Hash Table

1. Map interface:

```java
public interface Map<Key, Value> {
    public void set(Key k, Value v);
    public boolean contains(Key k);
    public Value get(Key k);
    public int size();
}
```

2. Example applications

3. Implementations

4. Recall bucket sort
   a. Example: Sort the following numbers: 57 25 38 15 29 42
      Using their first digit to predict location into the following buckets:

      ![Bucket diagram]

      b. Underlying array of lists
      c. Function
      d. Want $O(1)$ elements per bucket and $O(n)$ total buckets

5. Hash Table
   a. Example: Map from first name to last name
      i. Strings hashed by second letter, 5 buckets
      ii. Elements: Joey Kiernan, Elisa Chang, Jack Wadden, Emily Yu, Jesse Levitt
      iii. Build the data structure

### Decimal ASCII Codes

<table>
<thead>
<tr>
<th>Char</th>
<th>Code</th>
<th>Code % 5</th>
<th>Char</th>
<th>Code</th>
<th>Code % 5</th>
<th>Char</th>
<th>Code</th>
<th>Code % 5</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>97</td>
<td>2</td>
<td>j</td>
<td>106</td>
<td>1</td>
<td>s</td>
<td>115</td>
<td>0</td>
</tr>
<tr>
<td>b</td>
<td>98</td>
<td>3</td>
<td>k</td>
<td>107</td>
<td>2</td>
<td>t</td>
<td>116</td>
<td>1</td>
</tr>
<tr>
<td>c</td>
<td>99</td>
<td>4</td>
<td>l</td>
<td>108</td>
<td>3</td>
<td>u</td>
<td>117</td>
<td>2</td>
</tr>
<tr>
<td>d</td>
<td>100</td>
<td>0</td>
<td>m</td>
<td>109</td>
<td>4</td>
<td>v</td>
<td>118</td>
<td>3</td>
</tr>
<tr>
<td>e</td>
<td>101</td>
<td>1</td>
<td>n</td>
<td>110</td>
<td>0</td>
<td>w</td>
<td>119</td>
<td>4</td>
</tr>
<tr>
<td>f</td>
<td>102</td>
<td>2</td>
<td>o</td>
<td>111</td>
<td>1</td>
<td>x</td>
<td>120</td>
<td>0</td>
</tr>
<tr>
<td>g</td>
<td>103</td>
<td>3</td>
<td>p</td>
<td>112</td>
<td>2</td>
<td>y</td>
<td>121</td>
<td>1</td>
</tr>
<tr>
<td>h</td>
<td>104</td>
<td>4</td>
<td>q</td>
<td>113</td>
<td>3</td>
<td>z</td>
<td>122</td>
<td>2</td>
</tr>
<tr>
<td>i</td>
<td>105</td>
<td>0</td>
<td>r</td>
<td>114</td>
<td>4</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Next time: choosing a good hash function, and the open addressing scheme discussed in the textbook