1. (20 points) Provide short answers (a few sentences) to each of the following questions:

(a) What is an abstract class?
(b) Why should instance variables never be declared public?
(c) Consider some method foo() that calls another method bar(). Assume that bar() may throw a ThisOrThatException, which is a checked\textsuperscript{1} exception type. How must foo() be written to address this characteristic of bar()?

2. (20 points) Consider the following doubly-recursive method:

```java
public static void toZero (int n) {
    if (n == 0) {
        return;
    }
    System.out.println(n);
    if (n > 0) {
        toZero(n - 1);
        toZero(-(n - 1));
    } else if (n < 0) {
        toZero(n + 1);
        toZero(-(n + 1));
    }
}
```

Write the output that this method would generate if it were called with an argument of 4.

\textsuperscript{1}That is, not a subclass of RuntimeException.
3. (30 points) Consider *Pascal’s Triangle*, shown here in a usefully lopsided form:

<table>
<thead>
<tr>
<th>\col</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td>\row</td>
<td>0</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>2</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>3</td>
<td>3</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>4</td>
<td>6</td>
<td>4</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>5</td>
<td>10</td>
<td>10</td>
<td>5</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>6</td>
<td>15</td>
<td>20</td>
<td>15</td>
<td>6</td>
<td>1</td>
</tr>
</tbody>
</table>

Specifically, we define the value of any position in the triangle at row $r$ and column $c$ as:

$$T(r, c) = \begin{cases} 
1 & \text{if } c = 0 \text{ or } c = r \\
T(r - 1, c - 1) + T(r - 1, c) & \text{if } 0 < c < r 
\end{cases}$$

That is, the values at the edges (the left-column and right-most-diagonal) are always 1, while the interior values are the sum of the values above-and-to-the-left and immediately-above. Further consider the following method header:

```java
public static int[][] pascal (int r)
```

**Write this method** such that it creates a two-dimensional matrix of integers that are the first $r$ rows of Pascal’s Triangle. Note that each row is one element longer than the previous one, and so too should the second dimension arrays of this matrix be.
4. (30 points) Consider the following partially-written class:

```java
public class IntArray {

    private int[] _storage;

    public IntArray (int size) {
        _storage = new int[size];
    }

    public int length () {
        return _storage.length;
    }

}
```

Notice that this is a container class for int values. An IntArray must be created to have a specific length (just like regular arrays). Moreover, indexing into an IntArray is a bit different from normal arrays. Specifically, each entry may be accessed via either of two indices:

(a) Its forward index, which is that entry’s position counting forwards from the beginning of the array, represented as a non-negative integer; or

(b) Its backward index, which is that entry’s position counting backwards from the end of the array, represented as a negative integer.

In other words, for an array of length 4, the forward indices are 0, 1, 2, and 3; the backward indices to those same 4 entries are respectively -4, -3, -2, and -1. Complete this class by adding the following methods:

- A copy constructor.
- A deep equality comparison method.
- A setter that sets the entry at index i to the value v. Remember that i may be either a forward or backward index. If i is invalid—too large in magnitude for length—then this method should throw an ArrayIndexOutOfBoundsException that contains the offending i.
- A getter that returns the value at index i. As with the setter, forward and backward indices are allowed, and invalid indices should trigger the appropriate exception.