```java
// Lab 10: AssociationList
// I kept getting lost between methods while writing this because
they all are kind of the same.
// Andrea Smith
// CSCI 1913

class AssociationList<Key, Value>
{
  private class Node
  {
    Key key;
    Value value;
    Node next;
// Constructor shtuff. Does anyone read these?
    private Node(Key key, Value value, Node next)
    {
      this.key = key;
      this.value = value;
      this.next = next;
    }
  }

  private Node head = null;

// Create the head node
  public AssociationList()
  {
    head = new Node(null, null, null); // initalize head
  }

// DELETE: Deletes a node in AssociationList using the "left right"
method from lecture
  public void delete(Key key)
  {
    Node left = head; // Node being visited
    Node right = left.next;

    while(right != null)
    {
      if(isEqual(right.key, key))
      {
        left.next = right.next;
        break;
      }
      else
      {
        left = right;
```

```
46          right = right.next;
47        }
48      }
49    }
50
51    // GET: Returns the value of the key if the key is found in the
      given Node
52    public Value get(Key key)
53    {
54      Node right = head.next;
55
56      while(right != null)
57      {
58        if(isEqual(right.key, key))
59        {
60          return right.value;
61        }
62        else
63        {
64          right = right.next;
65        }
66      }
67      throw new IllegalArgumentException(); // Will only get here if
        while loop requirement isn't fulfilled.
68    }
69
70  // ISEQUAL: Checks if keys are equal
71    private boolean isEqual(Key leftKey, Key rightKey)
72    {
73      if (leftKey == null || rightKey == null)
74      {
75        return (leftKey == rightKey);
76      }
77      else
78      {
79      return (leftKey.equals(rightKey));
80      }
81    }
82
83  // ISIN: Checks if key is in the given node
84    public boolean isIn(Key key)
85    {
86      Node right = head.next;
87
88      while (right != null)
89      {
90        if (isEqual(right.key, key))
```

```
90        if (isEqual(right.key, key))
91        {
92          return true;
93        }
94        else
95        {
96          right = right.next;
97        }
98      }
99      return false; // Will only get here if while loop requirement
        isn't fulfilled.
100   }
101
102 // PUT: Adds a new node if the key isn't in the given Node
103   public void put(Key key, Value value)
104   {
105
106     Node right = head.next;
107
108     while (right != null)
109     {
110       if (isEqual(right.key, key))
111       {
112         right.value = value;
113         break;
114       }
115       else
116       {
117         right = right.next;
118       }
119     }
120     // If no such node exists, add a new node immediately after the
        head node
121     Node newNode = head.next;
122     head.next = new Node(key, value, newNode);
123   }
124
125 }
126
127 //
128 //  Tests for CSci 1913 Lab 10
129 //  James Moen
130 //  08 Apr 19
131 //
132 //  The TRY-CATCH statements catch exceptions thrown by ASSOCIATION
    LIST's
133 //  methods, so that the program can continue to run even if a
```

```
  •  method fails.
134  //
135  //  Each test has a comment that shows what it should print and how
  •  many points
136  //  it is worth, for a total of 40 points.
137
138  //  HOGWARTS. The Hogwarts dating service.
139
140  class Hogwarts
141  {
142
143  //  MAIN. Make an instance of ASSOCIATION LIST and test it.
144
145    public static void main(String[] args)
146    {
147      AssociationList<String,String> list = new
  •      AssociationList<String,String>();
148
149      System.out.println(list.isIn(null));         //  false         2
  •      points.
150
151      try
152      {
153        System.out.println(list.get(null));
154      }
155      catch (IllegalArgumentException ignore)
156      {
157        System.out.println("No null");             //  No null       2
  •        points.
158      }
159
160      list.put(null,        "Wormtail");
161      list.put("Ron",       "Lavender");
162      list.put("Voldemort", null);
163      list.put("Dean",      "Ginny");
164
165      System.out.println(list.isIn("Dean"));       //  true          2
  •      points.
166      System.out.println(list.isIn("Ginny"));      //  false         2
  •      points.
167      System.out.println(list.isIn("Ron"));        //  true          2
  •      points.
168      System.out.println(list.isIn("Voldemort")); //  true          2
  •      points.
169      System.out.println(list.isIn(null));         //  true          2
  •      points.
170      System.out.println(list.isIn("Joanne"));     //  false         2
```

```
    points.

    System.out.println(list.get("Ron"));         //  Lavender      2
    points.
    System.out.println(list.get("Dean"));        //  Ginny         2
    points.
    System.out.println(list.get("Voldemort"));   //  null          2
    points.
    System.out.println(list.get(null));          //  Wormtail      2
    points.

    try
    {
      System.out.println(list.get("Joanne"));
    }
    catch (IllegalArgumentException ignore)
    {
      System.out.println("No Joanne");           //  No Joanne     2
      points.
    }

    list.delete(null);

    System.out.println(list.isIn(null));         //  false         2
    points.

    list.put(null,    null);
    list.put("Harry", "Ginny");
    list.put("Ron",   "Hermione");

    System.out.println(list.isIn(null));         //  true          2
    points.
    System.out.println(list.get(null));          //  null          2
    points.
    System.out.println(list.get("Harry"));       //  Ginny         2
    points.
    System.out.println(list.get("Dean"));        //  Ginny         2
    points.
    System.out.println(list.get("Ron"));         //  Hermione      2
    points.

    list.delete("Dean");

    try
    {
      System.out.println(list.get("Dean"));
    }
```

```
205       }
206       catch (IllegalArgumentException ignore)
207       {
208         System.out.println("No Dean");             //  No Dean       2
  •         points.
209       }
210     }
211   }
212
```