Binary search, plus more programming with methods, objects, and arrays

1. Binary search
   a. Write a method that creates an array of 1000 random integers between 0 and 99, and returns the array.
   
   b. Write a method that creates an array of \( \text{num} \) random integers between 0 and \( \text{max} \), and returns the array.
   
   c. Write a method that takes an integer array as a parameter. The method should ask the user for 5 integers between 0 and 99, and print out the position of each one in the array (-1 if it can’t find it). Use a linear search method.
   
   d. Now modify the method so that it uses binary search instead.
   
   e. What needs to happen before you can run binary search? What do you need to import in order for this to happen?
   
   f. Which method will be faster, the linear search method or the binary search method? Why?
   
   g. Look at the example array below. For each key, indicate the positions in the array (the indexes, not the values) that a binary search would visit if it was searching for that key.

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>16</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>5</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>-25</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>101</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>45</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>-9</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Key: 4 Positions visited during binary search:

Key: 16 Positions visited during binary search:

Key: -25 Positions visited during binary search:

Key: 101 Positions visited during binary search:

Key: 45 Positions visited during binary search:

Key: -9 Positions visited during binary search:
2. **More programming with methods**
   
a. Write a method called `isEven` that returns whether an integer is even or not.

b. **Write a method called `isPerfect` that returns whether an integer is a perfect number or not.** A perfect number is one that is equal to the sum of its factors (not including the number itself as a factor). For instance, 6 is a perfect number because its factors are 1, 2, and 3, and $1+2+3=6$. 28 is the next perfect number, because $1+2+4+7+14=28$.

c. Write a loop that finds the next perfect number after 28. Use the `isPerfect` method from above (you don’t have to solve (b) to do this one, just pretend that you have a correct version of `isPerfect()` defined).

d. Write a method that returns the average value of a 2D array of doubles.

e. Write a method that returns the “average” Point from an array of Points. The “average” Point has an x coordinate that is the average of all the x coordinates in the array, and a y coordinate that is the average of all the y coordinates in the array.

f. Write a method that returns the Point object in an array that is farthest from `(new Point(0,0))` -- the origin.

g. Write a class that calls each of the methods you defined above.