Name _____ Lecture Section _____ TA _____

Login name _____

- Turn off your cell phone to avoid losing a letter grade when it rings.
- Use the amount of space provided to gauge how much you should write. Brevity is the soul of points. Points are not related to wit.
- Legibility counts, so be neat. If your writing is smaller than the typeface of this exam, I may deduct points.
- Points may be deducted for irrelevant, meaningless, or contradictory statements (and of course, just plain false statements). Please be sure to answer the question I asked!
- Do not complicate an example. Do not make up features of an example unless directed to do so. Simple is best! What migratory bird travels thousands of miles and then needs to refuel on Delaware shores?
- Some questions look hard at first, but if you for(i=0;i<3;i++) breatheDeeply(); you realize it is simpler than you first thought.
- **Do not change code I have written** unless explicitly directed to do so..

1. (6 pts) Java has interfaces Comparable and Comparator. As discussed in class, what is the main advantage of Comparator?

2. (6 pts) What is the signature (type, name, parameters) of the method that all Comparator objects must have?

   _____

3. (6 pts) What is the signature (type, name, parameters) of the method that all Comparable objects must have?

   _____

4. (6 pts) We can use a comparator in our own code, but often we use our comparators with what useful Java class?

   _____

5. (5 pts) Name two diverse habitats that occur in estuaries, according to the presentation from class:

   _____ _____

6. (5 pts) What is **one** topic that NERR is addressing regarding coastal resiliency?

7. (6 pts) To place a file in the (shared) git repository (as discussed), we use (one word)

   _____

8. (6 pts) To place a file on the local git stage (as discussed in class), we use (one word)

   _____

9. (6 pts) When we pass an object as a parameter to use one of its methods, that is an example of

   _____

10. The principle of modularity suggests that programs should be in convenient pieces for easy re-use, debugging, and maintenance. Which design pattern discussed in class suggests a way to decompose some applications?

    _____

11. Write one JUnit test for whether multiply properly handles “5 * 0”.

```
package main;
public class MyClass {
        public static int multiply(int i, int j) {
                return 0;
        }
}

//////////different file/////////////

package test;
import static org.junit.Assert.assertEquals;
import org.junit.Test;
import main.MyClass;

public class MyClassTest {









}
```

12. Read the entire problem before you begin. Use indenting AND braces to show me where each class or method or block begins and ends. Really.

    The code on these two pages will produce a Cow class that can be **sorted two ways** - by name (the so-called "natural" order) or by number of legs. Cows are designed to be stored in a collection, and as such they provide one method that operates on collections of cows.

    Write a class Cow with name and number of legs as attributes, and a constructor that initializes both. Add **only** three other methods: one to allow easy printing; one to perform a collection calculation (average); one to define the natural ordering of cows (use the fact that String implements Comparable).

    See **main()** on next page before coding. Use **good Java style** (as discussed in class) to receive full credit.

13. (15 pts)

```
public class Cow
{

    public String name; //public for exam purposes
    public int numLegs;
```

(a) //Constructor

(b) //Method to enable simple printing

(c) //Code to enable sorting by name (do not sort here)

} //END OF Cow CLASS

(d) (8 pts) //Code **outside Cow** to enable sorting by number of legs:

(e) (25 pts) Complete the main() for a different class (not Cow), following the comments and using the space provided. Assume that standard packages you need are imported.

```
public static void main(String[] args){
```

//Make a list of exactly two cows[1]

//Use Java's built-in sorting capability to sort the cows by name

//Use Java's built-in sorting capability to sort the cows by number of legs

//Print the list of cows, using as little code as possible.

} //end of main()

Do not write answers below this line.

[1] Use good cow names.