## 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 know, at a minimum, that the Station class has two private fields, a two-arg constructor, and two accessor methods. To ensure that you know the basics of writing classes, practice writing a default constructor, modifier methods, and a toString method (in your code but not necessarily here). Write the Station class on a separate sheet of paper.

Here are some example calls to the constructors of Station and SmartCard:

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

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

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

Station suburbia = new Station("Suburb", 4);

SmartCard jimmy = new SmartCard(20.00); //bought with $20.00

Let’s plan the SmartCard class. SmartCards keep track in which zone the passenger boarded, how much money is remaining, and whether the SmartCard was presented (true or false) to board or disembark (exit).

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 methods at a minimum (you are always welcome to write additional helper methods):

* addMoney (double d) – this method consumes how much money to add to this SmartCard and updates this SmartCard with the correct information.
* getBalance()– this method returns the balance on the SmartCard as a String formatted as dollars and cents.
* isBoarded()– this method returns a boolean value if the traveler is on board or not
* board (Station s)– this method consumes the Station where the traveler boards and updates the SmartCard with correct information. If the traveler tries to board without having previously disembarked, 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) – this method consumes the exiting station and calculates and returns the price the SmartCard holder would have to pay to disembark, as described above.
* disembark (Station s) – this method consumes the exiting station and updates the SmartCard, charging the customer the appropriate amount and setting the other fields correctly. It prints the trip’s two cities, the cost, and the card balance. If the traveler tries to disembark 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 prints the name of the two cities, the cost, and the balance on the SmartCard, as shown in the run below.

For this program, assume that the system will always work and that no one will lose a SmartCard between boarding and disembarking.

Consider a variety of test cases. Test cases should try to make the program **fail**. There is a big difference between testing to show that something works versus testing to show that something fails. Test the boundaries and test the extremes.

Here are sample runs for five different SmartCards:

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

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.

The shell is SmartCard\_Driver.java. Notice that Station, SmartCard, and the driver are all in this file.