```python
# Created by: Mr. Coxall
# Created on: Sep 2016
# Created for: ICS3U
# This class is used to define a bicycle object

class Bicycle:
    # this class defines a bicycle

    # class variable shared by all instances


    def __init__(self, gear, cadence = 0):
        # private fields

        self.__cadence = 0
        self.__speed = 0
        self.__gear = 0
        # if we are setting field, set them properly
        self.set_gear(gear)
        self.set_cadence(cadence)

        # public properties
        self.some_property = None

    # properties
    def get_cadence(self):
        # get the cadence property
        return self.__cadence

    def set_cadence(self, new_cadence):
        # set the cadence property
        if new_cadence < 0:
            #this is illegal, so do nothing
            pass
        else:
            self.__cadence_speed_recalculation(new_cadence)
            self.__cadence = new_cadence

    def get_speed(self):
        # get the speed property
        return self.__speed

    def get_gear(self):
        # get the gear property
        return self.__gear

    def set_gear(self, new_gear):
```

```
48          # set the gear property
49          if new_gear < 0 or new_gear > 10:
50              # do nothing, this is illegal
51              pass
52          else:
53              self.__gear_speed_recalculation(new_gear)
54              self.__gear = new_gear
55
56
57      # private methods
58      def __gear_speed_recalculation(self, new_gear):
59          # if you change the gear on a bike, the speed will change
60          old_gear = self.__gear
61
62          if old_gear > new_gear:
63              self.__speed = self.__speed - 5
64          elif old_gear < new_gear:
65              self.__speed = self.__speed + 5
66          else:
67              # same gear!
68              pass
69
70      def __cadence_speed_recalculation(self, new_cadence):
71          # if you change the cadence on a bike, the speed will
            change
72          old_cadence = self.__cadence
73
74          if old_cadence > new_cadence:
75              self.__speed = self.__speed + (1 + (new_cadence-
                old_cadence)/20)
76          elif old_cadence < new_cadence:
77              self.__speed = self.__speed + (1 + (new_cadence-
                old_cadence)/20)
78          else:
79              # same cadence!
80              pass
81
82      # public methods
83
84      def apply_brakes(self, speed_decrease):
85          # decrease the current speed by value passed in
86
87          self.__speed = self.__speed - speed_decrease
88          if self.__speed < 0:
89              self.__speed = 0
90
91      def current_state(self):
```

```
92          # returns the current state of the bicycle as a string
93
94          # this varaible is local to this method
95          return_string = 'Cadence: ' + str(self.__cadence) + '
            Speed: ' + str(self.__speed) + ' Gear: ' + str(self.__gear)
96
97          return return_string
98
```

```
New bike1
Cadence: 0 Speed: 5 Gear: 2 # of seats: 2
Current speed: 8
Set gear to 7
Current gear: 7
Current speed: 13
Set cadence to 60
Current cadence: 60
Current speed: 15
Apply breaks by 3
Cadence: 60 Speed: 12 Gear: 7 # of seats: 2

New bike2
Cadence: 20 Speed: 7 Gear: 3
The starting cadence is: 20
The starting speed is: 7
Set cadence to 90
Current speed: 11
Set gear to 3
Current gear: 3
Current speed: 11
Set cadence to 45
Current cadence: 45
Current speed: 9
Apply break 12
Cadence: 45 Speed: 0 Gear: 3
```