```
79          else:
80              return None
81
82
83  class ClassData():
84      def __init__(self, file):
85          points = readFileOfPoints(file)
86          self.points = points
87          self.len = len(points)
88
89      @property
90      def toNdarray(self):
91          return nu.array([ point.rawValue for point in self.points ])
92
93      @property
94      def mean(self):
95          return self.toNdarray.sum(axis=0).reshape(2,1) / self.len
96
97
98  class TestData(ClassData):
99      def __init__(self, file):
100         super().__init__(file)
101         correctClasses = readFileOfCorrectClasses(file)
102         self.correctClasses = correctClasses
103
104     def nearestNeighborMethod(self, class1, class2):
105         nearestNeighborArray = []
106         for point in self.points:
107             leastDistanceInClass1 = nu.array([ point.distanceFrom(class1Point.ndarrayV
108             leastDistanceInClass2 = nu.array([ point.distanceFrom(class2Point.ndarrayV
109             nearestNeighborArray.append( 1 if leastDistanceInClass1 < leastDistanceInC
110         return nearestNeighborArray
111
112     def euclideanDistanceMethod(self, class1, class2):
113         return [ 1 if point.distanceFrom(class1.mean) < point.distanceFrom(class2.mean
114
115     def weightDistanceMethod(self, class1, class2, weight):
116         return [ 1 if point.distanceFrom(class1.mean, weight) < point.distanceFrom(cla
117
118     def similarityMethod(self, class1, class2):
119         return [ 1 if point.similarityToPoint(class1.mean) < point.similarityToPoint(c
120
```