Objectives:
Get practice with writing classes and methods.

Overview:
The students in the class will develop a program to display block letters which can be put together to form messages. Each student will be assigned one or more letters of the alphabet.

Step 1
Create class called BigX, where X is a letter of the alphabet (for example, BigA) that represents a block letter. This class should have:
1. an instance variable letter of type char to hold the letter to be used for the display.
2. a constructor with no parameters that sets letter to the letter of the alphabet for this particular class.
3. the toString() method, which should produce a string in the shape of your letter.

Save BigX.java in the shared folder as instructed in class.

Step 2
We will test everyone’s letters using the driver class:

```java
public class bigLetters {
    public static void main (String args[]) {
        BigA A = new BigA();
        BigA B = new BigB();
        BigA C = new BigC();
        // and so on

        System.out.println ("" + A + B + C );
    }
}
```
Step 3

Once you get all this working, we will enhance the class by making it possible to use a different character to construct the block letter. To do this, create a second constructor that has a parameter of type char and sets the instance variable letter to the value of the parameter.

In the driver class the constructor will be tested using code like this:

```java
BigA A = new BigA('*');
System.out.println ("" + A);
```

which should produce output that looks like this:

```
  **  
 ****  
 ** **  
 ** **  
*********  
 ** **  
 ** **  
```

In order for this to happen, you will need to modify your `toString()` method to use the character stored in the instance variable, instead of always using the same letter (for example, A). Replacing all the occurrences of the standard letter with the instance variable can be so tedious though! You would need code like this:

```java
message += '
' + letter + letter + letter + letter + letter;
```

... so how about creating a method that can be used when you want to repeat the instance variable, for example, like this:

```java
message += '      ' + symbols(5);
```

This is an example of a utility method – it would not be intended to be used by other classes, only within the class itself. What kind of visibility modifier should this method have? What should its return type be?

Once you have the `symbols()` method working, you may notice that it is possible to generalize it to make it repeat any character, not just the instance variable – so you could have a second version of this method, with two parameters, so that the above code could be rewritten like this:

```java
message += symbols(6, ' ') + symbols(5);
```
Now, using your `symbols()` method, you can rewrite the `toString()` method by replacing, for example, the occurrence of:

```
    AA   AA
```

by:

```
symbols(8, ' ') + symbols(2) + symbols(4, ' ') + symbols(2)
```

(you can just read off your previous design).

**Step 4**

Now we will use the block letters in a GUI. Use `JTextArea` (it is like `JLabel` except it can contain multiple lines) and we will change the font so that the characters line up properly. For example, if you have a `JPanel` called `primary`, you can add a block letter to it like this:

```java
Font font = new Font("Courier New", Font.BOLD,12);
JTextArea A = new JTextArea( new BigA().toString());
A.setFont(font);
primary.add (A);
```

Repeat this code to add multiple letters. At this point, assuming your classmates have been doing their work, you can start adding their letters to your panel. Try spelling your name or another word. Enjoy!