The analysis and design of a class consists of a specification of its attributes and its operations. The specification of the attributes of a class is a table of the attribute objects, not unlike the analysis portion of a main program or function. It includes the description, type, and name of the attribute object, but does not need to include the kind or movement of the object.

The specification of the operations of a class is a list of function analyses and designs for each member, friend, or overloaded function of the class. The operations should be listed in the following order: constructors, destructors, accessors, conversion functions, mutators, I/O functions, and overloaded operators. Friend functions should be listed after the member functions. Finally, any free overloaded functions should be listed last. Since the attribute objects are accessible by all member and friend functions, they are not listed again in the analyses of these functions, unless being returned, but are to be used directly in the designs. For example, the analysis and design for the Temperature class from the textbook would be written as follows:

**Specification for Temperature class**

**Attributes - data stored in the object**

<table>
<thead>
<tr>
<th>Objects</th>
<th>Type</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of degrees</td>
<td>double</td>
<td>myDegrees</td>
</tr>
<tr>
<td>Temperature scale</td>
<td>char</td>
<td>myScale</td>
</tr>
</tbody>
</table>

**Operations - analyses and designs of the class functions**

(This is not a complete set of operations, but it gives an idea of format)

- Default constructor
  - Analysis - no objects
  - Design - what are the steps to solve this problem?
    1. Initialize myDegrees to 0.0
    2. Initialize myScale to 'C'

- Explicit-value constructor
  - Analysis - what data is received, passed back, returned, or local?
Design - what are the steps to solve this problem?

1. If islower (initialScale) then
   1.1 Convert to uppercase, initialScale = toupper (initialScale)
2. Select on initialScale
   2.1 Case of 'F', 'C', 'K'
      2.1.1 Initialize myDegrees to initialDegrees
      2.1.2 Initialize myScale to initialScale
   2.2 Default case
      2.2.1 Print error message
      2.2.2 Terminate program using exit(1)

- GetDegrees- returns the number of degrees of this temperature
Analysis - what data is received, passed back, returned, or local?

<table>
<thead>
<tr>
<th>Objects</th>
<th>Type</th>
<th>Kind</th>
<th>Movement</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of degrees</td>
<td>double</td>
<td>variable</td>
<td>returned</td>
<td>myDegrees</td>
</tr>
</tbody>
</table>

Design - what are the steps to solve this problem?

1. Return myDegrees

- GetScale- returns the scale of this temperature
Analysis - what data is received, passed back, returned, or local?

<table>
<thead>
<tr>
<th>Objects</th>
<th>Type</th>
<th>Kind</th>
<th>Movement</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Scale of temperature</td>
<td>char</td>
<td>variable</td>
<td>returned</td>
<td>myScale</td>
</tr>
</tbody>
</table>

Design - what are the steps to solve this problem?

1. Return myScale
- Fahrenheit - return the Fahrenheit equivalent of this temperature
  Analysis - what data is received, passed back, returned, or local?

<table>
<thead>
<tr>
<th>Objects</th>
<th>Type</th>
<th>Kind</th>
<th>Movement</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fahrenheit equivalent</td>
<td>Temperature</td>
<td>variable</td>
<td>returned</td>
<td>-----</td>
</tr>
</tbody>
</table>

Design - what are the steps to solve this problem?

1. Select on myScale
   1.1 Case 'F' (already in Fahrenheit)
      1.1.1 Return Temperature (myDegrees, 'F')
   1.2 Case 'C'
      1.2.1 Return Temperature (myDegrees * 1.8 + 32.0, 'F')
   1.3 Case 'K'
      1.3.1 Return Temperature ((myDegrees - 273.15) * 1.8 + 32.0, 'F')

- friend operator== - return true if two temperatures are equal, false otherwise
  Analysis - what data is received, passed back, returned, or local?

<table>
<thead>
<tr>
<th>Objects</th>
<th>Type</th>
<th>Kind</th>
<th>Movement</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Left operand</td>
<td>Temperature</td>
<td>variable</td>
<td>received</td>
<td>leftOperand</td>
</tr>
<tr>
<td>Right operand</td>
<td>Temperature</td>
<td>variable</td>
<td>received</td>
<td>rightOperand</td>
</tr>
<tr>
<td>Result of compare</td>
<td>bool</td>
<td>variable</td>
<td>returned</td>
<td>-----</td>
</tr>
<tr>
<td>Same scale temperature</td>
<td>Temperature</td>
<td>variable</td>
<td>local</td>
<td>localTemp</td>
</tr>
</tbody>
</table>

Design - what are the steps to solve this problem?

1. Select on leftOperand.myScale (find rightOperand equivalent in same scale)
   1.1 Case 'C'
      1.1.1 localTemp = rightOperand.Celsius()
   1.2 Case 'F'
      1.2.1 localTemp = rightOperand.Fahrenheit()
   1.3 Case 'K'
      1.3.1 localTemp = rightOperand.Kelvin()
2. Return leftOperand.myDegrees == localTemp.myDegrees

- friend operator+ - return new temperature with added degrees
  Analysis - what data is received, passed back, returned, or local?
### Design - what are the steps to solve this problem?

1. Return Temperature (leftOperand.myDegrees + rightOperand, leftOperand.myScale)

- friend operator<< - Output degrees and scale to an output stream

### Analysis - what data is received, passed back, returned, or local?

### Objects

<table>
<thead>
<tr>
<th>Objects</th>
<th>Type</th>
<th>Kind</th>
<th>Movement</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Left operand</td>
<td>Temperature</td>
<td>variable</td>
<td>received</td>
<td>leftOperand</td>
</tr>
<tr>
<td>Right operand</td>
<td>double</td>
<td>variable</td>
<td>received</td>
<td>rightOperand</td>
</tr>
<tr>
<td>Temperature w/added degrees</td>
<td>Temperature</td>
<td>variable</td>
<td>returned</td>
<td></td>
</tr>
</tbody>
</table>

### Objects

<table>
<thead>
<tr>
<th>Objects</th>
<th>Type</th>
<th>Kind</th>
<th>Movement</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Output stream to write to</td>
<td>ostream</td>
<td>variable</td>
<td>received, passed back, &amp; returned</td>
<td>outStream</td>
</tr>
<tr>
<td>Temperature to output</td>
<td>Temperature</td>
<td>variable</td>
<td>received</td>
<td>theTemp</td>
</tr>
</tbody>
</table>

### Design - what are the steps to solve this problem?

1. Write myDegrees and myScale to outStream
2. Return outStream

- friend operator>> - read a temperature from an input stream
1 Read in inDegrees and inScale from inStream
2 If islower (inScale) then
   2.1 Convert to uppercase, inScale = toupper (inScale)
3 Select on initialScale
   3.1 Case of 'F', 'C', 'K'
      3.1.1 Set theTemp.myDegrees to inDegrees
      3.1.2 Set theTemp.myScale to inScale
   3.2 Default case
      3.2.1 Set input stream fail bit using inStream.set_state(ios::fail)