SmartCard

Passengers on a certain subway system insert a SmartCard when entering a station and when exiting at their destination station. The SmartCard, after checking some things, subtracts the cost for traveling between stations. The stations are assigned to zones and the cost depends on the zone. Thus, each station knows its own name and zone.

Write the `Station` class. Include

- Two private fields
- A default constructor
- A two-arg constructor
- `getZone()`
- `getName()`

Know that the driver will instantiate four stations in four zones in this way:

```
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);
```

Write the `Station` class below.

Write the `SmartCard` class. Each `SmartCard` needs to store the money (the balance) and the Station where the passenger entered. A `SmartCard` also has a Boolean to record whether the passenger is on board the train (`true`) or not (`false`).

- How many fields does `SmartCard` need?
- Make a one-arg constructor to satisfy

  ```
  SmartCard buddy = new SmartCard(20.00);.
  ```

  Set the other fields to the standard default values: 0 for numbers, `null` for the Station object, and `false` for the boolean.
- Include these accessor methods:

  `getMoneyRemaining()` – returns the balance on the `SmartCard` as a double.

  `getBalance()` – returns the balance on the `SmartCard` as a `String` formatted as dollars and cents (Ex: $10.00). You will need to import java.text.DecimalFormat.

  `isBoarded()` – this method returns a `boolean` value as to whether the passenger is on board or not

  `getBoardedAt()` – returns the `Station` where the passenger entered. Note: it returns the Station object, not the name of the Station.

Now run the driver and test what we have so far. The output might be:

```
getMoneyRemaining() 20.0
getBalance() $20.00
isBoarded() false
getBoardedAt() null
```

When people enter a Station and put their `SmartCard` into the turnstyle, the `SmartCard` records the Station and the fact that they have entered. When people exit a Station by putting their card into the turnstyle, their `SmartCard` calculates and subtracts the cost of the trip.

Implement these actions with these methods:

`board(Station s)` – 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)`.

To test, have buddy board at Downtown and examine two fields on the SmartCard.

```
isBoarded() true
getBoardedAt() Station@5b480cf9
```

Have buddy board twice in a row: what should be printed?

Test what happens if buddy's SmartCard starts off with $0.25.

`cost(Station s)` – calculates and returns the cost to exit at this station, as follows:

- 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 1 to zone 1 costs $0.50.
- to travel from zone 1 to zone 2 costs $1.25
- to travel from zone 1 to zone 4 costs $2.75
- to travel from zone 4 to zone 1 costs $2.75

Test each situation, including (of course) the ones not listed.

`exit(Station s)` – if the traveler tries to exit without having previously boarded, it prints `"Error: Did not board?!"` and returns. If the cost of the travel 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 the balance and the boarding information, prints the names of the two cities, the cost, and the balance on the `SmartCard`. For example:

```
From Center City to Downtown costs $0.50.  SmartCard has $9.50
```

Test each situation.

Finally, let's allow the passenger to add some money to the SmartCard. Implement

```
addMoney(double d)
```

Test it.

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`. You will turn in `SmartCard` which also has the `Station` class.