```
#
# Student Class Definition
#
# 4 Instance Variables:
# _stuID str - 5 characters, 2 alpha (initials), 3 numeric (random)
# _name list - 2 str elements, first name, last name
# _testScores list - int values (could be empty, no tests taken)
# _avg float - the average of _testScores (0.0 if no tests taken)
#
# Constructor:
# __init__(self, stuID) Initialize _stuId to stuID, _name to a list with
# 2 empty str elements, _testScores to the empty list,
# _avg to 0.0
#
# 6 Public Instance Methods:
# getID(self) return _stuID
# getName(self) return _name
# getTestScores(self) return _testScores
# getAvg(self) return _avg
# setName(self, firstName, lastName) Change the 2 elements in _name to
# firstName and lastName.
# addTest(self, testScore) Append 1 testScore onto _testScores,
# then call _calcAvg() to set _avg to
# the updated value.
#
# 1 Private Instance Method:
# _calcAvg(self) called by addTest() to keep _avg accurate every time
# a new test score is added, returns a float value that
# is the average of the test scores, 0.0 if no test scores
#

class Student:

    def __init__(self, stuID):
        self._stuID = stuID
        self._name = ("", "")
        self._testScores = []
        self._avg = 0.0

    def getID(self):
        return self._stuID
    def getName(self):
        return self._name
    def getTestScores(self):
        return self._testScores
    def getAvg(self):
        return self._avg

    def setName(self, firstName, lastName):
        newName = (firstName, lastName)
```

```
    self._name = newName
def addTest(self, testScore):
    self._testScores.append(testScore)

    self._calcAvg()

def _calcAvg(self):
    numScores = len(self._testScores)
    average = 0.0
    for score in self._testScores:
        average += score / numScores
    self._avg = average
```