmartCard, it prints "Insufficient funds to exit. Please add more money." and returns. If it passes those two checks, the method updates all fields in the SmartCard, prints the names of the two cities, the cost, and the balance on the SmartCard, as shown in the run below.
* the shell contains 3 accessor methods getMoneyRemaining(), getBoardedAt(), getIsOnBoard()which you must implement following the directions in the shell, even though you will not use them in your code. They will be used by Grade-It to check your code.

For this program, assume that the system will always work and that no one will lose a SmartCard between boarding and exiting. The shell is SmartCard\_Driver.java. Notice that three classes, Station, SmartCard, and the driver, are all contained in this one file.

Here are sample runs for five different SmartCards:

Boarded at Center City. SmartCard has $20.00  
From Center City to Suburb costs $2.75. SmartCard has $17.25  
Error: did not board?!  
   
Boarded at Uptown. SmartCard has $1.00  
Insufficient funds to exit. Please add more money.  
   
Insufficient funds to board. Please add more money.  
   
Boarded at Center City. SmartCard has $10.00  
From Center City to Downtown costs $0.50. SmartCard has $9.50  
   
Boarded at Suburb. SmartCard has $10.00  
From Suburb to Downtown costs $2.75. SmartCard has $7.25

After your program works, test all the different cases that you can think of. (How do you know if you have tested them all?) Test cases should try to make the program **fail**. There is a big difference between testing to show that something works versus testing to find out where something fails.