Overview
For part one of the assignment, you'll be creating your own critter. For part two, you'll be using recursion to reverse arrays.

Setup
In all of the following, the > is a generic command line prompt (you do not type that).
You will need to create a new directory named pa6 in your cs8b home directory on ieng6.ucsd.edu and copy over files from the public directory.

```
> cd
> mkdir pa6
> cp ~/../public/pa6/* pa6/
> cp ~/pa5/* pa6/
```

Your pa6 directory should now contain the following files:

```
// code files
Critter.java               // do not change
CritterMain.java           // do not change - this class manages gameplay
Bear.java                   // you created in PA5
Lion.java                   // you created in PA5
Tiger.java                  // you created in PA5
Easy.class                // Easy Bot
Medium.class               // Medium Bot
Hard.class                  // Hard Bot
PA6Strings.java            // For ReverseRecurse formatting - do not change
TestReverseRecurse.java   // For running your Reverse Recurse - do not change
```

README (10 points)
You are required to provide a text file named README, NOT Readme.txt, README.pdf, or README.doc, with your assignment in your pa6 directory. There should be no file extension after the file name “README”. Your README should include the following sections:

Program Descriptions (6 points): Provide a high level description of what each of your programs do and how you can interact with them. Make this explanation such that your grandmother or uncle or someone you know who has no programming experience can understand what these programs do and how to use them. Do not assume your reader is a computer science major. The more detailed the explanation, the more points you will receive.

Short Response (4 points): Answer the following questions:

Vim Questions:
- In vim/gvim, what commands will indent N consecutive lines (starting from the cursor line) by one level where the indent level is defined to be two spaces? This will take two vim commands: one to set the number of spaces to indent with each indent level (default is 8), and one to actually indent N consecutive lines. Likewise what command will shift N lines left (de-indent N lines)?
In vim/gvim, what command will indent an entire curly-bracket block one level, while the cursor is currently on either the open or close curly bracket of the block? Likewise what command will shift an entire curly-bracket block one level left (de-indent block)?

How do you open a new line below and insert (one keyboard key for both steps)?

**STYLE (20 points)**
Please see style requirements listed in PA5.
You will be specifically graded on commenting, file headers, class and method headers, meaningful variable names, sufficient use of blank lines, not using more than 80 characters on a line, perfect indentation, no magic numbers/hard-coded numbers other than zero.

**CORRECTNESS (70 points total for the 2 programs)**

**Program 1: Critters Part 2 (35 Points)**

For this assignment you will be implementing your very own Critter! You need to name your critter file **MyCritter.java**. Don’t forget that it must extend the Critter class. Just like in part 1 you will override the 5 main methods. You may also want to override the other (non-final) methods from the critter class so that you can update internal state based on what happens to your critter. These other methods (win(), lose(), sleep() etc.) are all called on your critter when those events occur. For example the sleep() method is called on your critter once the Game puts your critter to sleep and wakeup() once the Game wakes your critter up. Note, these methods don’t control if your critter is asleep but they simply notify your critter of the event.

Your objective is to create the Ultimate critter! One who can outlive and beat the other critters in the world. Its Survival of the fittest! Will your critter survive?

Your critter will have to be able to win in 3 environments:

1. **(10 points)** A world of Lions, Tigers, and Bears (LTBs), Oh my!
2. **(10 points)** A world with an Easy Bot and LTBs
3. **(15 points)** A world with a Medium Bot and LTBs

If your critter is up to the challenge then you can have a go at the Hard Bot (and LTBs). If you are able to win in this situation (with the Hard Bot and LTBs) then you will be awarded Extra Credit!

To determine if your critter has successfully beat an environment, it will have the highest score after 1000 moves have occurred. Your critter will have to successfully beat an environment 2 out of 3 times in order to be awarded full points. (Yes - some outcomes are random, if you wish you be sure of a victory, be sure you win a significant fraction of the time.)

Good luck! And may the odds be ever in your favor!

**Program 2: Reverse-Recurse (35 Points)**

**ReverseRecurse.java**: Write an application that reads integer values from the keyboard into an array whose size is specified by the user (the user may enter fewer integers than the size specified, but not more) and reverse the elements in the array via two different recursive methods. One method directly modifies the original array. The other method returns a new array with the elements reversed preserving the original array. Both will use an "ends-and-middle" or "edges-and-center" recursion.
You will be given the main test driver that we will use to call/test your solution.
~/.ctest/public/pa6/TestReverseRecurse.java

You will write

```java
public class ReverseRecurse { ... }
public int[] initArray() { ... }
public void printArray( int[] array ) { ... }
```

```java
/*
 * The following two reverse() methods must be implemented
 * using recursion.
 */

public void reverse( int[] originalArray, int low, int high ) { ... }
public int[] reverse( int[] originalArray ) { ... }
```

`initArray()` will ask the user for a maximum number of integers expected, create an array of integers that size, read at most that many integers from the keyboard using a Scanner object (ignore extra input beyond the size of the array), and return the initialized array. The user must enter a positive integer greater than 0 for the size of the array. Keep asking the user for a positive integer greater than 0 until a valid value is entered.

If the user indicates EOF (see next sentence) or a non-integer input while the size is not determined yet, the program should exit with exit status 1 [System.exit(1);]. If the user hits <Ctrl>+D (Unix/Mac) or <Ctrl>+Z (DOS/Windows) to indicate EOF (no more input) or any non-integer before the entire array has been filled (but after the size has been set), return an array resized with only the values that were entered (no extra/unfilled array slots). Hint: Use System.arraycopy().

To read ints from standard input (System.in which is the keyboard by default), use a Scanner object (see the Javadocs for class Scanner).

```java
Scanner in = new Scanner( System.in );
```

You can then use method hasNextInt() to determine if the next token on the input is a valid int and method nextInt() to scan the next token as an int.

```java
if ( in.hasNextInt() ) {
    size = in.nextInt();
    ...
}
```

`printArray()` cycles through the passed array printing each integer on the same line separated by a space. Output a newline after all the elements are printed. Use either a for loop or the enhanced-for loop to iterate over the elements in the array. This method will print "Empty array" if there are no elements in the array (an empty array).

The first overloaded recursive reverse() will directly manipulate the array passed in by exchanging the low and high index values and recurse on the remaining middle/center elements of the array by passing modified values of low and high in the recursive call. Set up the Base Case(s) and Recursive Case(s).
The second overloaded recursive reverse() will not change anything in the original/passed array. Instead it will copy the first element of the original (passed) array into the last slot of a new array we are building to hold the reverse of the passed array and copy the last element of the original (passed) array into the first slot of the new reversed array. Then copy (think System.arraycopy()) the middle elements of the passed array into another new array (you need to dynamically create this new array based on the size/number of the middle elements) and recursively reverse these middle elements in this new array. (Leap of Faith.) reverse() returns an array with the passed array elements reversed (it does not modify the passed array). After the recursive call to reverse the middle elements, copy the reversed middle elements that was returned by the recursive call to reverse() into the middle section of the new reversed array being built and return this new array that now has all the elements of the passed array reversed. Set up the Base Case(s) and Recursive Case(s).

Make sure that the last two tests in the tester pass with “SUCCESS: No NullPointerException thrown.” message by dealing with null array input in your two reverse() methods. The key point here is that you do not manipulate the null object passed in as array in your reverse methods, but immediately return. If you do not deal with this special corner case, the last two tests will print “FAIL: NullPointerException – Fix me!”.

Please note that your ReverseRecurse will be ran through a script. We stress that you should match the output EXACTLY. In order to make your lives easier, we have included a PA6Strings.java for your convenience. In order to use these strings, you need to call these strings in this manner:

```
System.out.println(PA6Strings.EMPTY);
```

In order to make this work, we made the strings in PA6Strings.java static, so they can be accessed through the class name. Because of this, A PA6Strings.java object DOES NOT need to be instantiated. Also, because one of the parts of the string needs to be dynamically set, we are going to introduce a new print method: System.out.printf(); This is the same as the other print statements, except it can be used to insert values into a string. Let’s take the following example:

```
ENTER_INTS = "\nEnter up to %d integers:"
```

In this case, %d signifies a decimal placeholder. It needs a decimal value as input in order to place the string. It can be used in the following manner:

```
int decimal = 10;
System.out.printf(PA6Strings.ENTER_INTS, decimal);
```

This will print “Enter up to 10 integers:”. Now apply this thinking to make the number of integers that should be read dynamic.

Here are some example test runs: (the following outputs assume the last two tests pass)

```
[cs8bxx@ieng6-201]:~:$ java TestReverseRecurse
Maximum number of integers you wish to enter? -5
You must enter a value > 0; Try again.

Maximum number of integers you wish to enter? 0
You must enter a value > 0; Try again.

Maximum number of integers you wish to enter?
```
Enter up to 42 integers:
-9 4 33
88 23 53 76
55
-83
7757

<-------- User enters <Ctrl>+D here to indicate EOF

The original array:
-9 4 33 88 23 53 76 55 -83 7757

The array reversed (manipulating array directly):
7757 -83 55 76 53 23 88 33 4 -9

The array reversed again (manipulating array directly)
(should be back in original order):
-9 4 33 88 23 53 76 55 -83 7757

The copy of the original array:
-9 4 33 88 23 53 76 55 -83 7757

The array reversed (reversed array returned vs. direct manipulation):
7757 -83 55 76 53 23 88 33 4 -9

The original array showing original NOT modified:
-9 4 33 88 23 53 76 55 -83 7757

Testing reverse method (direct manipulation) with null array input:
SUCCESS: No NullPointerException thrown.

Testing with reverse method (returned reversed array) with null array input:
SUCCESS: No NullPointerException thrown.
[cs8bxx@ieng6-201]:~:$

Note the user entered <Ctrl>+D before all 42 integers were entered to indicate no more input. The array
returned by initArray() is sized to the number of actual integers entered.

[cs8bxx@ieng6-201]:~:$ java TestReverseRecurse
Maximum number of integers you wish to enter?
6

Enter up to 6 integers:
ttt

The original array:
Empty array

The array reversed (manipulating array directly):
Empty array
The array reversed again (manipulating array directly) 
(should be back in original order):
Empty array

The copy of the original array:
Empty array

The array reversed (reversed array returned vs. direct manipulation):
Empty array

The original array showing original NOT modified:
Empty array

Testing reverse method (direct manipulation) with null array input:
SUCCESS: No NullPointerException thrown.

Testing with reverse method (returned reversed array) with null array input:
SUCCESS: No NullPointerException thrown.
[cs8bxx@ieng6-201]:~:519$

Note any non-integer input (like "ttt") will terminate the input process. Again, see the example above on how to 
use Scanner and check for and read an integer from the console.

[cs8baxx@ieng6-201]:~:519$ java TestReverseRecurse
Maximum number of integers you wish to enter?
LOL
[cs8baxx@ieng6-201]:~:519$

Note any non-integer input (like "ttt") will terminate the program.

[cs8baxx@ieng6-201]:~:$ java TestReverseRecurse
Maximum number of integers you wish to enter?
5

Enter up to 5 integers:
1 2 3 4 5 6 7 8 9 10

The original array:
1 2 3 4 5

The array reversed (manipulating array directly):
5 4 3 2 1

The array reversed again (manipulating array directly) 
(should be back in original order):
1 2 3 4 5

The copy of the original array:
1 2 3 4 5
The array reversed (reversed array returned vs. direct manipulation):
5 4 3 2 1

The original array showing original NOT modified:
1 2 3 4 5

Testing reverse method (direct manipulation) with null array input:
   SUCCESS: No NullPointerException thrown.

Testing with reverse method (returned reversed array) with null array input:
   SUCCESS: No NullPointerException thrown.

[cs8baxx@ieng6-201]:~:$

[cs8baxx@ieng6-201]:~:$ java TestReverseRecurse
Maximum number of integers wish to enter?
5

Enter up to 5 integers:
11
22
33
44
55
<-------- Stops reading input after the 5th input
The original array:
11 22 33 44 55

The array reversed (manipulating array directly):
55 44 33 22 11

The array reversed again (manipulating array directly)
(should be back in original order):
11 22 33 44 55

The copy of the original array:
11 22 33 44 55

The array reversed (reversed array returned vs. direct manipulation):
55 44 33 22 11

The original array showing original NOT modified:
11 22 33 44 55

Testing reverse method (direct manipulation) with null array input:
   SUCCESS: No NullPointerException thrown.

Testing with reverse method (returned reversed array) with null array input:
   SUCCESS: No NullPointerException thrown.

[cs8baxx@ieng6-201]:~:$
Here is also an example where the null array input tests do NOT pass (and should be fixed):

```
[cs8baxx@ieng6-201]:~:$ java TestReverseRecurse
Maximum number of integers wish to enter?
2

Enter up to 2 integers:
1 2

The original array:
1 2

The array reversed (manipulating array directly):
2 1

The array reversed again (manipulating array directly)
(should be back in original order):
1 2

The copy of the original array:
1 2

The array reversed (reversed array returned vs. direct manipulation):
2 1

The original array showing original NOT modified:
1 2

Testing reverse method (direct manipulation) with null array input:
   FAIL: NullPointerException – Fix me!

Testing with reverse method (returned reversed array) with null array input:
   FAIL: NullPointerException – Fix me!
[cs8baxx@ieng6-201]:~:$
```

Note in the two examples above initArray() should only accept the number of integers specified by the user. This should all just work using something similar to the Scanner example above.

**Turnin**

To turnin your code, navigate to your home directory and run the following command:

```
> cse8btturnin pa6
```

You may turn in your programming assignment as many times as you like. The last submission you turn in before the deadline is the one that we will collect. Always recompile and run your program right before turning it in, just in case you commented out some code by mistake.
Verify

To verify a previously turned in assignment,

> cse8bverify pa6

If you are unsure your program has been turned in, use the verify command. We will not take any late files you forgot to turn in. Verify will help you check which files you have successfully submitted. It is your responsibility to make sure you properly turned in your assignment.

Files to be collected:
- README
- MyCritter.java
- ReverseRecurse.java

The files that you turn in must be EXACTLY the same name as those above.
(All other *.java files in your directory will also be collected as part of the turnin)

Extra Credit
5 points (5%) of extra credit will be given for successfully beating the Hard Bot in Critters.

NO LATE ASSIGNMENTS ACCEPTED.
DO NOT EMAIL US YOUR ASSIGNMENT!
Start Early and Often!