Quiz

The Class TrafficSignal represents some kind of signaling device at road intersections.
The instances shown below were created by saying:

The Class TrafficSignal represents some kind of signaling device at road intersections.
The instances shown below were created by saying:

```
# a trafficsignal

s = TrafficSignal.new( 20, 20, 20 )

s.cycle()   # prints: 20 seconds of green
                      20 seconds of yellow
                      20 seconds of red
```

```
# a long green trafficsignal

longgreen = TrafficSignal.new( 50, 10, 20 )

longgreen.cycle() # prints: 50 seconds of
green
                            10 seconds of
yellow
                            20 seconds of
red
```

Each TrafficSignal has its own length of green, yellow and red period as shown in the class diagram below.

Class Diagram

TrafficSignal
TrafficSignal (int green, int yellow, int red) -- accessors int red( ) int yellow( ) int green( ) -- cycle will run the trafficsignal once void cycle( )
--class members int red int yellow int green

Based on the information shown here, create the class TrafficSignal.
Make sure its `cycle( )` method prints out a single cycle through the signal as shown above.

Based on this class TrafficSignal, please create the subclass LeftTurnSignal.
A LeftTurnSignal is a special kind of TrafficSignal that also includes a turn period.
The intended relationship between TrafficSignal and LeftTurnSignal is shown in the class diagram below.
Make sure its `cycle( )` method shows a green, yellow, red and turn period.

Class Diagram	

LeftTurnSignal
LeftTurnSignal(int green, int yellow, int red, int turn) -- accessors int red( ) int yellow( ) int green( ) int turn( ) -- cycle will run the leftturnsignal once void cycle( )
--class members int red # from the TrafficSignal class int yellow # from the TrafficSignal class int green # from the TrafficSignal class int turn

IN ORDER TO RECEIVE FULL CREDIT, YOUR LEFTTURNSIGNAL CLASS MUST PROPERLY REUSE THE TRAFFICSIGNAL CLASS, CALLING IT PARENT CLASS CONSTRUCTOR.

Once completed, please upload your source files into the Midterm File Upload Area.

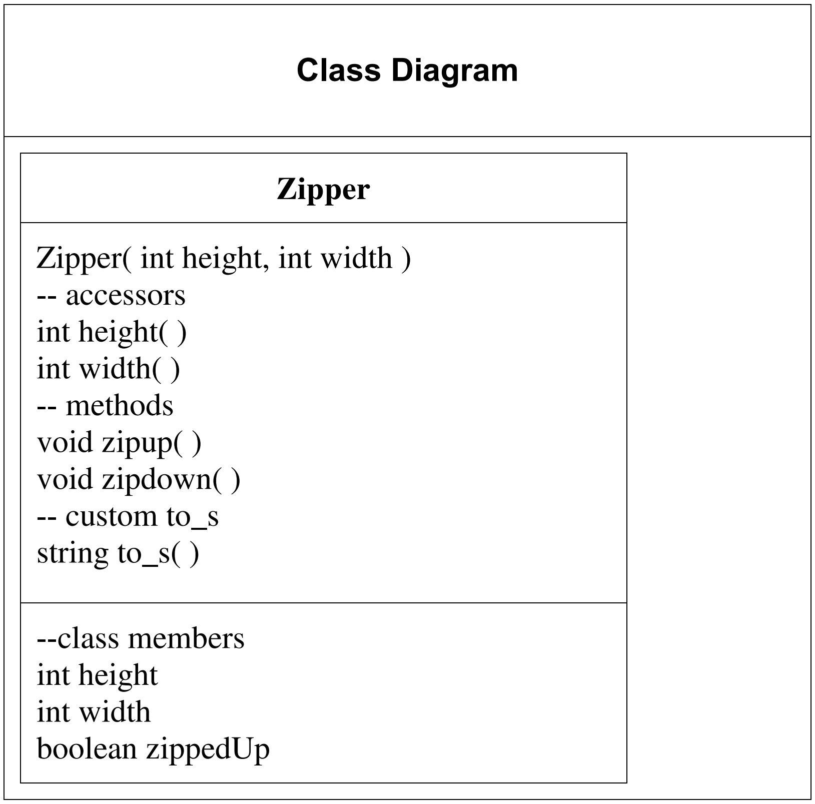
Please ignore the drop-down list box that appear here.

The Class Zipper represents some kind of clothing fastener.
The instances shown below were created by saying:

```
# a zipped-up zipper

z = Zipper.new( 4, 1 )
z.zipup( )
```

Each Zipper has its own height, width and closed field as shown in the class diagram below.



Class Diagram

Zipper

Zipper(int height, int width)
-- accessors
int height( )
int width( )
-- methods
void zipup( )
void zipdown( )
-- custom to_s
string to_s( )

--class members
int height
int width
boolean zippedUp

Based on the information shown here, a possible Ruby implementation is shown below.

```
class Zipper

   def initialize( height, width )
    @height = height
    @width = width
   end

   # accessor methods
   def height
    @height
   end

   def width
    @width
   end

   # other methods
   def zipup
    @zippedUp = true
   end

   def zipdown
    @zippedUp = false
   end

   def to_s
    "#{@height}-#{@width}-#{@zippedUp}"
   end

 end
```

Based on this class Zipper, please create the class Jacket. A Jacket has and uses a Zipper at various times.
The intended relationship between Jacket and Zipper is shown in the class diagram below.

Class Diagram	
Jacket	
Jacket(string color, string size, int zipperHeight, int zipperWidth) -- accessors string make( ) string model( ) -- methods void open( ) # unzip the zipper void close( ) # zipup the zipper	
--class members string color string size Zipper zipper	

IN ORDER TO RECEIVE FULL CREDIT, YOUR JACKET CLASS MUST PROPERLY REUSE THE ZIPPER CLASS.