CS 201, Summer 2015 — August 13th — Final Exam   Name: ____________________

Question 1. [5 points] What output is printed by the following program (which begins on the left and continues on the right)?

```
public class Q1 {
  public static void f(
        String a, String b) {
    String temp = a;
    a = b;
    b = temp;
  }
```

```
  public static void main(
        String[] args) {
    String p = "Boy";
    String q = "Howdy";
    f(p, q);
    System.out.println(p + " " + q);
  }
}
```

Question 2. [5 points] What output is printed by the following program (which begins on the left and continues on the right)?

```
public class Q2 {
  public String s;

  public Q2(String s)
    { this.s = s; }

  public static void g(
        Q2 a, Q2 b) {
    String tmp = a.s;
    a.s = b.s;
    b.s = tmp;
  }
```

```
  public static void main(
        String[] args) {
    Q2 x = new Q2("Oh");
    Q2 y = new Q2("Yeah");

    g(x, y);
    System.out.println(
      x.s + " " + y.s);
  }
}
```

Question 3. [5 points] What output is printed by the following program (which begins on the left and continues on the right)?

```
public class Q3 {
  public static void f(
      int[] a, int [] b) {
    int[] tmp = a;
    a = b;
    b = tmp;
  }
```

```
  public static void main(
      String[] args) {
    int[] x = new int[]{1, 2};
    int[] y = new int[]{3, 4};
    f(x, y);
    System.out.printf("%d %d\n",
      x[0], y[0]);
  }
}
```

Question 4. [5 points] Complete the following static method. It should return the index of the *last* occurrence of `val` in the given array (`arr`). As a special case, if `arr` does not contain any elements equal to `val`, it should return -1.

Here are some example JUnit tests (which assume that the `findLastOccurrence` method is in a class called `Q4`):

```
String[] sarr = new String[]{"A", "B", "A", "C", "A", "B"};
assertEquals(5, Q4.findLastOccurrence(sarr, "B"));
assertEquals(4, Q4.findLastOccurrence(sarr, "A"));
assertEquals(3, Q4.findLastOccurrence(sarr, "C"));
assertEquals(-1, Q4.findLastOccurrence(sarr, "D"));
```

Hint: Think about how to compare `val` to the elements of the array.

```
public static<E> int findLastOccurrence(E[] arr, E val) {
```

Question 5. [5 points] Consider the following method:

```
public static int countLinesInFile(String fileName) {
  try {
    FileReader fr = new FileReader(fileName);
    BufferedReader br = new BufferedReader(fr);
    int count = 0;
    while (br.readLine() != null) {
      count++;
    }
    br.close();
    return count;
  } catch (IOException e) {
    return -1;
  }
}
```

(a) Explain how it is possible that this method might open a file without closing it.

(b) Explain how to modify the method so that the file is guaranteed to be closed (if it is opened).

Question 6. [5 points] What output is printed by the following code?

```
Integer a = new Integer(42);
Integer b = new Integer(42);

if (a == b) {
    System.out.println("first");
}
if (a.equals(b)) {
    System.out.println("second");
}
```

Question 7. [5 points] What is the big-O upper bound on the worst-case running time of the following method? The problem size N is the length of the array passed as a parameter to the method. Explain your answer briefly.

```
public static void int mystery(int[] a) {
    int sum = 0;

    for (int j = 0; j < a.length; j++) {
        for (int i = 0; i < a.length; i++) {
            if (a[i] * a[j] == 1000000) {
                return -1;
            } else {
                sum += a[i] * a[j];
            }
        }
    }
    return sum;
}
```

Question 8. [5 points] Consider the following method (which begins on the left and continues on the right:

```
public static void prependEvens(
    List<Integer> list) {
  List<Integer> evens =
    new LinkedList<Integer>();

  // get all even values
  // and put them in evens list
  for (Integer v : list) {
    if (v % 2 == 0) {
      evens.add(v);
    }
  }
```

```
  // prepend the even values
  // to the list
  for (Integer e : evens) {
    // add at index 0, prepending e to
    // become the first element of list
    list.add(0, e);
  }
}
```

Let the problem size N be the number of elements in the parameter (`list`).

(a) What is the big-O upper bound of the running time if the list parameter is an instance of `ArrayList`? Explain briefly.

(b) What is the big-O upper bound of the running time if the list parameter is an instance of `LinkedList`? Explain briefly.

Question 9. [5 points] Consider the following classes:

```java
public class IntBox {
  private int val;

  public IntBox(int val)
    { this.val = val; }

  public int getVal()
    { return val; }
}
```

```java
public class IntPair
       extends IntBox {
  private int val2;

  public IntPair(
       int val, int val2) {
    TODO
  }

  public int getVal2() {
    return val2;
  }
}
```

Below, specify what code could be substituted for `TODO` so that the assertions in the following JUnit test code will succeed:

```java
IntPair p = new IntPair(17, 42);
IntPair p2 = new IntPair(23, 79);

assertEquals(17, p.getVal());
assertEquals(42, p.getVal2());

assertEquals(23, p2.getVal());
assertEquals(79, p2.getVal2());
```

Note that you are only specifying statements to replace `TODO`. You may not modify either class in any way.

Question 10. [5 points] Complete the following static method. It should return the number of elements of the given list which compare as either greater than the value `max` or less than the value `min`. The element type `E` is guaranteed to implement the `Comparable` interface.

The following JUnit test demonstrates the method:

```
List<Integer> list =
    Arrays.asList(41, 33, 26, 19, 34, 32, 32, 44, 19, 10);

// Count number of elements greater than 35 or less than 15
// (should count just the elements 41, 44, and 10)
int count = Q10.countElementsOutsideRange(list, 35, 15);
assertEquals(3, count);
```

Hints:

- Think about how the `Comparable` interface will help you compare the list elements to the `max` and `min` values

```
public static<E extends Comparable<E>>
int countElementsOutsideRange(List<E> list, E max, E min) {
```

Programming Questions

To get started, use a web browser to download the zipfile as specified by your instructor. Import it as an Eclipse project using File → Import... → General → Existing Projects into Workspace → Archive file.

Important: You may use the following resources:

- The textbook
- The lecture notes posted on the course web page
- Your previous labs and assignments

Do not open any other files, web pages, etc.

Question 11. [10 points] In the class `Q11`, implement the `findNthLargest` method. Given an array list and an integer `n`, it should return the `n`th largest value in the array, as determined by the given comparator object. For example: if `n` is 1, then the overall largest value in the array should be returned, if `n` is 2, then the second-largest value should be returned, etc.

Hint: Use either the `Arrays.sort` or `Collections.sort` method in your implementation.

Validate your method by running the `Q11Test` class as a JUnit test. Make sure that all of the tests pass.

Question 12. [15 points] In the class `Q12`, complete the definition of the generic method called `collate`. Given a `List` object, the method should rearrange its contents so that the elements with even indices are placed at the beginning of the list, and the elements with odd indices are placed at the end of the list. In other words, if a list containing elements `A B C D E F` is passed to the method, the list should be re-arranged so that the elements are `A C E B D F`.

Hints/specifications:

- The list is generic, and the elements have type `E`
- The method must modify the list passed as the parameter
- Suggested approach: create a temporary list, places the result values in it (in the correct order), then clear the original list and add the result values to it
- The `clear` method will clear a list (cause it to become empty), and the `addAll` can be used to add all values in a source collection to a destination collection

Validate your method by running the `Q12Test` class as a JUnit test. Make sure that all of the tests pass.

Question 13. [15 points] *Using recursion*, complete the definition of the method `insertCommas` in the class `Q13`. Given a string as a parameter, the method should return a string in which a comma is inserted between each pair of adjacent characters in the original string. For example, if the string `Hello` is passed to the method, it should return the string `H,e,l,l,o`.

Hints:

- Your solution *must* be recursion: do not use a loop
- Define an appropriate base case
- When solving a subproblem recursively, make sure it works towards a base case
- Think about how to extend the solution to the subproblem so that it is a solution to the overall problem
- The `charAt` method returns the character at a specified index (0 for the first character in the string)
- The `substring` method returns a substring with all characters from a specified start index (inclusive) to a specified end index (exclusive); if the end index is omitted, the returned substring contains all characters from the start index to the end of the string
- Use string concatenation to build the result string

Validate your method by running the `Q13Test` class as a JUnit test. Make sure that all of the tests pass.